User Stories in Practice: A Distributed Cognition Perspective

Karen Chance

A thesis submitted to
Auckland University of Technology
in partial fulfilment of the requirements for the degree of
Master of Computing and Information Sciences (MCIS)

2011

School of Computer and Mathematical Sciences

Primary supervisor: Jim Buchan
Secondary supervisor: Anne Philpott

Table of Contents

Table of Figures

Table of Tables

Abstract

User stories are a simple and direct way of creating and managing software requirements. They are widely used in contemporary practice. This widespread adoption suggests they are providing some benefits when compared to previous methods of managing requirements. But are these benefits realised in practice? If so, can these benefits be explained?

This thesis seeks to address these questions by studying how software development teams use user stories in practice and identifying benefits and challenges. The approach proposed in this thesis is to take the view of user stories as part of a Distributed Cognition system that has a goal of developing requirements and implementing these in a software application. In this perspective, the Distributed Cognition system can include any interactions between people, artefacts and aspects of the environment that contribute to these cognitive goals. Distributed Cognition theory is then applied to the user story process to identify cognitive activities distributed through the system and provide some insights and explanations related to practice, benefits and challenges from this perspective. Examples of these are:

- As the cards are of roughly equal units of effort, they can be used to illustrate sprint status;
- Using the story wall changes what had been an intensive cognitive process into a perceptual one;
- Keeping the wall meaningful – up-to-date and easily perceivable.

In addition, this thesis documents the user story process from semi-structured interviews with practitioners within an organisation. This detailed description will provide data for a “light” Distributed Cognition analysis compared to the traditional ethnographic approach used by other Distributed Cognition researchers. It is a “light” Distributed Cognition analysis in the sense that the analysis of the interview data limited in the cognitive interactions that are uncovered compared to data based on observation of the actual events. This thesis, however, posits that by appropriate interview question design and the flexibility offered by follow-up questions, the interviews will provide

sufficiently rich data on the Distributed Cognition interactions between people, artefacts (such as user stories) and the environment to provide insights into the user story process from a Distributed Cognition perspective. This "light" approach has the major advantage of being less obtrusive than observation, and so, generally, has a lower acceptance threshold for participating organisations. This study used the Distributed Cognition of Teamwork method to develop three models –Information Flow, Artefact and Physical. Part of this study is a review of the limits of the interview data for a Distributed Cognition analysis. The results suggest that interview data for the Information Flow and Artefact themes is adequate to provide insights into how this organisation uses user stories. However, the Physical theme is less detailed.

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Acknowledgements

I wish to acknowledge and thank my supervisory team, Jim Buchan and Anne Philpott for the substantial efforts they have made on my behalf in the past year.

I also wish to acknowledge and thank the members of AT who have participated in this study.

Ethics application number 10/261 and date of approval 8 November 2010

1 INTRODUCTION

1.1 Background

User stories were first introduced by Beck (2000) in his description of Extreme Programming (XP) as a simple and direct way of creating and managing software requirements. Since then they have been adopted outside the XP context as a mechanism to manage requirements and as a unit of planning (see, for example, Lean Software Development (Hibbs, Jewett, & Sullivan, 2009, p. 86; Poppendieck & Poppendieck, 2003, p. 79), Scrum (Mar & Schwaber, 2002)). User stories are now widely used in contemporary practice (Read, Callens, Nguyen, & de Vreede, 2011; Hugh Robinson & Sharp, 2010; Rodríguez, Yagüe, Alarcón, & Garbajosa, 2009).

This widespread adoption within the agile community suggests that they are providing some benefits when compared to previous methods of managing requirements. Certainly, proponents of agile methods claim a number of benefits to the use of user stories compared to a traditional approach to requirements management (Cohn, 2004; Lindvall et al., 2002). But are these benefits realised in practice? If so, can these benefits be explained?

This thesis seeks to address these questions to some extent by studying how software development teams use user stories in practice and identifying benefits and challenges. The approach proposed in this thesis is to take the view of user stories as part of a Distributed Cognition system that has a goal of developing requirements and implementing these into a software application. In this perspective, the Distributed Cognition system can include interactions between people, artefacts and aspects of the environment that contribute to these cognitive goals (Hollan, Hutchins, & Kirsh, 2000). Distributed Cognition theory is then applied to the user story process to identify cognitive activities distributed through the system and provide some insights and explanations related to practice, benefits and challenges from this perspective.

In this thesis, in order to apply a Distributed Cognition analysis to the user story process in practice, the user story process within an organisation is documented from semi-structured interviews with practitioners within this organisation. This will provide data

for a “light” Distributed Cognition analysis compared to the traditional ethnographic approach used by other Distributed Cognition researchers. It is a “light” Distributed Cognition analysis in the sense that the analysis of the interview data is likely to be limited in the cognitive interactions that are uncovered compared to data based on observation of actual events. For example, it is expected that the understanding of the role of the physical environment in the Distributed Cognition system could be less rich from analysing interview data (remembered perceptions of participants) than it would be from analysing sustained observation of the practice. This thesis, however, posits that by appropriate interview question design and the flexibility offered by follow-up questions, the interviews will provide sufficiently rich data on the Distributed Cognition interactions between people, artefacts (such as user stories) and the environment to provide insights into the user story process from a Distributed Cognition perspective. This “light” approach has the major advantage of being less obtrusive than observation, and so, generally, has a lower acceptance threshold for participating organisations. Part of this study will be a review of the limits of interview data for a Distributed Cognition analysis.

The next subsection provides some background to the use of user stories in the context of requirements engineering and the development of agile methodologies. It shows that user stories are part of the reaction to challenges with traditional requirements methods and highlights some of the claimed benefits. This leads to a discussion of the lack of empirical understanding of user stories in practice.

1.2 The Rise of the User Story

The importance of software requirements in software development was first recognised by Bell and Thayer in 1976. Previously, software professionals spent scant attention to requirements. They believed they were “correct by definition” and arose “naturally in response to a need” (Bell & Thayer, 1976, p. 61). One can see this assumption in the prevailing waterfall method (Royce, 1970). The widespread assumption was that not only are problems specifiable, but optimal solutions exist that can be specified in advance (Dybå & Dingsøyr, 2008). Bell and Thayer (1976) not only identify the importance of software requirements to the development process, they acknowledge they were the source of many of its problems.

Bell and Thayer's (1976) suggestion to develop "improved techniques" for requirements specification began a quest for tools, methods and methodologies to solve the problem with requirements (p. 66). None proved to be what Frederick Brooks calls the "silver bullet" (Brooks, 1987, p. 10). Brooks (1987) acknowledges that this quest has met with some success in reducing the accidental problems in software engineering (SE). At the same time Brooks (1987) points out these successes have not changed the essential nature of software development, of which the requirements process is the "hardest single part" (p. 18). While remaining sceptical they are "silver bullets," Brooks (1987) lists several promising avenues to explore, such as object-oriented software development, incremental development and rapid prototyping (p. 15-8).

As Brooks (1987) foresaw, these avenues did not change the essential nature of software development. In his research retrospective, van Lamsweerde (2000) points out that despite twenty-five years of research since Bell and Thayer's article, "much remains to be done" (p. 15). Recent surveys by both Cheng and Atlee (2009) and Hansen, Berente et al. (2009) confirm the continued relevance of this view. Cheng and Atlee (2009, p. 33) note that the increasing complexity of application domains as well as the complexity of modern computing and cyber-infrastructure raise new issues for requirements engineering. Hansen, Berente et al. (2009) also identify continuing challenges to requirements engineering resulting from complexity in infrastructures, organisations, and environments (p. 74).

In the mid-1990s, a new software development movement emerged from practice. Calling their movement "agile," these practitioners recognized the value of what Brooks (1987) saw as promising techniques. They saw how these and others work in practice. How these agile practitioners differed from their contemporaries is that instead of using these techniques within a traditional plan-based framework, they made these techniques the bases for new frameworks. Their methodologies emerged from what they found works in practice.

Kent Beck (2000) exemplifies this approach: "If code reviews are good, we'll review code all the time (pair programming)" (Beck, 2000, p. xv). Beck's methodology emerges by taking techniques to an extreme.

The user story is another of Beck's good techniques taken to an extreme. In Beck's view, user stories are initially written by the business stakeholders to capture short descriptions of their needs in simple natural language, usually on index cards. The story cards then go through a process of continuous improvement in collaboration with the software development team. This process includes elaboration, prioritisation, estimation and development of acceptance criteria.

The user story, however, is more than the story card. As Cohn puts it "the important parts are the conversations between customers and developers about the story" (2004, p. 15). As well as encouraging verbal communication, the story is used for iteration and release planning. At the point in which software development begins, the story cards are often used to manage development through a "story wall," usually mounted on a wall of the office.

There are a number of claimed benefits to the use of user stories as described here. These benefits relate to their simplicity and public nature and include:

- By emphasising high bandwidth verbal communication, they help share understanding effectively (Cohn, 2004, p. 145; Monochristou and Vlachopoulou, 2007);
- They are written in natural language and so are easy for the user to express, verify and prioritise (Cohn, 2004, p. 145);
- Their small size makes them easily estimable (Cohn, 2004, p. 145; Monochristou and Vlachopoulou, 2007);
- They encourage collaboration (Cohn, 2004, p. 145);
- By deferring detail they better cope with change in understanding of problem domain (Beck, 2000; Cohn, 2004, p. 145).

When the story cards are used in the story wall there are a number of further benefits claimed by proponents. These include:

- It is large, easily visible, and understood at a glance (Cockburn, 2004, p. 73);
- It is updated on a regular basis to motivate passers-by to attend (Cockburn, 2004, p. 73);

- Shows people information without having to ask anyone a question (Cockburn, 2004, p. 73; Sharp & Robinson, 2006a);
- Provides a lightweight traceability (Delgadillo & Gotel, 2007).

Despite the stated benefits and widespread adoption, it is evident, after an extensive literature review that little is empirically known about the user story process itself and its benefits and challenges in practice.

When user stories are studied, they are often examined as a component of a wider study of agile processes (Abrahamsson & Koskela, 2004; Layman, Williams, & Cunningham, 2004). Other researchers investigate sub-tasks of the user story process, such as prioritising, estimating, or identifying dependencies (Aiello, Alessi, Cossentino, Urso, & Vella, 2007; Gomez, Rueda, & Alarcón, 2010; Haugen, 2006).

This thesis, however, focuses on investigating user stories specifically, in order to understand how they are used in practice. This focus on studying user stories and understanding their use in practice is discussed in more detail in the next subsection.

1.3 Research Focus

The focus is on investigating the user story card as an artefact and how people interact with this artefact to achieve wider requirements-related tasks and goals. Clearly the function of the user story is central to the development process and the communication between business stakeholders and the software development team. Studying their function and use in practice as objects instantiated in the world provides an understanding of their value and benefits, and may suggest ways to amplify or improve these. For example, how does the physical, tangible nature of the story card support cognitive activities related to software development? The function of the user story extends to their role as a component of the story wall. For instance, how is their meaning changed and used as part of the story wall?

One significant contribution to the empirical research on user stories is the work of Sharp & Robinson. In 2006a, in their study of a mature XP team, they identify the user story artefact as a coordinating mechanism in a Distributed Cognition system. In 2006b and 2009 they focus more specifically on understanding the role of story cards and the

story wall as artefacts that are part of the information flowing in, around and among the team. They develop a model of the consequences of changes to the process. The study in this thesis takes a similar approach, with the emphasis on user stories as an artefact in a distributed cognitive system doing collaborative work. Rather than emphasising the *team* in the context of XP, however, the investigation in this thesis studies the *process* used during software development.

For an examination of these topics, Sharp and Robinson's (2006b) work suggests Distributed Cognition theory as an interesting lens (Hollan et al., 2000; Hutchins, 1995a, 1995b). User stories are conceptualised as part of a complex cognitive system that includes "collections of individuals and artefacts that participate in the performance of a task" (Flor & Hutchins, 1991, p. 36). As user stories are instantiated in the user story process as story cards, any such lens that focuses on artefacts as a central focus of a cognitive system is of interest.

A more comprehensive understanding of the process of software development will provide insights into why certain techniques and tools may be successful and suggest possible improvements. Understanding user story practice will enable practitioners to amplify those aspects found to be successful and identify those that are not.

An additional goal of this study is exploring the use of interview data for a Distributed Cognition analysis. Although most Distributed Cognition studies use ethnographic methods, including observation, this study's use of interview data to explore the user story process will also be of interest to the research community as a more accessible alternative.

1.4 Research Aim

The research aim of this study is to deepen the understanding of user stories in practice using Distributed Cognition theory to analyse the user story process. Studying this phenomenon is not an end in itself. The goal of this research is to explain and analyse the role of a user story in software development from a shared cognitive perspective.

In addition, as an empirical study, it aims to provide what Salo and Abrahamsson (2004) call "a scientific and thus more rational basis" for examining the value of "tools, methods and techniques" in software development (p. 409).

This purpose is of interest both to the research community and software practitioners. It will help researchers by exploring Distributed Cognition through interviews.

1.5 Research Questions

In order to achieve this aim, this research project will investigate the question: How is a user story involved in software development as an artefact in a Distributed Cognition system?

Secondary questions are:

- What is the software development process and how are user stories involved?
- What distributed cognitive activities involve user stories as a cognitive artefact?
- How do user stories support Distributed Cognition in terms of sharing understanding?
- How do user stories support Distributed Cognition in terms of communication?
- What are the benefits and challenges of using user stories in a Distributed Cognition system?

In this thesis, both stakeholders and users representing the business and the software development team are involved in software development. The business side includes those who "make decisions about what the system will be" (Beck, 2000, p. 89), including the project champion, stakeholders, business domain subject matter experts (SMEs), and end users. The software development team are those "responsible for implementing the system" (Beck, 2000, p. 89). These people include the project manager, business analysts, technical SMEs, technical architects, developers and testers. Essential to the agile approach is that all these roles are responsible for the development of the software system.

This thesis explores cognition distributed through the people involved in developing software, including stakeholders and users and members of a software development team. In addition, the cognition of these people is enhanced by interaction with the environment and/or artefacts, or developed through time (Hollan et al., 2000). Distributed Cognition advances that cognitive activities are not restricted to "the bounds of skin and skull" (Clark, 2008, p. 76).

1.6 Research Approach

This thesis reports on an interpretive case study; it is a case study that interprets the singular subject of the user story process as a Distributed Cognition system in a New Zealand environment.

This research project will look for rich communication, information flow and shared understanding among the software development teams of a New Zealand enterprise through a Distributed Cognition lens. There has been considerable research in the field of Distributed Cognition. Flor and Hutchins (1991) offer the earliest Distributed Cognition analysis in the field of software development. Observing two maintenance programmers making a change to a game program, Flor and Hutchins (1991) find that "successful software development is a consequence of an interaction of programmers and development artefacts in a system of Distributed Cognition".

Aranda and Easterbrook (2006) advocate the use of Distributed Cognition as a tool in SE research. They list four reasons why Distributed Cognition is particularly appropriate as a research approach:

- It centres on cognitive activities in the field;
- It views artefacts as embodied knowledge;
- It keys in on information flows;
- It analyses cognitive processes both as "resolutions of cognitive problems" and "learning and structuring activities" (Aranda & Easterbrook, 2006).

At about the same time, Helen Sharp and Hugh Robinson begin their work examining agile software development through a series of observational studies. In these ethnographically-informed case studies, they probe their data from a variety of perspectives. Three of the studies are Distributed Cognition analyses (Sharp & Robinson, 2006a; Sharp, Robinson, & Petre, 2009; Sharp, Robinson, Segal, & Furniss, 2006b). They examine mature XP teams and the role of artefacts such as story cards and the Wall in agile software development.

This thesis is a case study focusing on the role of user stories as artefacts in a distributed cognitive system. Investigating this system includes studying interactions involving the

people, other artefacts and the environment concerned with the creation and management of user stories, including the story wall. The emphasis of this study will be on how user stories support those cognitive processes that result in shared understanding through communication and information flow.

Information about the interactions and the user story process, as a basis for the Distributed Cognition analysis, will be based on in-depth interviews with members of the development teams. This does not follow the ethnographic approach, involving a sustained period of observation, used by traditional researchers of Distributed Cognition. It is argued that the use of interview data provides some benefits, although it is acknowledged that it is likely to result in a "lighter" Distributed Cognition analysis compared to one using observational data.

The advantages of this approach are two-fold. First, interviews are a lower barrier for recruiting participation in practice. Many organisations for reasons of confidentiality do not want to participate if the data collection method involves observation.

Secondly, as Walsham (1995) points out, observed data is merely the researcher's interpretation of the observed. Walsham echoes Geertz (1993), who points out that "what we call our data are really our own constructions of other people's constructions" (as cited in Walsham, 1995). In exploring the advantages of using interviews as a data collection method, Walsham (1995) applies van Maanen's (1979) definition of first-order and second-order data. The interviewees' interpretation of their experience is first-order data. The importance of this data is not in dispute. The researcher's constructions are second-order data. These second-order interpretations are informed by good theory and can result in insightful analysis (Walsham, 1995).

This research project will focus on the experiences of members of various software development teams in a New Zealand context. These people are self-selected from the IT division of a major financial organization in Auckland, New Zealand. This organization will be referred to as "Company A" henceforth, and the Information Technology (IT) division as "AT".

AT adopted what they refer to as "agile" three and a half years ago. They have developed their own agile methodology, most notably combining aspects of Extreme

Programming (XP) with Scrum project management. AT is a group of mature agile software development teams, using the definition of "mature" as having one year or more of agile experience (Dingsøyr, Dybå, & Abrahamsson, 2008). As such, insights drawn from this environment are not compromised with agile implementation problems. They are insights of practice.

The participants represent a variety of roles in software development teams such as project manager, business analyst, and developer in both Java and legacy programming languages. As it turned out, the product owner role was not represented directly in the interviewed participants because of availability constraints. All participants consented to the semi-structured interviews in writing and agreed to being digitally recorded. The Participant Information Sheet and consent form are in APPENDIX C and APPENDIX D.

The interview questions (APPENDIX B) were designed to encourage the participants to discuss how information is propagated through the team and transformed, focusing on user stories and the story cards on the story wall.

1.7 Novelty and Contribution

This study will extend the work of Sharp and Robinson by using a Distributed Cognition lens to understand the complex cognitive system around user stories. While Sharp and Robinson focus on social interactions of the software development team (Hugh Robinson, Segal, & Sharp, 2007), this study focuses on the user story process and the user story as a cognitive artefact, as understood from interviewing those involved with the creation and analysis of the user stories. In addition, examining user stories in a New Zealand context and non-XP context adds a different perspective to the current research literature.

1.8 Roadmap

The rest of this thesis is divided into six chapters. Following this introductory chapter they are:

LITERATURE REVIEW – Chapter Two will explore the knowledge areas of the research and probe extant research literature for relevance to the research questions;

RESEARCH DESIGN AND IMPLEMENTATION – This chapter establishes both the ontology and epistemology of the research approach taken for this thesis project. It justifies the selection of the research methodology and the data collection and data analysis methods. It defines the research project as an interpretive case study examining the user story process identified from interview data through a Distributed Cognition lens, employing the Distributed Cognition for Teamwork framework;

FINDINGS AND DISCUSSION – This chapter examines the findings from the data. It then continues with a discussion of the findings in terms of the research question and each of the secondary questions. It also compares findings with previous research literature in this area;

CONCLUSION – This chapter reviews the thesis and identifies recommendations, original contributions, limitations and indicating areas of future work.

1.9 Summary

This chapter provides the reader with an introduction to this thesis. It gives the background and motivation to this research. It defines the focus, approach, and aim of the research. These provide a basis for the articulation of the research questions. It briefly touches on the contribution of this research to the practitioner and research communities. It gives a brief description of the research methodology. Finally, it gives an outline of the structure of this thesis.

To support this research study, the next section - the literature review - explores the current practice and theory of agile software development as the context for user stories, user stories themselves, Distributed Cognition, and Distributed Cognition research applied to user stories and SE more generally. It also provides a wider context and vocabulary for this thesis. Lastly, it helps develop “sharper and more insightful questions” (Yin, 2003, p. 9).

2 LITERATURE REVIEW

This literature review will explore current user story practice, empirical research on user stories and finally, research on Distributed Cognition, both in theory and practice.

Section 2.1, “Agile Software Development”, provides a background of what current agile software development practice is, including the user story process as outlined by noted practitioner-authorities. It seeks to establish a theoretical framework for how user stories are used in practice, define a vocabulary and scope of user stories to construct a model with which to compare the AT user story process. This typical user story process section is followed by Sections 2.1.2, exploring the “Stated Benefits” and 2.1.3 “Challenges and Limitations”. These will define a theoretical basis for examining the subsidiary research questions in the research findings section.

Secondly, the literature review seeks to ground this thesis in empirical research in the Section 2.2, “What is Empirically Known about User Stories”. It explores both theory and methodology. It starts with the landmark survey of Dybå and Dingsøyr (2008) to explore earlier empirical agile research. This review then explores more recent empirical research involving user stories, in which the work of Sharp and Robinson (Hugh Robinson et al., 2007; Sharp & Robinson, 2006a; Sharp et al., 2009; Sharp et al., 2006b) helps, as Yin (2003) suggests, to sharpen the research questions.

The literature review concludes by examining the theoretical lens proposed in this thesis to analyse the user story process, namely Distributed Cognition theory. In 2.3, the final section of this literature review, Distributed Cognition theory is defined and its use justified through the work of founder Edwin Hutchins (1995a, 1995b) and others (Aranda & Easterbrook, 2006; Flor & Hutchins, 1991; Rogers, 1997). The framework of analysis used in this thesis, Distributed Cognition for Teamwork (DiCoT) is described in Section 2.3.1.

2.1 Agile Software Development

Traditionally, software development methodologies have been plan-driven. These methodologies, most notably the waterfall method, create highly-detailed requirements documents. The methodologies divide the software development process into phases

that end with a transition document. Those who create this document generally pass it to a different group of people who add more detail to the plan. Often months after the project has begun, a requirements specification is handed to a software development team to develop.

These serial “hand-offs” of the requirements specification often lead to problems. The distance of time and proximity between the people who want the software and those who create it can result in incorrect assumptions, misunderstandings and miscommunications. In addition, requirements change. The people who want the requirements may realise that they made mistakes, or they may better understand what they want.

There were persistent problems with creating software this way. In the mid-90s, a group of practitioners decided to invert the usual way of creating a software development methodology. As practitioners, they knew what works in practice. The problem was trying to graft these techniques onto the prevailing plan methodologies. These practitioners began to develop frameworks from the methods they had found to work.

This practical movement gained momentum around the turn of the century. Characterising their movement as “agile” - “quick-moving,” “nimble,” and “active” (Sykes, 1986, p. 18) - these proponents issued a manifesto of their values:

- Individuals and interactions are more important than processes and tools;
- Working software is more important than documentation;
- Customer collaboration is valued more than contract negotiation;
- It is more important to respond to change than follow a plan (Beck et al., 2001).

Widely acknowledged as the first explicit agile methodology (Abrahamsson, Salo, Ronkainen, & Warsta, 2002), Kent Beck (2000) advances his Extreme Programming (XP) methodology as a “lightweight, efficient, low-risk, flexible, predictable, scientific, and fun way to develop software” (p. xvii). Of primary interest to this study is Beck’s user story process (Beck, 2000, p. 89).

User stories have been widely adopted among agile methodologies to capture requirements (Poppendieck and Poppendieck, 2003, p. 79; Hibbs, Jewett et al., 2009, p. 86; Mar and Schwaber, 2002). Characterising user stories as "small atomic bits of functionality," Jeffries, Anderson et al. (2000) define user stories as a "short description of the behaviour of the system, from the point of view of the user of the system" (p. 37). Typically, these user stories are written on index cards.

However, the user story itself is more than a story card. Mike Cohn (2004) states that user stories are composed of three components:

- A written description used for planning and as a reminder;
- Conversations around the story to add detail;
- Tests that determine when the story is complete (p. 4).

Jeffries (2001) calls these components "Card," "Conversation," and "Confirmation". The user story card is a simple and direct method of requirements elicitation and specification. The simplicity of the story card contrasts with the in-depth detail of requirements documents in traditional software development. However, it is not what the story card is that is as important as how it is used. This is the user story process.

2.1.1 How are User Stories Used

In order to characterise the use of user stories in the case organisation to be studied in this thesis, it is useful to focus on what current practice is. It is the aim of this section of the literature review to synthesise a user story process beginning with Beck's (2000) early release planning template and the possible variations in current practice described primarily by Cohn (2004, 2007), Cockburn (2004, 2007), Jeffries (2001), and others. Sprint planning and management is discussed through reference to many of the same works. This typical user story process provides context for the case organisation's user story process to be identified.

Inevitably, such a process may not resemble any used in practice. The user story process is neither intellectual property nor defined by any one person, company or organisation as are standards such as the Rational Unified Process (IBM, 2011), ITIL and Prince2 (United Kingdom Office of Government and Commerce, 2011a, 2011b) and the Unified Modelling Language (Object Management Group, 2009). Indeed, as

the Agile Manifesto encourages adaptation of processes and techniques to suit specific project circumstances, it is not surprising that there is no one user story process (Beck et al., 2001).

In addition, this section will outline the Scrum agile project management methodology, especially in those points where it is commonly used with user stories. Scrum does not define any specific requirements development techniques for the implementation phase, and as such, many practitioners find it complements the user story process (Abrahamsson et al., 2002). In their survey of practitioners, Parsons, Ryu and Lal (2007) confirm the common use of this combination among practitioners.

What Beck refers to as the "Business" works with the software development team throughout the user story process (Beck, 2000, p. 86). This business team consists of stakeholders such as the product owner, the project sponsor, SMEs and users. This thesis will use the term "stakeholders" for the business team. The software development team usually consists of a project manager, business analysts, developers, and testers.

The agile software development process often encompasses project, release and sprint planning. In the first edition of his book, Beck (2000) outlines release and sprint planning as each consisting of three phases: Exploratory, Commitment and Steering (p. 89). As there is no accepted standard of the user story process, Beck's template will guide the discussion of release planning.

The phases and sub-phases of Beck's release planning are depicted in (Adapted from Beck, 2000, p. 89-92)

Table 1. These are discussed in more detail in the next subsection, along with comparisons and contrasts from other "agilists".

Beck's Release Planning	
Phases	**Sub-phases**
Exploration	Write a story
	Estimate a story
	Split a story
Commitment	Sort by value
	Sort by risk
	Set Velocity
	Choose scope
Steering	Iteration
	Recovery
	New Story
	Re-estimate

(Adapted from Beck, 2000, p. 89-92)

Table 1 Release planning

2.1.1.1 Developing User Stories

Exploratory Phase

This phase focuses on less-detailed long-range release planning in which the stakeholders and software development teams "find out what new things the system can do" (Beck, 2000, p. 89). During this phase, the stakeholders write user stories and the software development team estimates the stories. If they find that they cannot estimate a story, the stakeholders revisit the story to split it into smaller stories.

Writing a story

Cohn (2004) suggests a variety of ways to "trawl" for user stories (p. 43). He offers options such as observation, interviews, and questionnaires (Cohn, 2004, p. 45). However, the most common activity of the exploratory phase is the story writing workshop.

If possible, Cohn (2004) suggests that the stakeholders should "include representatives of as many of these user types as practical" (p. 9). However, it is not always practicable

to include members from every identified user group. Instead it is sometimes possible for the stakeholders to develop a "persona" for a missing end user. A persona is an "imaginary representation of a user role" (Cohn, 2004, p. 38). Developing a persona involves creating an imagined biography of a member of the missing end user group. It can be useful to tap into business and demographic data in developing this persona (Cohn, 2004, p. 38).

The goal of a story writing workshop is to "brainstorm as many stories as possible" (Cohn, 2004, p. 33). "Brainstorming" as an activity is "the spontaneous discussion in search for new ideas" (Sykes, 1986, p. 109). Thus, the facilitator of the story writing workshop should encourage spontaneity to aid the generation of new ideas.

One way to do this is to pay attention to the physical environment. The story writing workshop should held in a comfortable space. Moon (2008) suggests using a room with a capacity twice the projected number of participants so there is enough room to move around. Members of the business and software development teams should "intermingle" and not sit "across a table" as if adversaries (Jeffries, Anderson, & Hendrickson, 2000, p. 38). Moon (2008) advises to have handy items such as flipcharts, easels, markers, tape, index cards, candy, and Post-it notes.

Sessions often begin when the facilitator passes out index cards and pens to the stakeholders. The facilitator asks them to write down as many things they want to see in the new software, one user story per card. These user stories should consist of only two or three sentences at most.

Keeping the stakeholders focused and freely creating user stories is a major task for the facilitator. Jeffries suggests starting the session by having everyone scribble on an index card and then tear it up:

> *This gets us all in the frame of mind to touch the cards, to write on them, and to be flexible, ripping up and replacing cards whenever they need it (Jeffries et al., 2000, p. 38).*

Stakeholders thus know that the cards are tools that they can use and they do not have to be correct, especially the first time they write the stories down.

Often the meeting facilitator directs participants to use a format such as "As-a-I-want-so-that" template (Gallardo-Valencia & Sim, 2009). An example of a user story in this format is:

> *As a salesperson, I want an online calculator so that I can give duty-free prices over the phone.*

The "As-a" part of the template focuses on the user role for whom the story is important. In the above example, the perspective is the salesperson. The "I want" is the functionality that is valuable to that role. In the example, the online calculator is a function. However, it is not altogether clear what is to be calculated, if anything. This is where the importance of the "so that" part of the format comes in. The "so that" is both the context of the functionality and its value to the business (Gallardo-Valencia & Sim, 2009). The online calculator must calculate, for example, prices without GST included. Eventually, the acceptance criteria will be derived from the "so that" – what is involved in calculating duty-free prices?

This thesis will refer to this format as the "standard format."

Sessions also can begin by building what Cohn calls a "low-fidelity prototype" that maps "very high level interactions within the planned software" (Cohn, 2004, p. 49). The point of this representation is not to be a persistent design document; it is created to elicit and specify the requirements. Cohn (2004) elaborates: "walking through the workflows will help everyone involved think of as many stories as possible" (p. 51).

During a stakeholder workshop, stories may arise that could possibly distract the stakeholders or divert the flow. In order to maintain the spontaneity, facilitators have ways to "offload" these problems, such as a "Parking Lot", a "Smells" chart and a "Ground Rules" statement (Cohn, 2004, p. 51; Cohn, 2004, p. 157).

A Parking Lot is used to offload problem stories about which there are uncertainties or are sources of contention (Cohn, 2004, p. 38). By buffering these user stories, the flow of the story writing session flow is unimpeded. If any behavioural "ground rules" have been agreed upon, it is useful to have them prominently displayed so the facilitator can refer to them. Lastly, some use a Smells flip chart or whiteboard onto which people

may place user stories that are not "right" – such as stories that are too small or have too many details (Cohn, 2004, p. 157-9).

Estimating the story

To be estimable, the software development team must understand the business value and context of the user story (Beck, 2000, p. 90). If they have any questions, they need to ask the stakeholders (Beck, 2000, p. 90).

At this point, the estimates do not have to be exact, "just enough to help the customer rank and schedule the story's implementation" (Wake, 2003). The software development team throws flags if the user stories are too large – more than a few days – or if they cannot be estimated.

Beck (2000) talks in terms of Ideal Engineering Time for estimates: How long will a task take if the developer did nothing but the task? (p. 90) Others advocate using story points with which a team defines effort as they like, be a story point an hour or a day (Cohn, 2004, p. 87). Cohn (2004) also advises at least two rounds of estimation. Many estimate using story points at a later stage.

Splitting the story

If the story is too large for the software development team to estimate, they ask the stakeholders to split it into smaller stories (Beck, 2000, p. 90). Stories that are too large are often called "epics" (Cohn, 2006, p. 53). The smaller user stories are, the easier they are to estimate. They should also be small enough to do in one sprint (whatever that may be). Lastly, the smaller user stories are, the easier they are to plan. It will be easy to maximise the throughput in a sprint.

Commitment Phase

After the software development team finishes estimating the user stories, the exploratory phase is over. Commitment - the second of Beck's (2000) phases – begins (p. 92). If it has not yet been determined, the software development team and the stakeholders need to settle on how long each sprint will take. The length is determined by the need for the stakeholder feedback: the shorter the sprint, the quicker the

feedback (Beck, 2000, p. 90). The software development team will build a little and then show it to the stakeholders, to see if that is what they really wanted. Usually, a sprint is one to four weeks long (Beck, 2000, p. 133).

Sorting by Value

At this point the stakeholders set the scope of the next release and the software development team commits to delivering it (Beck, 2000, p. 90). The stakeholders sort the cards by value – they will determine the priority of the user stories. Beck (2000) states they sort by:

- Those essential for the system to function;
- Those not as essential but offer significant business value;
- Those that would be nice to have (p. 90).

As befits the practical nature of agile practitioners, many borrow one of the core techniques of the Dynamic Systems Development Method (DSDM) – the MoSCoW technique (Cohn, 2004, p. 98). This technique claims that the meanings of these priorities are explicit:

- "Must have" – Priorities that provide minimum usable functionality;
- "Should have" – Important to the system but not essential;
- "Could have" – If time is left, they can be done;
- "Won't have this time" – These are shifted to the next release (DSDM Consortium, 2008, pp. 64-65).

Sort by Risk

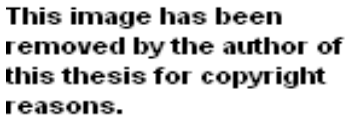
The software development team follow with "sorting by risk" (Beck, 2000, p. 90). The software development team classifies stories as:

- Those they can estimate precisely;
- Those they can estimate reasonably well;
- Those they cannot estimate at all (Beck, 2000, p. 90).

To be estimable, the software development team must understand the business value and context of the user story. If they have any questions, they need to ask the stakeholders to clarify the user story. Thus, there is the need for clear communication and sharing understanding for estimation.

Figure 1 depicts the process by which the software development team "sorts by risk" (Beck, 2000, p. 90). If the members of the software development team feel confident they understand the story, they may estimate it. Otherwise, they may decide the story is too big or too uncertain.

If the story is too big, they ask the stakeholder to split it. If it is too uncertain, it is because they have technical issues with the story. In these cases, they may create a "spike" to investigate the story. According to Wake, spikes are "quick throw-away (and thrown away!) explorations" into possible solutions to technical problems (2000, p. 78). Once the software development team completes the spike, they may estimate the story or ask the customer to split it. Thus, as depicted in Figure 1, the estimation process for a single user story may involve several, potentially repeating steps.



(Adapted from Wake, 2000, p. 103)

Figure 1 Iterative Nature of Story Estimation

Some practitioners use a technique called story card poker. Agile Manifesto signatory James Grenning (2002) created this technique to overcome a problem that frequently occurs during group estimations: a small number of individuals may indulge in endless debate while the rest of the group "drifts off" (Grenning, 2002, p. 1).

Grenning (2002) developed story card poker to circumvent these debates and keep the team focused:

> *The customer reads a story. There is a discussion clarifying the story as necessary. Each programmer writes their estimate on a note card without discussing their estimate. Anticipation builds. Once all programmers have written their estimate, turn over all the cards (Grenning, 2002, p. 1).*

In the unlikely event everyone agrees, they proceed to the next card. If not, the team will have already decided how they will handle disagreement. They may spend a limited amount of time discussing the card. They may accept by default the lowest estimate, they may play another round of story poker, or after discussion they may come to an agreement. In the end, they may decide they need more information and park the user story (Grenning, 2002; Cohn, 2006, p. 58).

One common method to denominate story points is to use the Fibonacci series, which progresses thus: 1, 2, 3, 5, 8, 13, 21…. (Cohn, 2006, p. 52) This sequence is useful for estimating user stories because the gaps between the numbers become larger as the numbers increase. Such a non-linear progression parallels how uncertainty increases with the size of user stories (Cohn, 2006, p. 52; Leffingwell, 2011, p. 110). However, as in most aspects of user story development, teams are free to adopt any standard of story points they like.

Set velocity

At this point, the software development team knows the length of their sprint; they also have an idea of what sizes the stories are. They can estimate their "velocity" (Beck, 2000, p. 90). The velocity is the "ratio between estimated development time and calendar time" (Beck, 2000, p. 72). Essentially, this velocity is an estimate of how much work they can do in a sprint.

Choose scope

With the velocity and reasonable story estimates, the stakeholders can divide the user stories into releases. Stories that are created during the release are put into the product backlog to be developed later.

Steering Phase

The steering phase is carried out during the release itself, to adjust release planning with "what has been learned" by both the stakeholders and the software development team (Beck, 2000, p. 91).

Sprint (Iteration)

Although Beck (2000) calls this sub-phase "iteration," as AT call their iterations "sprints," this thesis will follow in the use of that term. Sprint planning will take place throughout the release. However, the first time sprint planning takes place will be before the sprints begin.

Recovery

As the release progresses, sometimes the software development team needs to re-address their estimate of velocity (Beck, 2000, p. 91). If so, the stakeholders must re-prioritise the stories, sometimes dropping stories into the product backlog.

New story

Throughout the release, the stakeholders may discover a new story is needed. If so, new stories can be added, but stories with an equivalent estimate must be moved into the product backlog (Beck, 2000, p. 91).

Re-estimation

As with the velocity, the software development team may learn better how to estimate the user stories (Beck, 2000, p. 90). In that case, they can re-estimate the remaining stories in the release and adjust the velocity. If so, the stakeholders will need to re-prioritise stories for the rest of the sprint.

The Steering phase concludes Beck's release planning template.

Sprint Planning

While Beck's original release planning structure was useful, his 2000 sprint planning structure cannot be easily adapted. There is considerably less confirmation from other

practitioners. Indeed, the second edition of his book also dispenses with this formal structure of the Planning Game, as he calls it (Beck, 2002).

Having decided on a "set" of cards for the release, the stakeholders must determine the stories that will be developed during each sprint (Beck, 2000, p. 90). Cohn (2004) describes the process as a game: the stakeholders use the story cards to create different stacks based on priority. The size of the piles is limited by the velocity. The stack of index cards with the highest priority forms the user stories for the current sprint, and those for future sprints are the sprint backlog (Cohn, 2004, p. 9-10).

At this point, the software development team takes over and plans the sprint in much the same way they helped plan the release with the stakeholders. One way they plan the sprint is by breaking user stories into programmable tasks (Beck, 2000, p. 91).

Cohn (2006) warns at this point to avoid breaking, for instance, a user story into a user interface task to be implemented in one sprint and a middle tier task in a later sprint. He suggests "delivering a partial user interface, a partial middle tier, and a partial database" (Cohn, 2006, p. 125). This enables the software development team to deliver functionality that the stakeholders can see.

The software development team estimates how long it will take to complete these tasks. At this point, if there is more than one developer involved in the sprint, one of the developers takes responsibility for the task and finalise the estimate (Beck, 2000, p. 91; Cohn, 2004, p. 113-114).

In sprint planning, often a software development team needs to re-prioritise according to dependencies within a sprint. If a user story cannot be developed before another, less highly prioritised user story has been implemented, the less highly prioritised user story must be developed first. However, Cohn (2004) suggests that by building independent stories, dependencies can be reduced (p. 17).

Secondly, there are issues with the development load and balancing. Beck (2000) suggests developers use a "load factor" - the amount of time they are actually developing (p. 93). If all a developer does is code, the load factor is one. Adding up the points each of the developers have accepted and multiplying by the load factor gives

them an indication whether they are overcommitted. They may need to rebalance with their team.

Lastly, these user stories must be testable. "Testable" means that clear acceptance criteria can be applied to the user story. Bill Wake (2003) writes:

> *If the customer doesn't know how to test something, this may indicate that the story isn't clear enough, or that it doesn't reflect something valuable to them, or that the customer just needs help in testing.*

If it passes the tests based on the acceptance criteria, the user story is done.

Cohn (2004) points out that a user story untestable is if it has a non-functional requirement that has not been properly operationalized. He gives an example: "A user must find the software easy to use" (Cohn, 2004, p. 27). As "easy to use" is not defined it is difficult to test for it. Likewise, using terms such as "always" and "never" ("the system will be always available") make a user story untestable. Changing "always" to "99% of the time" makes a test plausible (Cohn, 2004, p. 27).

At this point, the software development team may decide that they were mistaken about their velocity, or they may realise their estimates are incorrect. They can re-estimate the stories and ask the stakeholders to re-prioritise the release to reflect the updated estimates. This may involve breaking stories into new ones. The software development team and the stakeholders will work out the new estimates and priorities.

2.1.1.2 Managing User Stories during the Sprint

Central to the management process is the story wall. Beck (2000) first suggests using a "Big Visible Chart" to monitor sprint management (p. 72).

The story wall is an example of Cockburn's (2004) "information radiators" which continually display information in a way that is easy to understand but also accurate, i.e. the information is continually updated (p. 114-8). Some teams use the story wall to break the sprint down into phases for each of the tasks, to move cards through the stages of development (Delgadillo & Gotel, 2007). Others add burn down and story point charts to the story wall. Jeffries (2000) recommends charts for "Resources, Scope, Quality, and Time" (p. 166).

One Scrum technique used widely is the daily scrum or stand-up meeting (Schwaber, 2004, p. 8). As shown in Figure 2, this daily meeting functions as an iterative event

(Adapted from Schwaber, 2004, p. 6)

Figure 2 Daily Scrum in Relation to Sprint

within an iterative event (the sprint). This short meeting enables the project manager (or scrum master, a somewhat different role) to ascertain the current status of the sprint. At the scrum, each team member must answer three questions:

- What have you done since the last scrum?
- What are you going to do today?
- Do you have any problems? (Schwaber, 2004, p. 28)

If there are any problems reported, the project manager (or scrum master) will try to resolve them.

As user stories are a "reminder" to talk about a requirement, the stand-up meeting enables developers and stakeholders to plan on further conversations.

At the end of the sprint, there is a feedback session with the stakeholders. While it is possible the software development team has already received feedback during the sprint, particularly if they have a co-located customer, the post-sprint session is to show what has been done (Beck, 2000, p. 93-4). The stakeholders run the acceptance tests and confirm the software development is as they expect or not. Based on this feedback, more changes will be made to the remaining sprints in the sprint planning sessions.

As described earlier, once the stakeholders and software development team learn from the sprint, they must steer the release. They may adjust the velocity and re-estimate the stories, add new stories and re-prioritise the remaining stories in the release.

Using the documents of Agile Manifesto signatories, advocates and practitioners, this account defines the scope and vocabulary of the user story process. Having described the user story processes, the next subsection will examine the stated benefits from the same types of sources, not only the benefits, challenges and limitations of using user stories.

2.1.2 Stated benefits

While the concept of the user story is an interesting one, unless there is a benefit in using it, there is little incentive to change to this process. Although there are many stated benefits of the agile software development, specific benefits focusing on user stories are relatively few. However, it is useful to define what these benefits are in order to compare them with those discovered in the AT user story process.

2.1.2.1 Benefits of the User Stories

- User stories depend on verbal communication, and it has a much broader communication bandwidth. This facilitates understanding and communication (Cohn, 2004, p. 145-7; Monochristou and Vlachopoulou, 2007);
- User stories are easy to understand because they are written by stakeholders in natural language (Cohn, 2004, p. 45);
- User stories are relatively small which makes them easier to estimate (Cohn, 2004, p. 145; Monochristou and Vlachopoulou, 2007);
- User stories encourage collaboration through the entire user story process (Cohn, 2004, p. 145);
- The size of the story card constrains the amount of detail in a story (Keith, 2010, p. 92);
- Cards can be physically manipulated (sorted, edited, replaced, and passed) by many hands in collaborative settings (daily scrums and planning meetings) (Keith, 2010, p. 92);

- By deferring detail they better cope with change in understanding of problem domain (Beck, 2000; Cohn, 2004, p. 145);
- By using the standard format, the user story focuses on the user, their role and the business value (Monochristou and Vlachopoulou, 2007).

In addition, there are benefits identified for the story wall that are different from the user stories.

2.1.2.2 Benefits of Story Wall

The benefits of the story wall are that

- It shows the sprint status;
- If used for the sprint (or stand-up) meeting, the story wall is updated, making it relevant (Cockburn, 2004, p. 733);
- It answers questions stakeholders may have without requiring them to interrupt the software development team (Cockburn, 2004, p. 73; Sharp & Robinson, 2006a);
- It can be large, easily read and visible. It can be understandable at a glance (Cockburn, 2007, p. 43).

Thus, the benefits of story cards and the story wall have been specified by a variety of sources. These will be useful to compare to those articulated by members of AT. However, there may be situations when the user story process is not the right method for a software development project. The following subsection discusses challenges and limitations reported.

2.1.3 Challenges and Limitations

Some practitioners report challenges and limitations in the use of user stories. Challenges are situations that can be overcome by imagination, caution and vigilance. Limitations are not easily overcome; they are limitations to the user story process.

A review of the practitioner literature uncovers four sources of challenges and limitations:

- People;

- Organisations;
- Customers;
- Projects.

This section will examine each of these categories, discussing the identified challenges and limitations, defining workarounds, and identifying limitations.

2.1.3.1 People Challenges

People can be significant hurdles to the user story process. First, the process demands a great deal of involvement from all participants. Some people are naturally shy; others may not want to express their opinion in front of their boss (Cohn, 2004, p. 52). For these challenges, the facilitators must use their facilitation skills to engage the stakeholders.

Sometimes it may be difficult to get a stakeholder workshop started. Wake (2000) has his rip-up-the-first-card technique to get the stakeholders in the correct frame of mind. To get things started, Cohn (2004) believes it is good to have on hand "competitive or similar products" to show and discuss (p 52).

Likewise, there may be arguments between stakeholders. Naturally, the use of Parking Lots and Smells charts are ways overcoming these challenges.

As Grenning (2002) points out, there is a "people challenge" when a software development team estimates user stories. For this reason, he (2002) developed his story poker technique to overcome a lack of interest from the entire group and the tendency for a small subgroup to have long disputes. His game engages the entire team.

2.1.3.2 Organisational Challenges

This type of challenge has been explored by many researchers. Beck (2000) reports the biggest barrier to the success of an XP implementation (and the user story process) is an organisation's culture. Koch (2005) and Robinson and Sharp (2005a) concur with Beck's assertion. Both observe that user stories and indeed, the whole of agile development, can be implemented by any kind of organisation, depending upon the organisation commitment (Koch, 2005; Robinson & Sharp, 2005a). However, they outline some organisational challenges.

Alan S. Koch (2005) reports two organisational characteristics that can impact on whether user stories will be implemented successfully. Koch (2005) identifies a hierarchical organisational culture as a possible challenge to the successful implementation of the user story process. The user story process requires free and open communication between the software development team and stakeholders. This necessity contrasts sharply with the "clearly defined and restrictive communication paths" that characterise many traditional organisations (Koch, 2005, p. 11). On the other hand, Koch (2005) notes that a flat organisation might find it difficult to impose the kind of discipline on both stakeholders and the software development team roles to make the user story process successful.

Koch also notes that the attitude of an organisation towards change may be a challenge. Organisations have "well-defined procedures and tools in place to capture, track and manage change and deviations that come along" (Koch, 2005, p. 12). These organisations may have problems shifting to the "change embracing" culture of agile teams.

Cohn (2004) reports that if the prevailing culture of an organisation is one of blame, sometimes stakeholders refuse to participate (p. 162).

Robinson and Sharp (2005a) examine the impact of organisation culture of three different companies on XP implementations. Issues they find influential to the user story process:

- The behaviour, beliefs, attitudes and values. Procedures, processes, and plans of the outer organisation may override XP processes.
- The detail of the organisational structure can mean that the XP customer (and other stakeholders) is merely someone assigned the job, not necessarily the best person for the role. This may limit the rich flow of information into the user story with a committed and appropriate XP customer (Robinson and Sharp, 2005a).

While the commitment of the organisation to make the user story process succeed is crucial, there may need to be changes to the organisational culture.

2.1.3.3 Customer Challenges

One challenge an external customer may impose on an agile software development team is the requirement to "augment [the user stories] with additional documentation if requirements traceability is mandated" (Cohn, 2004, p. 153). An extreme example of this challenge is if the external customer is the United States Department of Defense. They specify an extremely detailed standard of documentation for their software development projects (Joint OSD/Services/Industry Working Group, 1993). This may be a limitation.

Beck (2000) points out that the XP customer must know how to write stories, and learn how to frame the business value as acceptance criteria (p. 143). In addition, there are qualities that may be harder to define, such as an "attitude" to make one successful (Beck, 2000, p. 143). Presumably, as the Customer comes from the stakeholders, neither of these talents is developed there. Therefore, there is quite a high threshold for the on-going Customer.

Once a good Customer is found, the problem may be keeping them. Koskela and Abrahamsson (2004) find that the role of Customer particularly when on-site is "demanding" (p. 1). Martin, Biddle, & Noble (2004) note that the Janus-faced responsibilities of the co-located Customer may put him or her "under significantly more pressure than developers or other participants in the project" (p. 51). This factor introduces a question of how long any one can remain in that role.

2.1.3.4 Project Challenges

Many have pointed out that size may be a challenge for the implementation of the user story process. The sheer number of user stories in a project may be one problem because it may be "difficult to understand the relationship between stories" (Cohn, 2004, p. 153).

Secondly, Koch (2005) notes that the preference for the immediacy of face-to-face communication "places a limit on size" of teams (p. 21). On larger teams, important information "must be written down" or it "does not get dispersed among as many on the team" (Cohn, 2004, p. 153). It becomes harder to have a large number of people actively involved in story writing workshops (Koch, 2005, p. 21).

Distributed software projects have even more challenges. There is a loss of the communication bandwidth, though the improvement of teleconferencing facilities may in the future somewhat alleviate this problem (Koch, 2005, p. 21).

Security or safety considerations may add non-negotiable features to a software development project (Koch, 2005, p. 25). This may make the user story process inappropriate. As Beck (2000) points out, XP may not be appropriate to build missile nosecone software (p. 155). This may be a limitation.

In summary, this section of the literature review has created a theoretical user story process which gives this study a vocabulary and context with which to discuss the AT user story process. It further defines the commonly claimed benefits, limitations and challenges. These benefits, limitations and challenges can be compared with those uncovered during the AT research.

Having gained a way to describe the user story process and the context in which it is used, focus turns to research literature to understand what is empirically known about user stories.

2.2 What is Empirically Known about User Stories?

As noted in the Introduction, user stories are not well-represented in research literature. This section of the literature review uses as a starting point Dybå and Dingsøyr's (2008) landmark survey of empirical agile software research and concludes with a review of more recent literature.

2.2.1 Dybå and Dingsøyr's Landmark Survey

In their systematic literature review of empirical agile software research, Dybå and Dingsøyr (2008) examine 1995 articles on agile software development through the end of 2005. They find only 36 of sufficient rigor, credibility and relevance. Of these 36, none focus specifically on user stories.

Why are there so few empirical studies on agile software development? Without a doubt, there are difficulties involved in empirical agile research. The difficulties SE research has accurately capturing metrics that Kaner and Bond (2004) identify are

compounded in a software development environment that values “interactions,” “collaboration,” “lightness” and “responsiveness to change” (Beck et al., 2001)

Further, one can see in the Agile Manifesto that the processes and tools, comprehensive documentation, contracts and plans that attract these types of metrics that are useful in traditional empirical research are a lower priority in agile software development (Beck et al., 2001).

The problem is worse when traditional metrics such as “lines of code” are applied to an agile software development team. One of the common agile practices – refactoring – reduces the number of lines by simplifying the code. Thus, in an agile project, progress reduces the number of lines instead of increasing it.

Beck (2000) suggests a team’s velocity is the best metric for an agile software development project. Unfortunately, velocity is an idiosyncratic measure. It is dependent upon factors such as sprint length, complexity of work, and number of team members. While this measurement is meaningful for software development teams and their managers, it is not useful for empirical research.

New ways of measuring

Two of the empirical studies Dybå and Dingsøyr (2008) judge sufficiently rigorous propose new ways of researching agile software development. Both study user stories as part of a comprehensive look at XP development. As such, both must be considered in this literature review as possible ways of studying the use of user stories.

The first new method is the controlled case study reported by Abrahamsson and Koskela (2004). In the same year, Abrahamsson collaborated with Salo (2004) on an article presenting this method. Acknowledging the difficulties of researching a software development method characterised by “rapid iterative cycles and continuous changes in process and product requirements,” Salo and Abrahamsson (2004) offer the controlled case study as a method to provide “systematic empirical research on agile principals and methods” (p. 412). In the years since, Abrahamsson has involved in several controlled case studies (Keränen & Abrahamsson, 2005; Moser, Abrahamsson, Pedrycz, Sillitti, & Succi, 2008).

Salo and Abrahamsson (2004) cite as one of the advantages of carrying out a controlled case study is being able to create “close-to-industry” conditions with “business pressure” (Salo & Abrahamsson, 2004, p. 413). What “close-to-industry” conditions mean in Abrahamsson & Koskela’s (2004) study is a project of four student developers building “a virtual file cabinet”. Abrahamsson & Koskela (2004) measure time, size, and defects of this project. Their most interesting finding is perhaps controversial – they find that there is “little need for actual customer involvement” (p. 79). Although the students use user stories to create software, this article does not explicitly examine the user story process.

Problems with the controlled case study: Despite claims of being able to reproduce a “close-to-industry” environment with “real business pressure,” this study still does not report on an industrial environment (Abrahamsson & Koskela, 2004).

Secondly, there is a high degree of control over the implementation environment. For example, Abrahamsson & Koskela (2004) expect the developers to keep development diaries. As students they probably feel the need to do so. However, it is unrealistic to believe that one could get developers to keep development diaries in an industrial environment.

When using students, one must always deal with a learning curve. Two weeks before the start of the project, Abrahamsson & Koskela (2004) assigned the students two agile books to become familiar with the method. Despite this, they find that the students quickly adapt to giving more realistic estimates in two weeks (2004, p. 80). However, estimating is only one part of the user story process, a consequence of studying time, size and defects.

The research of Abrahamsson & Koskela (2004) does not suggest a method to use to study user stories in practice. There is an additional question of how useful measuring time, size and defects are in agile software development research. Lastly, this research does not offer much empirical information about user stories.

The second of the studies that propose a new method to study agile is that Layman et al. (2004). They call their method the Extreme Programming Evaluation Framework (XP-

EF). Their case study uses it to examine the agile implementation of a team at Sabre Airline Solutions (Layman et al., 2004).

Layman et al. (2004) compare metrics from this team's third and ninth releases. In the two year time span between these releases, the team implemented XP and gained proficiency in using it.

Layman et al. (2004) compile three categories of metrics:

- XP Context Measures: Quantifying how unique the context in which the project takes place;
- XP Adherence Metrics: How "XP" is the project's XP standards;
- XP Outcome Metrics: The kind of data gathered by traditional software development context, such as test defects/KLOEC of code for pre-release quality, defects/KLOEC of code 6 months after release, the Putnam Productivity Parameter for programmer productivity, customer interviews and a survey for team morale (Layman et al., 2004).

By comparing the outcome metrics of the third and ninth, Layman et al. (2004) find fewer mistakes both pre- and post-release, greater programmer productivity, higher customer satisfaction and better team morale.

Problems with the XP-EF: As noted earlier, it is questionable whether the kinds of metrics collected in the Outcome Measures are adequate for studying agile (in this case, XP) projects. Both the Context Factors and Adherence Metrics require extensive data collection that many industrial environments may not maintain. Further, while the discipline of tightly defining the context is useful for both the research and practitioner communities, since there is no XP "Body of Knowledge" it is questionable what value there may be in measuring what the individual team members understand XP as.

XP-EF is a large framework that would probably be best utilised in comparing an XP implementation, not the user story process.

Human and Social Factors

To gain a deeper understanding of user stories requires knowing more than metrics such as time, size and defects. In the same article, Dybå and Dingsøyr (2008) identify a stream of agile research that studies human and social factors (p. 844). Several of these are ethnographic (or "ethnographically-informed") studies of agile software development. In these studies, researchers "attend to the taken-for-granted, accepted and un-remarked aspects of practice, deliberately considering all activities as strange" (Hugh Robinson et al., 2007, p. 541).

In the first of these articles, MacKenzie and Monk (2004) study the Deskartes Universal project at company KMS, visiting the project two or three times a week for three months. They watch pair programming sessions and team meetings. They read source code, design documents and code repository logs (MacKenzie & Monk, 2004).

MacKenzie and Monk (2004) regard the presence of Dilbert cartoons, Coke bottles, and "thick semi-popular computer press 'how-to' books" as if regarding spores of the species computis programmus (p. 95). They view equally strange how the team handles the story cards. MacKenzie and Monk (2004) observe the team filling them out, shuffling them, dealing them out, handing them in, 'spawning' new ones, displaying, swapping, storing them, writing on them and putting them away (p. 98). MacKenzie and Monk (2004) liken these activities to a card game. They observe as central to these activities focus and commitment. This focus and commitment is evident in the questions, negotiations and discussions (MacKenzie & Monk, 2004) that transform and re-transform the meaning of the user stories.

MacKenzie and Monk (2004) demonstrate the desirability of the ethnographic practices of looking at well-recognised activities as strange. The value of this practice may lead to the discovery of something new in a direction previously thought well understood.

In the same way, Sharp and Robinson (2004) also hope to uncover implicit and previously overlooked features of practice. In addition, they look for the difference between what participants say they do and what they actually do.

Helen Sharp and Hugh Robinson have collaborated on four of human factors agile studies that Dybå and Dingsøyr (2008) identify as empirically significant (Hugh Robinson & Sharp, 2004; H. Robinson & Sharp, 2005a; Hugh Robinson & Sharp,

2005b; Sharp & Robinson, 2004). They aspire to an approach that is ethnographically informed: "participant-observation that gives no *a priori* significance to any particular feature of practice" (Sharp & Robinson, 2004, p. 355).

Over the decade, Sharp and Robinson have built a considerable collection of data which they reuse in each of a series of XP studies. They focus on aspects of culture and community in successful XP practice, characteristics of XP teams, the kinds of social interactions that arise in practice, and how organisational culture influences XP practice.

In the first of the articles, Sharp and Robinson (2004) "explore the values, beliefs and assumptions that inform and shape agile practice" (p. 355). The research on which this article is based is an ethnographic study involving participant-observation. Data collected includes "field notes, audio recordings of discussions and meetings, photographs of physical layout and copies of artefacts" (Sharp & Robinson 2004 p. 357). The researchers pursued themes in the data analysis.

Sharp and Robinson (2004) find these characteristics or themes of XP culture:

- Respect for both individuals and team;
- Both individuals and team are responsible;
- Preservation of the quality of working life;
- The XP process values the work of both individuals and the team gives them "faith in their own abilities to achieve the goals they have set for themselves" (p. 373).

Each of these characteristics is not accidental, they are essential for running a successful XP team.

In the same year, Robinson and Sharp (2004) expand on their previous study by supplementing it with another ethnographic study of a second mature XP team working in a different environment. Examining the data for the four themes they found in the earlier article, Robinson and Sharp (2004) find that while all the previous four characteristics are evident, they manifest themselves differently. For example, the culture of one team has stress-relieving techniques; the other team may not even take breaks. However, despite "being different in nearly all other aspects apart from their use of XP," both teams are characterised by:

- Respect;
- Responsibility;
- Preservation of the quality of working life;
- Faith in their own abilities (p. 147).

In addition, Robinson and Sharp (2004) unearth a new characteristic from the data – trust. They state that trust is complementary with the other five characteristics, "and needed in order for them to flourish throughout the team's activities" (p. 146). They specifically point out that both the user story process and pair programming need trust in order for everyone to respect what each other brings to the process.

Robinson and Sharp (2004) summarise that there is no cause and effect relationship between teams that practice XP and teams who show these characteristics. Rather, they identify a "reflexive relationship" between these characteristics and the twelve XP practices that is mediated by the specific circumstances.

In *Organisational culture and XP: three case studies*, Robinson and Sharp (2005a) add the observations and other data from another XP team to the data they have previously collected. For this article they look at the larger organisation culture and examine its impact on the XP team.

The organisation to which the new XP team belongs is a large multinational bank. Thus its organisational culture differs sharply from the two previous teams whose products are software. As noted before, Robinson and Sharp (2005a) conclude that XP can thrive in very disparate environments and organisational cultures. However, it is obvious that both the organisational culture and structure can impact the agile software development culture.

In the last of the articles of interest to this study that are included in Dybå and Dingsøyr's 2008 systematic literature review, Robinson and Sharp (2005b) re-examine their ethnographic data of the three teams in the previous article. Their objective is to explore the "social interactions involving individuals or groups of individuals which arise from practices and which have consequences for the practices" (Robinson & Sharp, 2005b, p. 100).

The practices they examined were:

- Pair programming;
- Customer Collaboration;
- How each team kept track of progress (Robinson and Sharp, 2005b).

Naturally, pair programming is not of interest to this study. However, user stories are part of the other two practices.

Robinson and Sharp (2005b) conclude there are "no simple rote rules" (p. 107) to an XP implementation. Each of the social practices of XP depends on many factors. One cannot simply move programmers next to each other and expect pair programming to thrive. An XP implementation involves more than simple techniques; it may involve issues not normally considered in conventional software development approaches: the social side.

While Sharp and Robinson have not focused on user stories in particular, their research has much to offer this study in presenting one approach to researching the human or social aspects of agile development. Along with MacKenzie and Monk, Sharp and Robinson articles demonstrate an approach to analyse the "human factors" of agile software development. Robinson and Sharp (2004) state: "XP is as much about human values as about technical values" (p. 139). These works are of interest to a study such as this which examines human factors, information transformations, triggers for action, and information breakdowns.

2.2.2 Current Research

Research literature since 2006 often examines user stories in terms of distributed software development (Chubov & Droujkov, 2007; Hirschfeld, Steinert, & Lincke, 2011; Sohan, Richter, & Maurer, 2010). As the research environment of this study is collocated, this research is not relevant.

Still others have focused on one component of user story process such as estimation (Aiello et al., 2007; Bugayenko, 2009; Gomez et al., 2010; Haugen, 2006; Miranda, Bourque, & Abran, 2009). While these articles may be interesting, they are not about the information flows and transformations of the user story process.

Likewise, research that questions the appropriateness of user stories for agile software development (Gallardo-Valencia, Olivera, & Sim, 2007; Morreale et al., 2006) are not in scope.

Six recent articles focusing on user stories have been found of interest to this thesis. In the first of these, Yagüe, Rodriguez & Garbajosa (2009) study the "hidden requirements" uncovered during the user story process (p. 180). They find that discovering errors in testing shows the user story process is working: if a test fails, it does not necessarily mean there is a failure in design or coding. It "may show the evidence that some assumed requirements had not been considered" because they were invisible (Yagüe et al., 2009, p. 181).

This article is a case study of a product called TOPENprimer for a biogas power plant (Yagüe et al., 2009, p. 183-4). This project was developed by a team of eight in six months managed with scrum techniques as well as user stories, pair programming and Story Test-Driven Development (STDD). Their aim is to develop an improved test process to identify these hidden requirements earlier.

Unfortunately, Yagüe et al. (2009) do not go into great detail about their analysis process. Their conclusions veer from the obvious - the earlier the team finds hidden requirements, the less the rework and refactoring is required - to a view that consideration should be given to how much time is spent in meetings using their improved process (Yagüe et al., 2009, p. 183-4).

In the next study, Gallardo-Valencia and Sim (2009) conduct a field study on requirements validation in an agile software development team. Gallardo-Valencia and Sim share the same concern and Yagüe et al. (2009) – whether the requirements are valid or not.

What Gallardo-Valencia and Sim (2009) focus on that is of interest to this thesis is their work on the information flows through the software development team. Their field study includes two days of observation, six semi-structured interviews with people from representative roles, and artefacts.

The research context is a company that provides online financial management application for senior citizens to manage their investments. The team consists of ten people, including a scrum master, customer ("product owner"), technical manager, developers and testers.

Gallardo-Valencia and Sim (2009) note the primary unit of work is the user story, which consists of written reminders, test cases and conversations. The validation process consists of the involvement of each member of the team to the on-going definition of the user stories through these components. As is oft mentioned, the agile environment demands everyone to be responsible for the code (Beck, 2000, p. 59). Gallardo-Valencia and Sim (2009) find the same collective ownership over the requirements. They liken the use story process as an unwritten "live requirements knowledge" constantly being honed through interactions between members of the team: "written requirements knowledge is complemented by requirements knowledge that is shared through conversations among stakeholders and software team members" (Gallardo-Valencia & Sim, 2009, p. 67).

Sharp and Robinson continue their ethnographic research into agile software development teams with another four articles. Sharp and Robinson (2006a) use a Distributed Cognition framework as a "unifying approach" for studying the "socially complex work situation" of agile software development (p. 1). They add fresh ethnographic data based upon a newly observed agile software development team to data they had already collected (Sharp & Robinson, 2006a). They consider the XP team, its interactions and environment with a Distributed Cognition approach. They also offer findings with which to compare this study's findings.

Using their data, Sharp and Robinson (2006a) find information flows, transformations and applications in the four teams:

- Distributed Problem-Solving;
- Verbal and Non-verbal Behaviour;
- Coordination Mechanisms;
- Communication pathways;

- Knowledge Sharing and Access (p. 4-7).

Sharp and Robinson (2006a) find that problem-solving is highly distributed both across people and time. Teams communicate through verbal and non-verbal behaviours such as talking, gesturing, notes, drawings and doodles. Although there are other ad-hoc mechanisms, the story cards and story wall are the primary coordinating mechanisms. Sharp and Robinson (2006a) find the story wall is a way of sharing knowledge. In addition, Sharp and Robinson (2006a) find that the Distributed Cognition approach is particularly useful for finding "break downs" in information flow (p. 8).

They find the "main transformation taking place in this cognitive system is that of transforming the story into executable code" (Sharp & Robinson, 2006a, p. 9). Outside the user story process, Sharp and Robinson (2006a) find very little information propagation. The user story remains the "central focus of development from the time it is created until the code is handed over" (Sharp & Robinson, 2006a, p. 9). The reason teams can create executable code from these "simple flows" is that the user stories are "small, manageable chunks" (Sharp & Robinson, 2006a, p. 9).

The importance of this research to this thesis is that they are using a framework to look at the software development team, their environment and artefacts as a single cognitive unit.

In the same year, Sharp et al. (2006b) narrow their focus to story cards and the story wall, "simple yet powerful" artefacts that "provide key tangible evidence of information sharing" (p. 65). While the two researchers observed for eight working days, unlike their earlier ethnographic studies, they did not participate (Sharp et al., 2006b). Data consisted of "extensive field notes, photographs, and copies of work artefacts" (Sharp et al., 2006b, p. 66).

Sharp et al. (2006b) examine a mature XP team and describe and analyse their use of story and task cards, the relationship of the cards to the story wall, and the relationship of the cards to the rest of the XP team. While the approach once more is Distributed Cognition, this time they use the Distributed Cognition for Teamwork (DiCoT) framework. DiCoT is a "methodology and representational system to support the Distributed Cognition analysis of a small team" (Sharp et al., 2006b, p. 66).

DiCoT consists of three themes – physical, artefact and information flow. Each of these has an array of principles (APPENDIX A). Sharp et al. (2006b) find many examples of the principles. They advance that the story cards and the story wall are mediating artefacts. Physical layout principles such as "Horizon of Observation" and "Space and cognition" focus on the immediate obviousness of the project status from "The Wall" (Sharp et al., 2006b, p. 67).

However, their focus is the flow of information through, around and within the XP team. They find the main information transformation of the XP team is to transform the story card into working code (Sharp et al., 2006b). To that end, the XP team has a series of meetings that function as information hubs in which the understanding of many people arrive to effect the transformation. The initial iteration planning meeting acts not only as an information hub but also an information buffer, since information may arise that may apply in the next iteration.

Sharp and Robinson (2006b) find that there are few mediating artefacts, which are story cards and the story wall. These are primarily involved in the process – the user story process – which means they contain little information. Secondly, they find information flows promote situational awareness, and the XP team environment is an information-rich, open and accessible environment (Sharp et al., 2006b).

Sharp et al. (2006b) then interrogate the DiCoT themes to answer "What If?" questions (p. 74). They acknowledge the article is only an example of a DiCoT analysis. Sharp et al. (2006b) do not deploy the complete DiCoT analysis, involving models and summaries for each of the themes. They use this article to share their findings.

Sharp and Robinson (2008) follow up this preliminary work using the DiCoT method with another research article on three XP teams they have analysed before. They (2008) focus on the artefact theme of DiCoT, especially exploring "the significance of the paper-based nature of these artefacts" (p 506). They posit that these artefacts are "mechanisms of co-ordination and collaboration" (Sharp & Robinson, 2008, p. 507).

Sharp and Robinson explicitly detail the uses for each team of the story cards, the wall and how they are used to coordinate and collaborate. Then they combine inter-team

lists of co-ordination activities, the effects of the medium (story cards) and issues for distributed agile teams.

Sharp et al. (2009) once again revisit the data of the six XP teams they have collected. Motivated to extend their earlier work on story cards and the story wall, they investigate their physical nature through a notational perspective in addition to the usual social perspective. They endeavour to define a "generic view of these artefacts and their use" (Sharp et al., 2009, p. 109). For one conversant with their earlier work, this view is familiar. Sharp et al. (2009) walk through Green's fourteen cognitive dimensions in relation to the physical artefacts. They then apply these same dimensions to the social perspective.

For example, they discuss how there is another level of abstraction in the use of the story card: "This higher level is not and cannot be contained in the card notation or the story wall, but must rely on a wider understanding held within the team" (Sharp et al., 2009, p. 114). Essentially, by locating the physical artefacts within the larger cognitive system (from the Distributed Cognition perspective) they can usefully apply these dimensions.

The final study by these researchers examined in this literature review is a chapter in the book *Collaborative Software Engineering*. They propose that the success of agile software development depends not merely on the technical agile process but also the "intimate social activity which emphasise close collaboration, co-ordination and communication within the development system" (Hugh Robinson & Sharp, 2010, p. 94).

Using the ethnographically-informed approach and the data of six XP teams, Robinson and Sharp (2010) show that agile development practice involves *collaborative* and *communicative* social activity evident through pair programming and customer collaboration. Robinson and Sharp (2010) find that evidence shows that collaborative and communicative social activity is evident across all six teams in pairing and customer collaboration. The kind of technical practice this involves is test-first coding, refactoring, simple design and continuous integration in addition to pair programming. While pairing is outside of the scope of this research project, it is worth noting that Robinson and Sharp did find consistent patterns among the six teams.

Unfortunately, for the aspect of collaborative and communicative activity that is in scope for this project – customer collaboration – Robinson and Sharp find that these activities are so rich and situated that it is "difficult to identify recurring collaborative activities and communication patterns" (Hugh Robinson & Sharp, 2010, p. 97). Lastly, Robinson and Sharp (2010) affirm once again that the story cards and the wall co-ordinate the collaboration and communicative activities of an XP team.

A literature review of current agile research literature has uncovered several possibilities for research method. Of the research literature explored, the work of Sharp and Robinson, and in particular 2006b confirms the original interest in Distributed Cognition. Through the work of Sharp and Robinson, one can see the use of user stories as a worthwhile subject for a Distributed Cognition analysis. Not only is cognition shared amongst the members of the team, it emerges over time. The story card is an example of artefact.

The first two sections of the literature review define user stories and the story wall, the user story process and explore empirical research literature on user stories. This exploration shows that user stories are not only an interesting phenomenon but also not well known empirically.

After studying the literature, it is apparent the kind of research methodology would be one of the "human and social factors" described by Dybå and Dingsøyr (2008). The ethnographic method of MacKenzie and Monk (2004) provokes interest for studying the use of user stories in practice, as do the ethnographically-informed case studies of Sharp and Robinson (2006b, 2009).

It is the Distributed Cognition lens of Sharp and Robinson (2006b, 2009) that gives rise to the final section of the literature review in which Distributed Cognition theory is examined, both in itself and as a research approach for the study of user stories.

2.3 Distributed Cognition as a Theoretical Lens to Study User Stories

The theory of Distributed Cognition finds cognitive activities shared amongst people, embedded in artefacts and the environment, and emerging over time (Hollan et al., 2000). Whenever two people "put their heads together to figure something out," they

are taking advantage of the synergy of Distributed Cognition. Whenever one uses a calculator, a slide rule, or GPS device, one is distributing cognition.

The founder of the field of Distributed Cognition is Edwin Hutchins of the University of California San Diego. As a cognitive anthropologist, Hutchins (1995a) has always tended to research in the field rather than a laboratory where he surmises that cognition "is not just influenced by culture and society, but that it is in a very fundamental sense a cultural and social process" (p. 3).

Hutchins (1995a) defines cognition as "computation realised through the creation, transformation, and propagation of representational states" (p. 49). With this intimate interweaving of mind and body, it is not inappropriate people often refer to "seeing" a solution to a problem. Hutchins points to Nobel laureate Herbert A. Simon and his view of the cognitive activity of problem-solving. Simon uses mathematics as a starting point by pointing out that theorem solving is a matter of making what is inherent but hidden obvious (Simon, 1996, p. 132). Simon (1996) then extends this idea to the whole of problem-solving: "Solving a problem simply means representing it so as to make the solution transparent" (p. 132). Problem-solving in the world is not evidence of cognition; it is cognition.

The research Hutchins collected on patrol with a US Navy warship resulted in his 1995 book, *Cognition in the Wild*. His focus is how the team collaborates to "fix" the ship's position, especially in those situations in which the ship threads through narrow channels. Hutchins (1995a) opens the book with an emergency situation to illustrate how the crew combines with specific tools, gauges, and meters, and maps to determine the ship's position. Hutchins (1995a) points out the cognition in action:

> *Many kinds of thinking were required to perform this task. Some of them were happening in parallel, some in coordination with others, some inside the heads of individuals, and some quite clearly both inside and outside the heads of the participants (p. 5-6).*

This kind of event prompts Hutchins to see a complex cognitive system involving a number of actors working in concert with the physical environment and tools to solve a problem. He continues his study by examining how Polynesian seafarers exploit star

positions and the horizon to know where they are. They project structure into the physical environment, onto the night sky. This structure in turn aids their navigation.

Navigation is merely an example of how human beings solve a difficult and complicated problem none of them would be able to solve easily (or at all) by themselves. It is about how people leverage their cognition with tools and the environment. Hutchins opposes the then prevalent idea that there is a "firm drawing of the inside/outside boundary" (Hutchins, 1995a, p. 355).

Hutchins (1995b) continues this work in a different environment – the cockpit of a commercial airliner. His research article "How a Cockpit Remembers its Speeds," focuses on how cockpit crews share cognition while landing. Critical to safe landing is the knowing what the speed is. Throughout descent the pilot must balance the speed with flap position to keep the plane from stalling.

Hutchins (1995b) advances the view that a cockpit as a single complex cognitive system. The theory of Distributed Cognition looks "in the right places for answers to questions" (p. 287). Through a variety of representations - from the permanent speed card, through devices such as "speed bugs" to the ephemeral reading aloud of speed by the non-flying pilot, this cognitive system combines auditory and visual channels to give the pilot a redundancy of representations to enable him or her to set flap position correctly.

Much of Hutchins work is in reaction to the then prevailing Computational Theory of the Mind (CTM). This theory holds that people copy images of the external world into their minds from the external world and then carry out computations on them. Bodies are just sensors; all cognition takes place in the mind.

In contrast, Hutchins advances that if people want to know what the world looks like, they just look at it. People are "loosely coupled" with the world and do not need to make mental representations to mediate between the mind and the world.

In a later article, Hollan et al. (2000) explain:

> *Whereas traditional views look for cognitive events in the manipulation of symbols inside individual actors, distributed cognition looks for a broader*

> *class of cognitive events and does not expect all such events to be encompassed by the skin or skull of an individual (p. 175-6).*

In a similar way, Donald A. Norman rejects the CTM. A one-time colleague of Hutchins at the University of California San Diego, Norman works primarily in the "user-centred design" field, focusing on the development of "cognitive artefacts." These are tools that aid the mind.

Norman looks at cognition in a different way than Hutchins. While Hutchins focuses on how groups solve problems "in the wild", as befits an anthropologist, Norman researches Distributed Cognition in a laboratory. With Zhang, he examines how people solve the Tower of Hanoi (ToH) experiments (Zhang & Norman, 1994). What they find is that if the rules of the ToH game are "represented externally," or embedded in the game as rules through the structure of the artefacts, these external representations can:

- Function as memory aids;
- Impart information can be perceived and used without being interpreted and formulated explicitly;
- Anchor and structure cognitive behaviour;
- Change the nature of a task (Zhang and Norman, 1994, p. 118-9).

In essence, Zhang and Norman (1994) find that these external representations are "an indispensable part of the representational system of any distributed cognitive task" (p. 119).

Philosopher Andy Clark (2008) also sees the human mind as much more capable than simple machines. He contrasts two themes of the mind as BRAINBOUND and EXTENDED (Clark, 2009, p. xxvii). Clark (2008) sees as BRAINBOUND the Cartesian idea that all intelligence resides in the brain with the body as a mere sensor (Clark, 2009, p. xxvii). Equally BRAINBOUND is the CTM. Clark (2008) illustrates the inadequacy of modelling the mind on a central processing unit by looking at a true example of the CTM - Honda's Asimo robot. Clark (2008) points out how Asimo walks:

> *…by means of very precise, and energy-intensive, joint-angle control systems, biological walking agents make maximal use of mass properties*

> *and biomechanical couplings present in the overall musculoskeletal system and walking apparatus itself (p. 3-4).*

Needless to say, it has none of the bodily fluency that humans exhibit in movement. The true model of human cognition is the EXTENDED mind model with an "inextricable tangle" of what Clark calls "feedback, feedforward and feedaround loops" (Clark, 2008, p. xxvii).

These activities are not restricted to a central processing unit as in the Computational Theory of Mind. As do Norman and Hutchins, Clark sees as flawed the idea that humans copy representations and then manipulate them. He believes the human mind is not like a computer, that humans can manipulate and use information outside of their minds (Clark, 2008, p. 141).

Thus a computer is a poor model for the human brain. If humans had to make the kind of cognitive effort Asimo does to walk, there would be little else for them to think about other than joint angle.

The question arises, what is the significance for an agile software development team of this internal versus external representation debate? If the external representation view is taken, then the stakeholders and software development team are intimately embedded in the external world. This means that the external world, such as user story cards, can help them think. Furthermore, this activity "leaks" out from the mind to the external world (e.g. user stories) (Clark, 2008, p. xxviii). Therefore, it is worth considering how well the artefacts such as a user story or story wall function as part of a complex cognitive system, and whether these external representations can be improved to better support this Distributed Cognition.

2.3.1 Theory of Distributed Cognition as a Lens

Distributed Cognition is used as a useful theoretical lens through which to examine and explain collaborative teamwork (Rogers & Ellis, 1994). Therefore it can apply to the teamwork undertaken to an agile software development team. One needs to go no further than the Agile Manifesto to see the agile values (interactions, collaboration, responding to change) align with the Distributed Cognition domain (Beck et al., 2001).

One early research article applying Distributed Cognition theory to SE is that of Flor and Hutchins (1991). They observe two programmers working together to complete a simple maintenance task. Using video and keystroke loggers, Flor and Hutchins break down the information flows in a single task. They were able to resolve information flows down to gesture and eye movement.

Flor and Hutchins (1991) find the system defined by these two programmers working in a shared space on a shared task had different cognitive abilities than individuals working alone. Flor and Hutchins (1991) suggest that this kind of analysis is useful for exploring collaborative work at the system level (p. 56).

Since that early work, many Computer and Information Science fields have usefully employed Distributed Cognition. It has been embraced by the Human-Computer Interaction (HCI) and Computer Supported Collaborative Work communities (Hollan et al., 2000; Rogers & Ellis, 1994; Wright, Fields, & Harrison, 2000).

In their 2006 article, Aranda and Easterbrook advance Distributed Cognition as an appropriate lens through which to study SE. Although there is so much research in certain aspects of SE, they see little of it on how software engineers really work. Aranda and Easterbrook (2006) state:

> *We do not have a theoretical framework that links the separate phenomena we observe, unifies our perspectives of the domain and allows us to generate testable predictions of software projects and software teams (p. 35).*

However, Aranda and Easterbrook (2006) believe that with the development of the theory of Distributed Cognition, SE researchers have an interdisciplinary tool to explore software development in practice. It gives researchers a systemic view of software teams. It is this view that is needed to study the use of user stories in practice.

Distributed Cognition also gives researchers a perspective “to study artefacts as “embodied knowledge – “they store rules and process that simplify cognitive tasks of their user” (Aranda and Easterbrook, 2006, p. 36).

Aranda and Easterbrook (2006) also point out the advantages of being able to define the information flows around a software development team. This advantage is of particular interest to research in agile software development, with its high intensity communication and collaboration, use of artefacts and physical environment.

Lastly, Aranda and Easterbrook (2006) claim a Distributed Cognition framework allows researchers to study SE on short term and long term bases. On a short term level, they see the focus is "resolutions of cognitive problems". In the long term, it allows the researcher to "investigate long-term learning and structuring activities" (Aranda & Easterbrook, 2006).

In summary, Aranda and Easterbrook (2006) suggest these topics to be the minimum data to be collected for a Distributed Cognition analysis:

- Patterns and structures of groups and their interactions;
- Artefacts (tool, documents) and patterns of artefact use;
- Nature and frequency of tasks;
- Shared understanding, breakdowns and recoveries (Aranda & Easterbrook, 2006).

Aranda and Easterbrook (2006) suggest widening the tools for data collection and data analysis for a Distributed Cognition analysis. They offer methods such as social network analysis and artefact analysis. For this study, such suggestions are of interest, as the data in this instance is interview data.

Another method of data analysis is the Distributed Cognition for Teamwork (DiCoT) method. Furniss developed this method in his Master's thesis on the London Ambulance Service (2004). It is also the method used by Sharp and Robinson (2006b).

APPENDIX A shows three of the themes used in DiCoT to categorise the main principles identified by Blandford and Furniss (2006). These are used to analyse the user story process in the study described in this thesis. The details and justification are detailed in the next subsection.

2.3.2 Distributed Cognition for Teamwork Method

Distributed Cognition for Teamwork (DiCoT) combines principles of Distributed Cognition with the ideas of from the user-centred Contextual Design (Sharp et al., 2006b). DiCoT comprises a systematic exploration of three functional levels of a cognitive system (Blandford & Furniss, 2006; Furniss, 2004; Furniss & Blandford, 2010).

The three functional themes are:

- Artefact
- Information Flow
- Physical

Furniss (2004) points out while these themes may be separate, together they model the propagation and transformation of information in a cognitive system.

In addition, Furniss (2004) specifies Social and Evolutionary themes though he does not use them in his thesis. Other researchers also mention these themes, though none explicitly develops them (Blandford & Furniss, 2006; Furniss & Blandford, 2006; Sharp et al., 2006b). Thus, the focus in this study is on the three themes that have been developed.

For each theme, Furniss (2004) proposes to derive three output artefacts:

- A set of principles that pertain to that theme;
- A set of diagrammed representations that illustrate the structure and flow from the relevant perspective;
- A set of tabular representations that present a summary of the system, details, further observations, and emergent issues, all from the viewpoint of the theme (Blandford and Furniss 2006).

2.3.2.1 Artefact Theme

The artefact theme centres on artefacts and the use and adaptations of physical objects to store, transform and communicate information. While artefacts may be physical

objects, there are information artefacts such as language, writing, counting and other formal systems (Furniss, 2004, p. 17).

There are five principles associated with the artefact theme.

Coordination of Resources

Furniss (2004) combines Hutchins' (1995a) use of coordination with Wright's concept of information resources (Wright et al., 2000) to define this category as "internally and externally coordinated to aid action and cognition" (p. 18). Hutchins talks about the importance of coordination between representations as they are propagated between one representational medium to another. If they are insufficiently coordinated they may "break down" (p. 117).

In his Resources Model, Wright develops on Hutchins' definition of coordination. Wright offers six information resources that are necessary to coordinate:

- Plans – a known sequence of action;
- Goals – the state to which action should progress;
- Affordances – Wright calls affordances as "the set of possible things that can be chosen" (p. 8). His definition is informed by Norman's (1999) restatement: "Affordances reflect the possible relationships among actors and objects: they are properties of the world" (p. 42);
- History – the steps from the plan already executed, decisions, interpretations already made;
- Action-effect – delineation of possible cause-and-effect relationships;
- Current state (Wright et al., 2000).

Thus, by involving an artefact or artefacts in coordinating these resources, a team can avoid having to "internally coordinate the activity, which will become more demanding with the increasing complexity of the activity" (Furniss, 2004, p. 18). Essentially, members of a team are able to see differences and problems with representations if they are both visible. Bruno Latour (1988) puts it in another way, "Positive feedback will get under way as soon as one is able to muster a large number of mobile, readable, visible resources at one spot to support a point" (p. 12).

Creating Scaffolding

According to Hollan et al. (2000), the organisation of the mind is an emergent property of the coupling of internal and external resources. People "enlist the world to perform computations for them" (p. 191). They use it to simplify cognitive tasks.

Hollan et al. (2000) point to a simple analogue example - striking a bundle of spaghetti on a table to find the longest one. Another example is how people playing Tetris manipulate the falling blocks to save computational effort. In general, people modify the environment "to cue recall, to speed up identification, and to generate mental images faster than they could if unaided" (p. 191).

Mediating Artefacts

Mediating artefacts include any artefacts that are brought into coordination in the completion of the task (Furniss, 2004). Mediating artefacts are too numerous to list but include language, writing, counting, maps, signposts, computer programs, mental models and diaries. In this research project, the focus is on artefacts that help mediate representations as they propagate through the Distributed Cognition system.

Representation-Goal Parity

In a Distributed Cognition system, matching external representations to goals makes cognition simpler. Furniss (2004) adopts Wright's Distributed Information Resources Model to suggest the way information is presented may ease cognitive tasks (p. 17). Wright in turn uses the example of Hutchins' speed bugs (Wright et al., 2000, pp. 12-13). A speed bug is a marker attached to the speed dial that indicates the speed at which the pilot must alter flap configuration. While it is possible for a pilot to keep an internal representation of the trigger number in his or her mind, doing this would also require him or her to constantly transform the pointer position on the analogue speed dial into a number, an internal representation to compare with the remembered trigger number. This is a more taxing cognitive task. Instead, by attaching the speed bug, the pilot can forget the trigger number and simply wait for the pointer to reach the marker, signalling the need for an appropriate action.

Looking for evidence of representation-goal parity, one must find examples of simplification of information in relation to such triggers (intermediate goals) or end-

goals. Furniss (2004) adds: "The closer the representation can be to the cognitive need or goal of the user, the more powerful that representation will be" (p. 17).

2.3.2.2 Information Flow Theme

The information flow theme analyses how information is processed, by whom and by what, how and why it is transformed. Focusing on communication between members, this theme defines where information arises and where it flows. It also documents both the roles and the sequence of events, and the transformations information makes on its way through the system. Lastly, it defines the kind of support structure the team may develop to manage the flow of information.

Information Movement

In order to collaborate, the stakeholders and software development team must move information between each other. It is important not only to define what kinds of information are involved, but also where it arises and how it is consumed in the system. In addition, attention must be paid to the mechanisms by which information moves (Blandford & Furniss, 2006).

Information Transformation

As information moves through the system, it is often transformed from one representation to another. The reasons for these transformations are many; transformations can be from written language to spoken language, from spoken language to a graphic representation. Or transformations can be as simple as paraphrasing what one has just said to make it more understandable. It can be drawing a process diagram on a whiteboard based on what the stakeholders say their system is. Another type of transformation is filtering, which Furniss (2004) significantly documented in his thesis on the London Ambulance Service control room. Each representation moves from one representational state to another, often when changing media.

Information Hubs

Often there is a central hub where different streams of information meet. An information hub is a locus of information processing. This hub may be the place where

decisions are made, or it may be the place where information is filtered, to be passed on to someone else to action.

Buffering

Information hubs often have a buffer to cope with the flow of information. An information buffer is a kind of "working storage" that allows the component working through the information at the hub to offload information and recall it when able to fully address it. As Hutchins (1995a) explains, buffering counters the impact of "destructive interference from processes running in parallel" (p. 195).

Communication Bandwidth

Each method of communication has a "bandwidth," an amount of information the method transmits. Face-to-face communication transmits more information than any other method (Furniss, 2004, p. 20). As such, this characteristic must be considered in the cognitive system.

Hutchins and Palen (1997) describe the ways of "constructing meaning" from the part of the face-to-face communication bandwidth that is not spoken language. They explain: "space, gesture, and speech are all combined in the construction of complex multi-layered representations in which no single layer is complete or coherent by itself" (p. 2).

Informal Communication

Communication can flow in both formal and informal ways, and it is important to consider what kind of information is being imparted in ways that may not be documented or specified.

Behavioural Trigger Factors

A team may "know" their roles so well that they do not need to be told or have a controller determine what to do next. The team may be able to function in response to appropriate triggers.

2.3.2.3 Physical Theme

The physical theme describes the physical environment of the Distributed Cognition system and how it affects the performance of the system and the individual components of the system. Revolving around the most influential senses of the human being – vision and hearing, the physical theme questions what can be seen, heard or electronically accessed by the team and the members of the team. These elements can "shape, empower and limit the calculations that individuals perform" (Blandford and Furniss, 2006, p. 29).

Space and Cognition

Just as white space on a web page can draw attention to the most important text, space can help a Distributed Cognition system solve problems. Hollan et al. (2000) point to Kirsh's three categories of space:

- Spatial arrangements that simplify choice;
- Spatial arrangements that simplify perception;
- Spatial dynamics that simplify internal computation (Kirsh, 1995).

Attention must be paid to the use (or not) of space as a cognitive device in a Distributed Cognition system.

Naturalness Principle

The components of a Distributed Cognition system – in particular members of the team – "easily, more reliably and naturally" use and understand "spatial and perceptual" representations whose "format of representation" maps to what they represent, as opposed to those rendered into an arbitrary formal symbol system (although the latter representations are more appropriate in some situations) (Norman, 1993, p. 72).

Perceptual Principle

This intuitive mapping described above in the Naturalness Principle, between the spatial and perceptual representations and the objects they represent are "natural" if they are analogous to the real perceptual and spatial environment (Norman, 1993, p. 72). For

instance, in most circumstances, it is not appropriate to model colours in Red Green Blue (RGB) numbers.

Subtle Bodily Supports

This principle identifies the use of parts of one's body to support cognition. Furniss (2004) uses the example of one following where one is reading with one's finger.

Situation Awareness

Focusing on how well components of a Distributed Cognition system are aware of the current state of the system, situation awareness examines how accessible information about the current state of the system is, whether there are any black boxes that hide cognition. Furniss (2004) points out that this is "influenced by the proximity of the person, involving both observation and overhearing conversations" (p. 19).

Horizon of Observation

Situation Awareness may be influenced by proximity, and horizon of observation often influences this proximity. If one's horizon of observation is nearer – if a wall separates one from co-workers – then understanding what is happening in the team takes more effort than in situations where the horizon of observation is much farther away.

Arrangement of Equipment

This principle parallels the space and cognition principle in that physical layout of equipment can affect ease or difficulty of choice, perception or computation. Discussing his study of navigation, Hutchins (1995a) notes that "the computational consequences of the locations of equipment may interact in unexpected ways with other aspects of the ship's operation" (p. 197).

The DiCoT method allows one to decompose a Distributed Cognition system into three themes – physical, artefact, and information flow. The identified principles within these themes aids in identification of what could aid cognition. These themes are an interesting perspective in which to explore such a system.

In summary, the theory of Distributed Cognition is a relatively new field of cognitive science. It posits a person in a loose coupling with the world. Cognition is both in and outside the human mind, or used together.

Aranda and Easterbrook advocate using Distributed Cognition to study SE. The work of Sharp and Robinson (2006b, 2009) provide a roadmap to using Distributed Cognition in the study of user stories and the story wall.

A review of Distributed Cognition literature confirms it as an appropriate theoretical lens for studying user stories in a collaborative team context. It also identifies a suitable framework, DiCoT, to systematically analyse a distributed cognitive system.

2.4 Summary of Literature Review

By exploring current practice as evidenced in the work of agile methodology founders, educators and advocates, this literature review developed a theoretical model of the user story process. While this model may not be used by any single organisation in practice, it gives a framework, vocabulary and context with which to discuss user stories in general and the AT user story process in particular.

From establishing this context, empirical research literature was reviewed. It was discovered that the user story process is relatively unknown. In addition, questions were also asked about how to study user stories. It became apparent that to understand deeply user stories in practice the research method should examine, as Dybå & Dingsøyr (2008) put it, “human and social factors” (p. 844).

The one significant body of work found exploring user stories is that of Sharp and Robinson (2006b, 2009). They use a Distributed Cognition lens to examine how a mature XP team works. From an examination of the empirical agile research, this review progressed to an exploration of Distributed Cognition theory and its appropriateness for SE research.

Thus, the outcome of the literature review – to develop “sharper and more insightful questions,” as Yin (2003, p. 9) enjoins – brings this thesis to question the prevailing reliance on observation as data. Thus, this research will employ a novel use of interview data in a Distributed Cognition analysis. This novel combination of data

collection and data analysis will be delineated in the next section, Research Design and Implementation.

3 RESEARCH DESIGN AND IMPLEMENTATION

The research design revolves around the nature of the research questions. The motivation of this thesis is to gain a deeper understanding of the use of user stories in practice through re-interpreting the user story process as a Distributed Cognition system. As discussed in Section 1.5, the overarching research question is:

> *How is a user story involved in software development as an artefact in a Distributed Cognition system?*

This is essentially an exploratory question, exploring practice "in the wild". The aim is to possibly generate new hypotheses to test, rather than testing hypotheses. It is expected that the insights from this study will provide some understanding to (re-) interpret other empirical studies. The next sections discuss the research design in light of this research question.

The components of a research design include selection and design of:

- Research methodology;
- Data collection methods;
- Data analysis methods;
- Research scope;
- Unit and level of analysis.

Each of these components is essentially a question to be answered; each limits the choices of the other components. The kind of research questions a study seeks to answer determines the kind of research methodology. The selection of a research methodology must be appropriate for the kind of questions being asked. Selection of a research methodology brings both the research methods such as the data collection and analysis methods into focus. All these components of the research design must be appropriate for the kind of research questions asked.

This paper is based on an empirical research project to answer the research questions. As this research question is of an explanatory nature in a contemporary setting beyond

the manipulative control of the researcher, the most appropriate research method is a case study (Yin, 2003, p. 13-14).

In essence, this research project is an interpretive case study of the use of user stories with a Distributed Cognition lens for analysis. In this case the data for the Distributed Cognition analysis is inferred from data gathered from interviewing practitioners about the use of user stories, probing the user story process as a unit of analysis. The specific framework of data analysis will be the Distributed Cognition for Teamwork method discussed in Section 2.3.2.

3.1 Research Methodology and Philosophic Approach

As the point of this thesis is to discover knowledge about the world, specifically about how user stories are used in one organisation, it is imperative to define what knowledge means in this study, and how it may be acquired. In particular, it requires a statement of what is believed to exist and how one knows how to say anything about it (Sjoberg, Dyba, & Jorgensen, 2007).

Orlikowski and Baroudi (1991) agree with this need to make explicit the assumptions "regarding the nature of the phenomena being studied" (p. 1). While there is no absolute classification of paradigms, this report adopts that of Orlikowski and Baroudi (1991). They suggest there are three main paradigms in which scientific research grounds itself: positivist, interpretivist, and critical.

This research project adopts an interpretivist approach. Thus, the interpretivist paradigm is the primary focus. However, as case studies can be applied to any paradigm, a contrast with the others is valuable (Easterbrook, Storey, & Damian, 2008). There are ontological and epistemological assumptions attendant to the choice of a paradigm.

Ontological and Epistemological Assumptions

Ontology is the study of the nature of reality. As mentioned above, it defines what one believes to exist. Each paradigm has an ontological basis of what can and cannot be said of reality. In the positivist paradigm, there are subjective personal values and there are external truths. The researcher can keep values and truths separate. The critical

paradigm takes the opposite position – each person constructs his or her reality. Thus one must maintain a critical view of one's values while doing research. Lying between the two, interpretivism sees a "reality-for-us as an intersubjective construction of the shared human cognitive apparatus" (Walsham, 1995, p. 75). A research project which studies people interacting with each other, artefacts and the environment fits appropriately with a paradigm that focuses on the interpretations and meanings people apply (Walsham, 1995, p. 74).

Epistemology is the theory of knowledge. At one time, the exponents of empiricism and rationalism disputed the legitimacy of scientific research. On the one hand, the empiricists advanced that one can only know what one sees, and thus the centrality of data. On the other, the rationalists ascribed to the supremacy of the mind, and with it, symbol systems such as mathematics.

Both of these schools had limitations. On the one hand, empiricist Hume extrapolated the emphasis on sensory data. He doubted whether people can ascribe causality to anything based on strictly sensory data. He questioned whether people can see such a thing as causality. Without causality, the basis of science came into question. On the other hand, while the rationalists had no problems with causality, they struggled with mind-body dualism: trying to reconcile which one was more real and the implication of how they interface with each other.

In the end, positivism united both of those schools by using data (empiricism) and statistical analysis (rationalism). The positivist methodology – the scientific experiment – advances a knowable external reality. If the results of an experiment confirm the hypothesis within statistical relevance, the hypothesis has been confirmed.

The first challenge to the hegemony of positivism was Karl Popper in the early twentieth century. He critiqued one of the bedrock concepts of positivism – that one can prove an idea by finding evidence that it is true, or verification (Thornton, 2009). Popper points out that that proof may be based upon incomplete knowledge. As the common aphorism goes, one may have never seen a black swan, but that does not mean they do not exist. Popper advanced falsification as the appropriate approach (Thornton, 2009).

Popper did not challenge the idea that there was a knowable external reality. He merely questioned whether science can validate it with averages and confidence intervals. Future information – the sighting of a black swan, for instance – can usurp previously unquestioned knowledge.

Later in the twentieth century, Thomas Kuhn (1970) explicitly challenged not merely the supremacy of positivism (post or otherwise) but that of any philosophic foundation. He put forward the idea of paradigms; "a criterion for choosing problems that, while the paradigm is taken for granted, can be assumed to have solutions" (p. 37). If positivism is a paradigm, there may be more. This concept of paradigms increased scrutiny on the idea of scientific truth.

In the years since, two new approaches have arisen: interpretive and critical. Of these two, interpretive is the paradigm of interest in this thesis. Interpretive is characterised by a belief that reality is constructed, as opposed to eternally the same for all observers. As a consequence, there is no longer a "wall" between the observer and what is observed. Interpretive work no longer starts with a hypothesis and defines variables to manipulate; rather, it uses existing theory to develop a research question and then examines the research question though data collection and analysis.

Interpretivism is of interest in this study as the researcher does acknowledge a constructed external reality subject to interpretation from the viewpoint of different agents. Thus, for a thesis with research questions concerning communication and sharing understanding (as described in Section 1.5), the researcher must be able to interpret reported activities as indicating cognition. Indeed, for a Distributed Cognition analysis, such activities *are* cognition.

While there must be an external reality that is shared among the members of the AT software development teams, the study of artefact use and the activities revolving around them depends on the participants' interpretation of this use. Likewise, the study of cognitive activities - both reported and inferred - incline this thesis towards the interpretivist paradigm.

3.2 Research Methods

Research methods for interpretive work in the field commonly are case studies and ethnography. As reported in the literature review in Section 2.2, researchers with similar aims to that of this thesis used such research methods. For example, the research method used by MacKenzie and Monk (2004), as discussed in 2.2.1 was ethnography. On the other hand, Yagüe, Rodriguez & Garbajosa (2009) performed a case study, while Gallardo-Valencia and Sim (2009) conducted a field study. In attempting to combine the best aspects of both methods, Sharp & Robinson (2006a, 2006b, 2009) use what they call ethnographically-informed case studies.

Ethnographic studies assume a participant-observer looking at the observed anew (i.e., without the benefit of theory). However, there are limitations of ethnographic studies. The participant-observer may actually affect the data; the participant-observer must be the ultimate interpreter. Lastly, observation can be quite lengthy in the case of ethnography. Even Sharp and Robinson (2006b) spend a week on site.

As this research project examines the use of user stories in practice, a case study is a logical candidate for research method.

But what exactly is a case study? Easterbrook et al. (2008) still believe it is not clear what case studies are. Geering (2007) concurs that the case study is a "definitional morass" (p. 17) and offers his own version:

> *A case study may be understood as the intensive study of a single case where the purpose of that study is – at least in part – to shed light on a larger class of cases (a population) (Geering, 2007, p. 20).*

This definition suggests that the case study is appropriate for this research as it is centred on a single case – the user story process of the IT division of a New Zealand financial company.

However, it is not altogether clear whether this study suggests it can "shed light on a larger class of cases. To move beyond this "morass," one must examine the landmark text of Yin (2003). He provides a detailed, five-part definition of the case study (p. 13-14).

The following Table 2 compares the characteristics of this research project with those of Yin's classic definition.

A Comparison of Yin's Definition of a Case Study with this Research Project	
Yin's definition (Yin, 2003, p. 13-14)	**This research project**
Investigates a contemporary phenomenon within its real-life context.	Investigates the use of user stories in practice in an organisation.
Is helpful when the boundaries between phenomenon and context are not clearly evident.	While the user story process is the unit of analysis, it cannot be abstracted from the context – the organisation, culture and people.
Copes with the technically distinctive situation in which there will be many more variables of interest than data points	The use of user stories in practice may involve many different factors that outstrip the ability to capture specific data.
Relies on multiple sources of evidence with data needing to converge in a triangulating fashion	The sources of information from different roles triangulate, along with information from artefacts.
Benefits from the prior development of theoretical propositions to guide data collection and analysis.	This research has benefited from the theory of Distributed Cognition to frame data collection and analysis.

Table 2 Comparison of Yin's Definition of a Case Study with this Research

An examination of Table 2 confirms the appropriateness of the case study method to the research aims of this study. First, this research is focused on a single case, which makes it appropriate for a method that examines "a spatially delimited phenomenon observed at a single point in time or over a period of time" (Geering, 2007, p. 19).

Secondly, as Salo and Abrahamsson (2004) point out, one key distinction in research methods that determine appropriateness is the "level of control" (p. 409-10). Studying

the use of user stories in practice inevitably means the researcher has no control over the context. Thus, a case study is useful when "the reductionism of controlled experiments is inappropriate" (Easterbrook et al., 2008, p. 11). Colin Potts (1993) locates value in the fact case studies can incorporate scale and complexity, unpredictability and dynamism - that cannot be reproduced in a laboratory.

A case study does not derive knowledge from sampling or statistical confidence but on a deep scrutiny of a small sample size. It is precisely this deep scrutiny that Runeson and Host (2009) see as an advantage of case study research. Woodside (2010) proposes that it is the "*deep understanding* of the actors, interactions, sentiments, and behaviours occurring for a specific process through time" that is the principal objective of case study research (p. 6, emphasis in original).

Another possible research method for this project is the ethnographic approach. Indeed, ethnography is an ideal candidate for a research project using a Distributed Cognition lens. Hollan and founder of the field of Distributed Cognition Edwin Hutchins advocates the use of cognitive ethnography (Hollan et al., 2000).

MacKenzie and Monk (2004) point to one of the core ethnographic techniques of regarding familiar activities as "strange" enables the researcher to not only discover new phenomena but perhaps re-discover something previously thought well understood. Sharp and Robinson (2004a) also find this technique useful to uncover implicit and previously overlooked features of practice. It also helps find the difference between what participants say they did and what they actually do.

Robinson et al. (2007) describe the field work of ethnographers as:

> *immersing themselves in the area under study for several months if not years, documenting what takes place via a range of means that can include contemporaneous field notes, audio and video recordings of discussion and meetings, photographs and sketches of the physical layout, copies of various documents and artefacts and records of interviews with practitioners (p. 541).*

For a research project examining practice, this kind of long-term immersive access is difficult to obtain. For this reason, researchers Robinson et al. (2007) have adopted ethnographic techniques into the case study method.

As an interpretive case study, this research project differs from Sharp and Robinson's ethnographically-informed case studies. First, the role of the researcher is different. Walsham (1995) advocates a detached observer in contrast to Sharp and Robinson's participant-observer.

Secondly, Walsham (1995) points to three different uses of theory in the interpretive case study – as an initial guide to design and data collection, as part of the iterative process of data collection and analysis, and as a final product of research. Robinson et al. (2007) in their ethnographic approach state: "we try to eschew *a priori* theory in our research".

3.3 Approach to Data

Both the data collection method and the data are qualitative. As qualitative research methods originated in the interpretivist tradition in social sciences, qualitative research is an understandable but not inevitable approach (Seaman, 2008).

However, there are reasons that the qualitative approach is appropriate in a study of agile development. Butler et al. (2001) point out that research into Extreme Programming "relies strongly on the results of impressions and feelings of developers within the environment" (p. 140). This assertion applies also to AT's agile environment. In the literature review, it is pointed out that the points of emphasis delineated in the Agile Manifesto are not easily quantifiable (Beck et al., 2001).

3.3.1 Data to be collected

Still, the dichotomy between qualitative and quantitative disappears if one follows the view of Miles and Huberman (1994). They assert that all data is qualitative as it refers to "the essences of people, objects and situations" (Miles and Huberman, 1994, p. 9). Qualitative researchers merely abstract away from the essences to numbers; qualitative researchers probe them to obtain understanding of the case (Easterbrook et al., 2008). Seaman (1999) agrees that qualitative methods allow us "to delve into the complexity of the problem rather than abstract it away" (p. 558). By studying qualitative data, this study seeks a *deep* understanding of the use of user stories in practice.

The specific qualitative data to be used in this study relates to understanding the user story process and the context within which it occurs. The description of this process is obtained from the remembered perception of practitioners who are involved in their use. This data collection is justified and discussed in the next section.

3.3.2 Data collection techniques

This study uses interview data to probe the experiences of practitioners in agile software development teams using user stories. Walsham (1995) argues that interviews allow access to how the interviewees understand their own experiences while allowing the interviewer to remain a detached observer. Although many claim the interview is an artificial situation (Myers & Newman, 2007), Robinson et al. (2007) view interview data as "in the wild" if the interviews conducted *in situ*.

The data this project has collected is primarily from semi-structured interviews. In addition, the researcher has been able to examine artefacts in the interview.

There are three primary types of interview, structured, semi-structured and unstructured. In its extreme, the structured interview can resemble gathering questionnaire data in a face-to-face manner, focusing on quantitative data (Seaman, 2008). The researcher either does not want or does not recognise the value of any unforeseen avenues of inquiry. On the other hand, it may be difficult to create consistent qualitative data from unstructured interviews.

Avoiding the problems of either of these options, the semi-structured interview is a valid candidate for the qualitative research method for an interpretive case study. Semi-structured interviews usually include specific questions to build a context from which the researcher can generalise. They also include open-ended questions designed to not only explore areas the researcher has identified in advance, but also allow the interviewee to give their interpretation without fitting into a template. In response, the interviewer has the flexibility to follow up on any of these potentially profitable, unanticipated avenues of inquiry.

The specific planned interview questions, and their relationships to the research questions, are shown in Section 3.4.2. Probing and follow-up questions were used to clarify answers or extend the discussion to cover aspects of the user story process

related to the planned Distributed Cognition analysis. This is discussed in more detail the next section.

3.3.3 Data Analysis

The decision to use Distributed Cognition theory as a theoretical perspective to understand collaborative teamwork involving user stories, as described in Section 2.3, influences the data analysis methods to a large extent. The aim is to re-interpret the described user story process as Distributed Cognition and identify the interactions of user stories with people, other artefacts and the environment as part of a Distributed Cognition system. This involves inferring and identifying instances of principles of Distributed Cognition theory from the description (interview) data.

One common data analysis tool is thematic analysis, in which the researcher builds a theory from these patterns from the codes. However, as this is an interpretive case study, and existing theory has informed the process from the development of the research question, the data analysis uses codes from a framework situated in the field of Distributed Cognition.

In this thesis, the method selected to systematically analyse the data is the Distributed Cognition for Teamwork (DiCoT), as discussed in 2.3.2. The first step in the data analysis is coding the interview data to the themes and principles outlined in this framework. Goodwin (1994) states that the value of data coding is that it is a way to transform phenomena into "objects of knowledge that animate the discourse of a profession" (p. 606).

After having transformed the interviews into a form that can be loaded into a qualitative data analysis tool, the researcher examines the interview data looking for significant "codes". These are accumulated and the researcher can then examine the codes for patterns and similarities related to the DiCoT categories and principles.

3.4 Implementation

As a study of how a commercial software development team practices, this research project investigates a contemporary phenomenon – the use of user stories. The boundaries of this phenomenon are not clearly defined, since one characteristic of user

stories is to reflect the desires and expertise of a wide variety of roles. This study also focuses on the "technically distinctive situation" of studying human beings in a situation out of the control of the researcher. It relies on the triangulation of the viewpoints of several people captured in interviews, along with field notes. Walsham (1995) sees as interview data as being of two different "orders":

> *Second order concepts rely on good theory and insightful analysis, and mere collection of in-depth case study data does not provide these concepts (p. 75).*

This project endeavours to match theory and insightful analysis to the second order concepts in the interview data.

3.4.1 Development and Implementation of Research Process

Having formed a motivation to research user stories in practice, the researcher examined research literature to find out what is empirically known about them. As depicted in Figure 3, having established a research motivation and reviewed research literature, research questions were developed. From that, the data collection methods were interrogated and interview data was decided upon. As this research report is proposed through a Distributed Cognition lens, the Distributed Cognition of Teamwork method was examined.

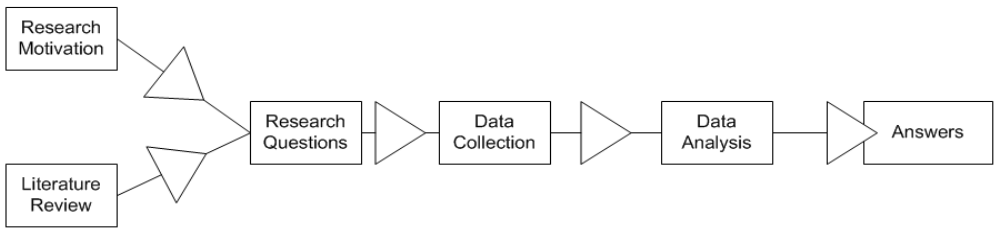


Figure 3 Research Process

This thesis is a process analysis of a collaborative process from an interpretivist point of view. The phenomena are conceptualised as a Distributed Cognition activity in order to gain a deeper understanding of how the user story is used in practice.

Thus this research is designed to focus on a specific case – an organisation and its process – and analyse that process through a Distributed Cognition lens. Using Distributed Cognition theory focuses on specific aspects, on specific data and a unit of analysis.

3.4.2 Case Study Protocol

3.4.2.1 Research Plan

Develop AT User Story Process

The purpose of the interview data is to:

- Analyse the user story lifecycle in the context of the software development process;
- Identify benefits and challenges related to user stories;
- Compare these results with research literature.

The research plan (Figure 4) of this thesis focuses on the development of a detailed description of the AT user story process based on the interview data. This description maps AT's user story process to the conventional software development process and requirements engineering (requirements elicitation, analysis, specification, validation). The AT user story process will be compared and contrasted with the user story process detailed in the literature review.

The detailed description is the first step in the analysis of the Distributed Cognition system.

Distributed Cognition analysis

Using the NVivo qualitative data analysis application, the transcribed data is coded for cognitive activities and DiCoT principles. Through a process similar to a hermeneutic circle, the researcher worked back and forth, from the DiCoT principle codes to the detailed user story description, in order to develop a deeper understanding of the cognitive activities of the system.

Familiarity with the data allowed the researcher to develop patterns. By pattern matching from the data to cognitive activities and the DiCoT principles, this study establishes a chain of evidence from the interview data to answering the research questions.

With this evidence, inferences can be made to develop a Distributed Cognition system, labelled the Consequences in Figure 4. This analysis is used to develop a discussion of the cognitive activities involved in the AT user story process. Benefits and challenges are explored and compared and contrasted.

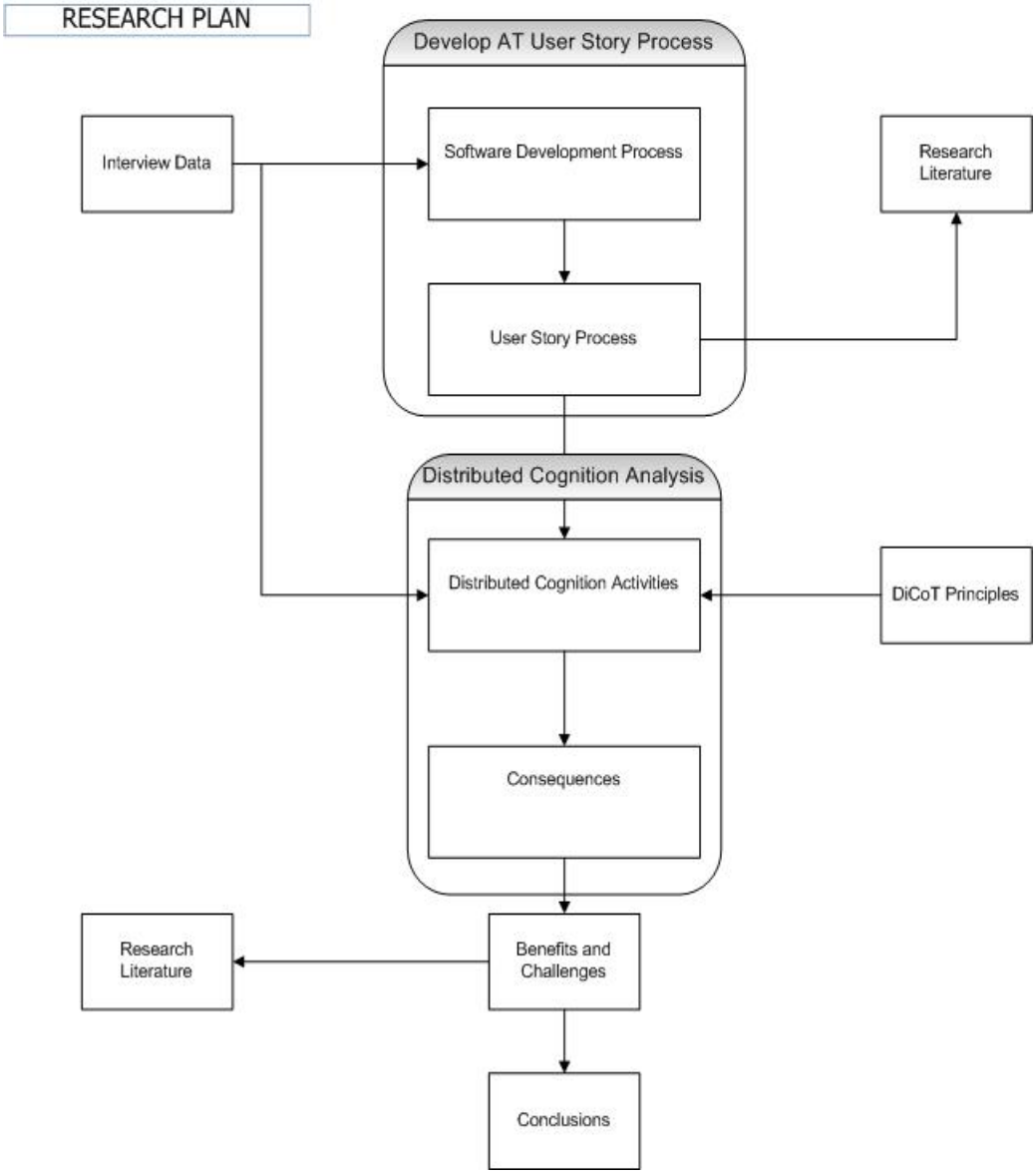


Figure 4 Research Plan

3.4.2.2 Unit of Analysis

The unit of analysis for this case study is the user story process as described by members of AT at Company A. This unit of analysis involves people, artefacts, and the environment as part of the process. While this unit of analysis is a cognitive system, it will be referred to as a "functional system". This is an emphasis Hutchins (1995a) gives to focus on the structure and process of cognition external to the human agents (p. 142).

3.4.2.3 Scope of unit of analysis

Having defined the unit of analysis, the next step is to delimit the scope. Hutchins (2010) suggests choosing the boundaries for a functional system where "connectivity is relatively low" (p. 706). The user story process is defined as from the initial idea of the stakeholder to the moment when one of the software development team moves a story card into the final "Complete" column on the story wall. At that point, the focus changes to that of the software created. The software at this point may be combined with that of other and earlier user stories to become a high-level feature. The user story loses relevance.

While the user story process may seem "connected" to an approval process or legacy deployment schedules, these have little impact on the actual user stories, and thus are out of scope. Likewise, those individuals, processes and artefacts not involved in the creation, development or implementation of user stories are also out of scope.

3.4.2.4 Participant Selection

Contact with Company A commenced in November 2010. The researcher approached the AT Agile lead, a prominent agile professional in the Auckland area, about the possibility of a research partnership. Eight interviews were carried out in February 2011. The Agile lead emailed AT for interview volunteers. Thus, each interviewee self-selected themselves. Their confidentiality was assured.

As evident from Table 3, the participants span several roles in the typical AT software development team. Most are experienced software development professionals and have been involved in the agile implementation at Company A. Thus, their answers to the interview questions draw upon considerable software experience, both in traditional development methods and in AT's Agile.

The comfort and privacy of the participant was considered for the interview venue. Each interview was carried out at the Company A premises. The interviewees selected where the interview would take place. Most interviews were held in small meeting rooms off the main Company A reception. One meeting was held in the cafeteria in the middle of the afternoon. Another was held in a meeting room on the AT floor.

AT Research Participants				
Role	**Length of Interview**	**Years in IT**	**Years at Company A**	**Years using user stories**
Business Analyst	1:22	10	3	5
Business Analyst	1:12	12	3	3
Project Manager	1:20	3	3	2
Project Manager	1:02	8	8	2
Project Manager	1:00	15	3.5	3.5
Developer	1:11	17	12	4
Developer	1:09	30	10	2.5
Developer	1:54	20	9	3

Table 3 AT Research Participants

3.4.2.5 Implementation of the Interview Process

Context

Not only is an account of the context necessary to report a case study (Yin, 2003, p. 22), it is also necessary for a Distributed Cognition analysis. Context is required for generalisability. The first questions of the interviews focused on contextual questions.

Semi-structured Interviews

The interviews were conducted by the researcher and, on occasion, the researcher's supervisor. The interviews were guided by the interview question form (APPENDIX B).

Connection between interview questions and research questions

To motivate and explain the data, the interview questions need to reflect the research aims and in particular the research questions. There questions essentially centre on two domains. They aim to document the software development process and explore the kinds of evidence of cognitive activities shared by people, artefacts and the physical environment. The last question attempts to tease out benefits and challenges.

The following tables show the relationship between the research questions and the interview questions (APPENDIX B).

The first research question is linked to three interview questions, as depicted in Table 4. The interview questions focus on the creation (elicitation and verification), maintenance and use of user stories.

Research Question	**Interview Questions**
What is the software development process and how are user stories involved?	**Can you please describe the overall process of how a user story is created?** How is the client involved? Who else is involved and what is their involvement or goal(s)? (How are you involved?) How is the content of the user story verified with the client? (Who, when where why?) Who gets a copy of the completed user story (how when where why)? Any standard ways of doing things? Templates?
	Are User Stories ever updated/changed? Why? Describe the overall process for this - from the trigger to the change being incorporated. Who is involved? How, when, where, why?
	What use is made of the user stories? (Who, when, how why?)

Table 4 Research Question 1 Linked to Interview Questions

The interview questions centre on the usual procedure: the process, the roles involved, the output and any other procedure. The questions are concerned with the process of user story creation.

The next three research questions listed in Table 5 map to two main categories of interview questions. The first research question examines the areas and events in which

one would expect to find Distributed Cognition activities. The second two research questions attempt to categorise such activities in ways often identified in elicitation and verification.

These interview questions focus on understanding what information is required during the process of user story creation, maintenance and the use of user stories. It is by looking for the information, the questions will highlight the flow and transformation of representations and the use of artefacts to support and sustain cognitive activities.

Research Questions	**Interview Questions**
What distributed cognitive activities involve user stories as a cognitive artefact? How do user stories support Distributed Cognition in terms of sharing understanding? How do user stories support Distributed Cognition in terms of communication?	How is the information from the client obtained? (Represented? Meetings? Who is involved? Physical layout of the room? Whiteboard? Are there any other ways the user story is modelled? Is there any other way a user story is represented? Are there any other intermediate representations/artefacts? How is the information from the client analysed and used to make a user story? (Who when where why) What other information is used to create user stories How is the physical representation of a user story created (who, when where why?) How is the content of the user story verified with the client? (Who, when where why?)
	(What information is used to make the change? How is this represented? How is it shared? What other artefacts involved?)
	What information from user stories is used in planning? ….designing?

	….coding …testing …other activities?? What information from user stories is used to create other artefacts? Info on user stories-how used? User story states?

Table 5 Research Questions on Cognitive Activity Mapped to Interview Questions

These interview questions in Table 6 map to the question of benefits, challenges and limitations of the user story process.

Research Question	**Interview Question**
What are the benefits and challenges of using user stories in a distributed cognitive system?	What do you think the main purpose of a user story is? (In your role?) Different at different times of the development lifecycle? What are the most important pieces of information on a user story? And what are they used for? (Do you use them for) during different phases of the lifecycle. Who (what role) do you think a user story is most important to? Why? Do you think user stories contribute to evolving and sharing understanding of the business requirements? How?

Table 6 Research Question on Benefits and Challenges Mapped to Interview Questions

They are designed to allow the participant to freely discuss what they think of user stories and relate experiences from which the benefits and challenge can be culled.

These tables emphasise the purpose of the interview questions in relation to answering the corresponding research questions.

Data recording

Data was recorded through field notes and a digital recorder, after asking the interviewee if they objected to the voice recording. Interviews extended from one hour to nearly two. Eight interviews were conducted, totalling over 10 hours of interview time.

3.4.2.6 Implementation of Data Analysis

After all the interviews were completed, the researcher transcribed the interviews.

Besides the initial interview and during the transcription, the researcher has listened and taken notes of all interviews at least three times. The researcher has read each of the transcripts at least twice. The repeated visits to the data enabled the researcher to develop a deep understanding of the data and the user story process as a whole. This understanding was used to create a detailed description of the user story process.

The interview data was then loaded into NVivo 9, the qualitative analysis tool. The data was coded according to two different schemes. The first is to identify cognitive activities. The cognitive activities preliminarily identified are:

- Breakdowns
- Clarifying
- Communication
- Information Seeking
- Problem-solving
- Questioning
- Sharing Understanding.

Through the repeated analysis of the interview data a more comprehensive list was developed, discussed in Section 4.4.

The second coding scheme applied consists of the principles of Distributed Cognition for Teamwork. This framework is the Distributed Cognition for Teamwork (DiCoT) framework first developed by Dominic Furniss (2004) in his research of the London Ambulance Service.

The data analysis centres on developing a DiCoT analysis. For each theme, Furniss proposes at least three output artefacts:

- A set of principles that pertain to that theme;
- A set of diagrammed representations that illustrate the structure and flow from the relevant perspective;
- A set of tabular representations that present a summary of the system, details, further observations and emerging issues (Blandford and Furniss 2006).

In this study, the tabular representation has been dropped as too unwieldy. As such, there is no need for a summary, and the details have been made more descriptive for each theme. Thus, "Detail" in the Physical Theme has been replaced by "Use in Physical Environment," in Artefact with "Use of," and in Information Flow with "Flow of Information". Secondly, because there was not an observation component to the data collection, certain aspects are unable to be diagrammed. AT interviewees provided the researcher with illustrative story wall photographs.

In addition to these formatting changes, there are significant differences between Furniss' use of DiCoT and this research that have required reporting changes. As mentioned above, Furniss (2004) studied the Central Ambulance Control room of the London Ambulance Service. This team was hierarchical and each role had specific and non-overlapping functions (Furniss, 2004). Indeed, the call takers followed scripts (Furniss, 2004, p. 33). Information flowed in one direction only.

As such, it is easier to develop agent-based diagrams since this flow of information is one way. In the AT user story process, while there are roles, communication and understanding flows in many if not all directions. When the interviewees testify to these multiple levels of communication, it is impossible to develop a Furniss agent-based diagram.

Likewise, the use of DiCoT by Sharp & Robinson (2006b) focus on the one team, a difference with this research report which studies a process shared by the members of the IT division of a New Zealand company. Sharp & Robinson (2006b) can develop detailed descriptions of their team and its artefacts and environment while this research

study focuses on triangulating the different interviews to develop a detailed description of the AT user story process to use as input to the DiCoT analysis.

3.4.3 Research Design Validity

An interpretive case study using qualitative data requires different kinds of tests for validity than a quantitative study would. The tests for validity revolve around:

- Objectivity – Whether the researcher and chosen methods are objective;
- Validity – Whether or now the research finding accurately represent the case;
- Reliability – Whether the results of the study can be reproduced;
- Generalisation – Whether the results say anything about the world as it is.

3.4.4 Limitations / Threats to Validity

There are several threats to validity that must be acknowledged.

Possible case study weakness:

- Easterbrook et al. (2008) claim the possibility for interpretation and researcher bias in data collection and analysis is greater in a case study than other methods. One suggestion to overcome this possibility is an explicit framework for data collection.

  In this research project, this framework is provided by the coherency of the research questions and interview questions, and an awareness of the possibility of researcher bias.

Possible interview weaknesses:

- Participant Range Problem: This research was based on interviews with the software development team. Due to circumstances, the researcher did not get the opportunity to recruit stakeholders to participate in this study, in particular the product owner. The researcher recognises that this is a limitation as we do not have data from the stakeholder perspective. Thus, our description of the user story process is biased towards the software development team.
- Self-Selection Problem: There may be a tendency for the kind of participants who self-select that they want to take part because they think positively of the

subject. If so, there may be people who have a different perspective that is excluded from data collection.

However, this in and of itself is not necessarily a threat to validity. It is imperative however, the researcher to keep this potential problem in mind.

- Comfort Factor: The interview situation may feel strange to the participant and make them feel uncomfortable (Myers & Newman, 2007).

  To counter this possibility, the researcher went to the participant's office. The participant chose where to have the interview.

- Trust: The participant may not trust the interviewer. This may mean the interview is not as accurate as possible (Myers & Newman, 2007).

  The researcher endeavoured to put the participant at ease while maintaining a professional presence.

- Reflexivity Problem: The interviewee replies with what he or she thinks the interviewer wants to hear (Yin, 2003, p. 86).

  One possible way of overcoming this problem is to structure open-ended questions that do not indicate what the researcher expects to hear.

- Problem with Reliability: As semi-structured interviews may vary even when interviewing the same interviewee with the same open-ended questions, it would be difficult to get repeatable results. However, it is hoped that if the researcher had another eight interviews at Company A the researcher could build a similar detailed description of the user story process.

- Problem with Validity: the researcher does not know whether or not the interviewee is exaggerating (or even lying).

  By using the multiple interviews to document a process, the redundancy of information gives the researcher confidence the responses are valid.

- Problem with Generalisability: It is difficult to generalise based on one interview. However, the researcher obtained eight interviews. The researcher used the interviews to build a process, using the interviews to "triangulate" the result.

- Inaccuracies due to poor recall/inaudible audio recordings (Yin, 2003, p. 86). Recording the interview is a good way of solving the poor recall problem.

  However, it is inevitable there are moments in which the recording is not clear. Repeated listening sometimes clarifies these incidents.

Possible researcher bias:

- The researcher is an ex-developer. This characteristic could lead to bias if the researcher relies on the researcher's experience to collect data to justify any pre-existing view of software development. Thus, the researcher must be aware and ward against interpreting the data through the researcher's own experience.
- Like the self-selected participants who one might suppose are interested in the topic, the researcher finds agile ideas attractive. This possible bias would manifest itself in the researcher only noting and interpreting what is perceived as positive in the findings. Again, the researcher must guard against this possibility and evaluate the evidence on its basis alone.

3.5 Summary

This chapter justifies the research design based on the nature of the research aim and questions. An interpretivist approach is adopted using a case study methodology. The unit of analysis is the user-story process viewed as a Distributed Cognition system and Distributed Cognition for Teamwork (DiCoT) is used as a framework of analysis. Information about the user story process is obtained from interviews of practitioners and this is used to characterise the process from a Distributed Cognition perspective. It is acknowledged that the analysis based on interview data is likely to result in a "lighter" Distributed Cognition analysis compared to one using observational data. Several limitations and threats to validity are identified and discussed.

The next chapter presents the findings from the implementation of this research design and discusses the implications of these findings.

4 FINDINGS AND DISCUSSION

This chapter answers the research questions by examining the evidence from the analysis of data collected. This starts with a description of AT's software development process in Section 4.1. This provides a context for the user story process that could influence the inference of the user story process from the interview data. Section 4.2 provides a detailed description of the user story process based on the interview data. In Section 4.3 this detailed understanding of the activities, representations and transformations related to the user story process are mapped to the three themes from the DiCoT framework. In Section 4.4 this Distributed Cognition perspective of the user story process is then used to explore and analyse the collaborative work activities and the cognitive activities involved in the user story process. This provides a detailed understanding of the user stories as part of distributed cognitive system. This distributed cognitive system is compared to a cognitive ecosystem in 4.5. The benefits and issues of the user story process as identified by interviewees are discussed and explained in light of the Distributed Cognition model in Sections 4.6 and 4.7 respectively.

4.1 The Software Development Context

Company A's internal IT division, AT, began introducing agile three and a half years ago. The interviewees consistently describe their teams as now completely agile. All interview participants use this term "agile" when referring to AT's software development methodology. To an outside observer it is a combination of XP development techniques and Scrum project management techniques, most notably the stand-up meeting and product backlog.

Comparison of interviews shows that the perceived implementation of Agile at Company A is consistent, at least at a high level. The participants frequently use the same words to describe the user story process. On evidence of the interviews, the agile development implementation seems to be well accepted. Participants are enthusiastic about the use of Agile in their software development projects. They sometimes use emotional words about certain aspects: "I love my wall". Participants sometimes compared Agile favourably with the previously used waterfall method.

Despite this apparent consistency, participants often warn that other teams may "do things differently". One participant says: "We have some standards and guidelines and things, but each [team] does things how they see fit based on the people they have on that team".

This observation attests to the freedom teams at AT are given to shape the Agile process to a particular project, most often depending on the size of the project and the characteristics of the stakeholders and project team. For example, project managers use different techniques to gather user stories. Similarly, there are differences among the developers depending on the underlying programming language paradigm. The legacy developers have a much longer time frame for software release, but they have adapted to the quicker releases of agile methodology. Another aspect of difference is the team size which varies from a reported low of four people to as many as twenty-five.

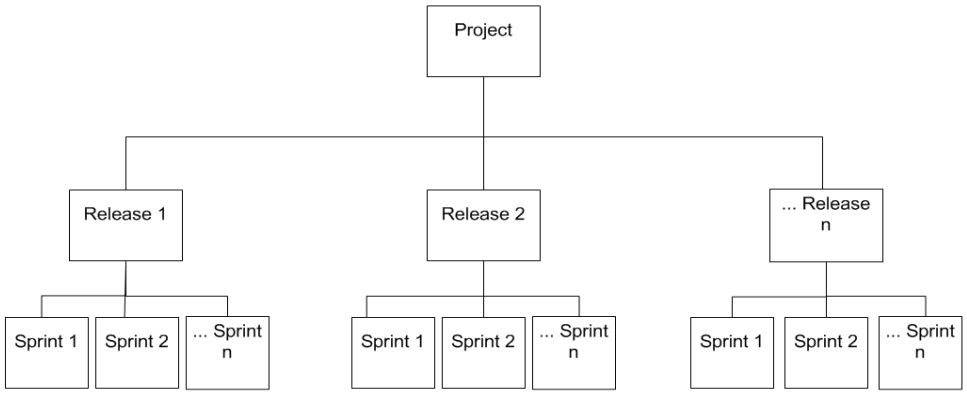


Figure 5 AT Project Hierarchy

Software development at AT involves project, release and sprint planning. The AT project hierarchy is illustrated in above Figure 5. It shows how the project is broken into a number of releases, which are in turn broken into a number of sprints.

Each planning stage requires user story elaboration, estimates and prioritisation at an appropriate granularity to change, as one participant says, the "twinkle in somebody's eye" into a useful software tool.

Several participants use the terminology of Company A's parent company to describe the software development process as consisting of three phases: Concept, Initiate, and Delivery. Each of these phases is crucial to the creation and use of user stories.

During the Concept Phase, an AT project manager will liaise with a group of key stakeholders to develop what is variously described as "high-level features" and "main functionality". These key stakeholders are often the project owner and the project sponsor. However, it is dependent upon the project.

At this stage, the project manager often recruits a business analyst and technical lead to develop rough time estimates. The stakeholders will apply priority to the high-level features, assigning them to releases.

It is during the Initiate Phase that user stories are created. The primary way the user stories are created is through stakeholder workshops. The end products of these workshops are story cards and eventually, electronic records in Jira, the open source issue tracking application AT has adopted for project management. The final stage of the Initiate Phase is the sprint zero, during which the details of the sprints for the current release are worked out with current software development team and the key stakeholders.

Lastly the Delivery Phase consists of a series of iterations called sprints. Over these sprints, the functionality of the high-level feature envisioned for this release is incrementally created. The development team maintains a story wall upon which the story cards are displayed within work phases according to the different roles that are required to develop working software – Analysis, Development and Testing. This story wall is a key component of project management.

4.2 User Story Lifecycle

The user stories are created in the Initiate Phase and implemented in the Delivery Phase. However, attention must also be paid to the Concept Phase in which preparation for the project begins. It is at this phase that the first user stories may emerge as high-level features. The user story lifecycle will now be discussed in relation to these three development phases in the next three subsections.

4.2.1 Concept Phase: Laying the foundation for user stories

There are several outcomes of the concept phase, according to the interviewees:

- The project manager understands the system as it is;
- The project manager gathers from the key stakeholders the high-level features;
- With a business analyst and technical lead, the project manager produce early-phase estimates of how long it would take to develop these high-level features;
- The stakeholders determine high-level priority of these features leading to early release plans.

Interviewees identify a number of activities taken towards achieving these goals, as described in the following.

At Company A, three different kinds of customers for software development projects were identified. These are: (1) on-sellers of Company A financial services, (2) other large financial organisations, or (3) an internal team. These different types of customer sometimes require different approaches to achieving these goals. Where this is significant, it is mentioned in the following discussion.

Often the initial discussions about a project are held over coffee with one of the Company A's Customer Relations Team and the client. If the project progresses beyond these informal discussions, a project manager is assigned to the project. The details of this approval are beyond the scope of this study since it has little direct relationship to the user story process. The project manager begins conversations with the key stakeholders, usually the project owner and possibly the project sponsor.

One common goal for project managers at this stage is to develop an accurate picture of the system to be changed. However, the ways in which they accomplish this task are various. Usually in the initial meetings with key stakeholders, project managers may develop a representation of the current system. Some build process diagrams; others create "As Is" diagrams. Still another reports developing a "Strengths, Weaknesses, Opportunities, Threats" (SWOT) analysis. These representations are similar to the "low-fidelity prototype" which Cohn (2004) suggests are useful triggers for generating user stories later in the process, as described in Section 2.1.1.

Interviewees note one problem project managers often encounter during this stage is that the key stakeholders do not really know how the end users use the system. They may know what the system was originally created to do and they may have read the user manual. However, there may be several layers of management between these stakeholders and the end users. The participants describe two main approaches that project managers use to understand the reality of the use of the current system "in the wild": interviewing end users and observing end users. In both these methods, participants echo what Cohn (2004) suggests for supplementary "trawling" techniques, as described in the literature review in Section 2.1.1.

With the key stakeholders, the project manager develops the "high-level features" and "main functionalities" for the project. These are the ideas that will become the user stories, or more probably, generate many user stories. Lastly, the project manager often recruits a business analyst and technical lead to use their experience to estimate the time it will take to develop the project at a high level. With these estimates, the stakeholders will prioritise the high-level features into the tentative releases for the project. Each release delivers significant useful functionality. The highest priority high-level features form the input to the Initiate Phase, which is discussed next.

4.2.2 Initiate Phase: The Creation of User Stories

It is the general view of the interviewees that the user story process begins in the Initiate Phase. Figure 6 gives an overview of the story creation process as described by the interview participants.

The first step of the Initiate Phase is the stakeholder workshop. The point of these workshops is to gather requirements for the release. While it is possible for a small project to need only one workshop to create all the user stories needed, usually there is a series of stakeholder workshops. Sometimes there is a need to have the buy-in of a large number of stakeholders, requiring a number of workshops. If the application crosses several specialised domains, there may be smaller, domain-specific workshops in addition to larger ones that deal with the application as a whole.

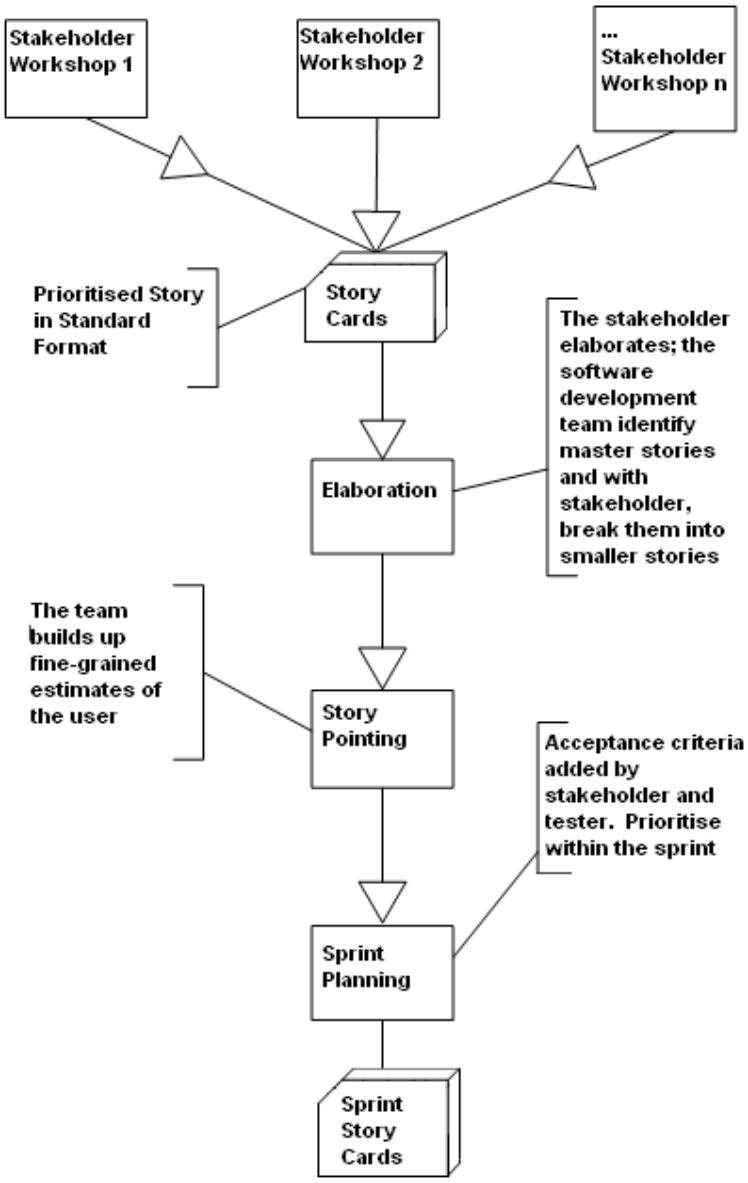


Figure 6 Story Development Process

Another potential challenge for the facilitator observed by the interviewees is the experience and ease with which stakeholders can participate in the workshop. This is also noted in literature by Wake (2003) and Cohn (2004). Some stakeholders may be experienced; some have even been sent on agile training courses. However, for some, participation in these workshops is a rare event. As one interviewee describes it, “some

of them may not even know where to start". To address such issues, the facilitator may arrive with example pre-written user stories, or frameworks and foundations to give them an idea of what is expected.

Typically either the project manager or the business analyst may facilitate the workshops. In general, the facilitator passes out index cards (or Post-it notes) to the stakeholders and gives them time to brainstorm user stories, one idea on one card. As previously described in the literature review by Gallardo-Valencia and Sim (2009) in Section 2.1.1., most interviewees report they encourage the stakeholders to use the standard format so that they will think of the role and context of any user stories: one interviewee describes this as "the sooner you do the 'As-a-I-want-so-that,' the sooner everyone is on the same understanding".

After ten or fifteen minutes of brainstorming user stories, the facilitator calls the workshop to order. Starting at one side of the room, the facilitator asks each stakeholder in turn to read aloud one card. Every one discusses the user story, and if there is a general consensus that it is a good, useful story, the story card is displayed prominently, either on a wall, whiteboard or flipchart.

If the aim of the workshop is to generate high-level features, the facilitator may ask the stakeholder to write their story on the whiteboard, as there are relatively few high-level features. The facilitator asks the stakeholder to then read it aloud, a feedback loop that unearths "subtle nuance".

Either kind of meeting continues with the next person and their user story.

At this stage, one facilitator interviewee reports, it is useful to group the story cards into domain areas to "get a picture of the kinda broad scope". For example, be a lot of stories that can be grouped into the domain of "customer details". This categorisation often helps the facilitator plan future domain-specific workshops.

When issues arise, AT facilitators commonly use techniques reported in the typical user story process, as described in Section 2.1.1. One such technique several participants report using is the "Parking Lot" – a flip chart on which the facilitator can "park" contentious issues. Another reason to park an issue is if a stakeholder needs to

investigate an issue around the user story. One participant reports using a “Smells” flip chart.

At the end of the process, the facilitator ends up with a stack of user stories or a list of high-level features (which will be used as input to another round of stakeholder workshops). In addition, there may be a process diagram or a series of screen mock-ups developed with the stakeholders on a whiteboard. Some interviewees report using cell phone cameras to capture the whiteboard graphs.

The business analyst will use this information to work on the user stories after the workshops. One business analyst task is to identify the “epics” or “master” stories. These are stories that are still at a very high level of abstraction and need to be broken into smaller stories.

Business analysts also identify the need for design spikes or “investigate stories” outside these workshops. Both of these techniques are used when there are stories where the technical feasibility is uncertain. A design spike is relatively informal. It may simply involve the business analyst asking a developer whether something is technically plausible, or how difficult it would be to implement. An “investigate story” is a description of the required investigation captured on a card. They are used in planning, similar to story cards, and are even put on the story wall. An “investigate story” is one of several different kinds of stories that the software development team create.

The business analyst may also identify gaps in the user stories generated in the user story workshops and may create “placeholder” stories to bring back to the next workshop. These stories typically involve functionality such as security or administrative rights or stretch across several functionalities.

An agile purist may blanch at anything but user-generated stories on the wall. However, one participant argues that it is difficult to plan using the story wall if this kind of work is not on it.

At the final workshop, the stakeholders prioritise using the MoSCoW technique as described in Section 2.1.1. This process is the second step in Figure 6. This stage

corresponds to the "sort by value" step in the standard XP user story process (Section 2.1.1.1). Several facilitators specifically mention the need for a large boardroom-style table. The story cards are spread out in a line down the middle of the table with each corner corresponding to a MoSCoW category. Working with each other, the stakeholders move the cards to the appropriate corners.

Most interviewees state that after prioritising the user stories into the four categories, they probably will only take the "Must Haves" for the current release. At that point, many interviewees report that they repeat the previous technique with the "Must Haves" in a line down the centre of the board table.

The following excerpt from the transcript illustrates the simple decision algorithm and language used to achieve the prioritisation.

> *And then, so then, what you have to do is take two musts and put them side by side and say, 'Which one's more must?'*
>
> *And they go, 'Ah, that one.'*
>
> *Then you go, 'That's priority one. Which one's more?'*
>
> *'That one.'*
>
> *'So that's priority two. Which one's more?'*
>
> *'That one.'* ***Business Analyst***

Essentially prioritisation becomes a binary decision between the card the facilitator is holding and the next card in the line until the held card finds its place.

Interviewees observe that throughout the prioritising exercise, the project manager or business analyst needs "extremely strong facilitation skills". Sometimes stakeholders will oppose any user story they do not understand. Other times stakeholders who did not challenge the value of a user story during the previous MoSCoW prioritisation while object to whether it truly is a "Must Have." Often, the facilitator may make use of the Parking Lot to set contentious user stories aside.

All user stories written on story cards are duplicated electronically in the Jira project tracking tool. While there is no rule of when exactly to enter the user stories, most business analysts believe it is better to wait until after the stories have been prioritised

so that they only have to update each story once. Therefore, if it has not already been done, the business analyst will enter the user stories into the Jira at this stage. This use of Jira is so ingrained into the culture of AT that "JIRA" has come to mean the electronic version of the user story as stored in the software application. Throughout this study "JIRA" will be used to indicate the electronic user story, and "Jira" will be used to refer to the software.

In addition to the project tracking system Jira, each software development team sets up a project area on the wiki. This wiki has several purposes.

First, it holds information that does not easily fit on a story card or in Jira, such as tables of algorithms, tables of statistics or complicated regulations. Indeed, information may have already been accumulating in the wiki even before there are user stories.

Second, as more and more projects add information to the wiki over time, the stakeholder might simply need to supply a hyperlink to the necessary information, as it may have already been loaded by another project. The benefit for Company A of using a wiki is that it can both standardise information and reuse it easily. In addition to domain information, the wiki holds a "shared code base" so that all teams can share software components and basic user interface conventions.

Lastly, as the wiki is persistent, production support uses it to understand the applications they support.

One participant calls the wiki a "living documentation repository". Effectively, it acts as a compendium of partial solutions to frequently-encountered problems (Hutchins, 1995a, p. 353).

4.2.2.1 Sprint Zero

The next stage is sprint zero. Sprint zero is described as the "sprint ahead" for sprint one. During a sprint ahead, the business analyst works on the user stories for the next sprint while the developers and testers work on the user stories for the current sprint. As one business analyst describes it: "development are working on that chunk of cards, and I'm working on this chunk of cards, while I support that chunk".

However, sprint zero is described as more than the first sprint ahead. There is a certain amount of planning and preparation for the entire release. The business analyst needs to gather enough information about the user stories so that the software development team can create code. At the same time, the software development team works on design spikes and perhaps investigates what is available in the open source world.

Once the business analyst has enough information, there is story elaboration, the next step shown in Figure 6. The elaboration process is broken down in more detail in Figure 7, illustrating its iterative nature. As it sounds, the purpose of this stage is to make sure everyone on the team understands the user stories. Some participants report stakeholder involvement in elaboration; others state the business analyst is the one to elaborate. Regardless, the facilitator goes through the user stories, explaining each in detail. The development team may ask questions or make suggestions. They may alert the stakeholder to technical dependencies that require the story priority to be changed.

Likewise, the developers and testers may realise a story is still too big and request that it be split. One developer believes one of the most important development roles is “to break [the stories] down into small, small stories”.

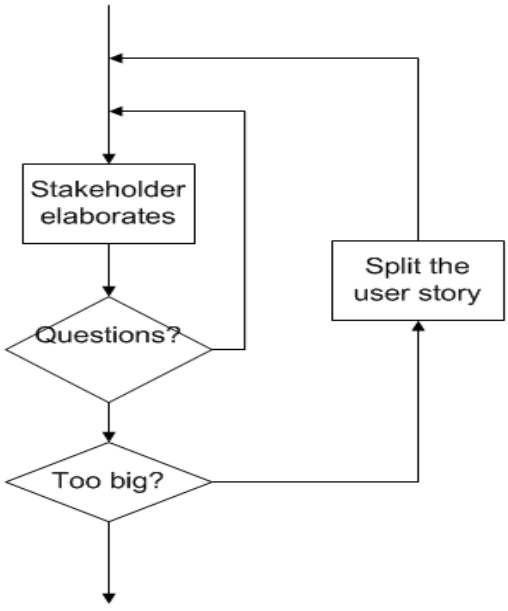


Figure 7 Elaboration as Iterative Transformation

Ideally, a user story should be deliverable in two or three days; otherwise, it will not be what interviewees describe as “trackable”. For AT, “trackable” user stories are ones that move through the sub-phases on the story wall quickly. Participants point out that if they created user stories that take three weeks to complete, they may not know there is a problem with a user story until the end of the sprint.

However, if the team is able to create user stories in roughly two or three days effort, if a story does not move it calls attention to itself. Everyone looking at the story wall on a daily basis gets the feedback that there may be a problem with that story card, and they help resolve the problem.

Another participant points to the desirability of having available a number of small size stories to fill whatever small gaps of time that may open up during the sprint.

After everyone in the software development has a comfortable shared understanding of the user stories, the software development team estimates how long it will take to develop each user story (the fourth step in Figure 6). As with the elaboration step, estimation may be iterative, as depicted in Figure 8. Most interviewees report the development team plays the “story poker” game, Grenning’s (2002) estimation technique described in the literature review. This process is also known as “story pointing”. The team goes through the stories once again, and each selects a card with a number of “story points” they think is appropriate to represent the amount of effort it will take. These numbers usually are but not always from the Fibonacci series. After estimating, there may be the need to break the story into smaller stories.

At this point, the team estimates their velocity. If this particular team has worked together on another project, as sometimes happens, it is not difficult to estimate. Others have to make a best guess. At this point, the stakeholder and the team can work out what they can do in a release. With the information of estimates and velocity and priorities, they roughly know what is in the different sprints.

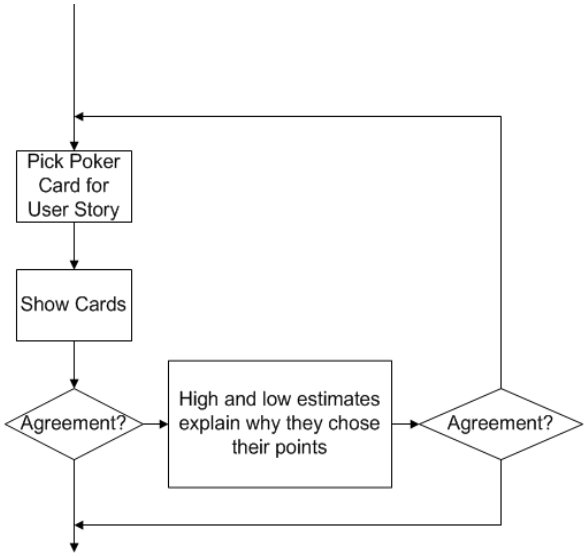


Figure 8 Estimation as an Iterative Process

Before the beginning of the first sprint, most participants report there is a sprint planning session (the last step of Figure 6, before the story cards are complete). This meeting will be repeated during the wrap week throughout the release, to prepare for the next sprint. At this point, the only information outstanding is the acceptance criteria. Like much of the information throughout the process, some acceptance criteria may have been noted on the story card. However, as some interviewees note, developing acceptance criteria too early may lead to premature "solution mode." Solution mode focuses not on what to develop but how to develop, which is inappropriate.

At the sprint planning session, the stakeholder is responsible for the acceptance criteria. However, most interviewees note the importance of the tester in the acceptance criteria process. The tester helps the stakeholder define the acceptance criteria in testable ways. The rest of the software development team helps by alerting the tester and stakeholder of specific technical issues that may also need to be tested. In theory, acceptance criteria go on the back of the story card. However, most interviewees state that they are entered into Jira instead.

Besides adding the acceptance criteria, most interviewees view the sprint planning session as a final chance to make adjustments before the sprint. The team may more "finely tune" the user stories. If necessary, the team and the stakeholder once more break user stories into smaller stories. Some teams create separate tasks for legacy and

Java developers. They may make a final re-estimation, and if any further dependencies emerge, the team may ask the stakeholder to reprioritise.

At this point, the project has been developed using the minimum information needed at any one point. After this first sprint planning session, the stakeholders and the software development team understand the release; one participant claims they know the first two sprints fairly well, and they are ready to implement at least some of the stories for the first sprint.

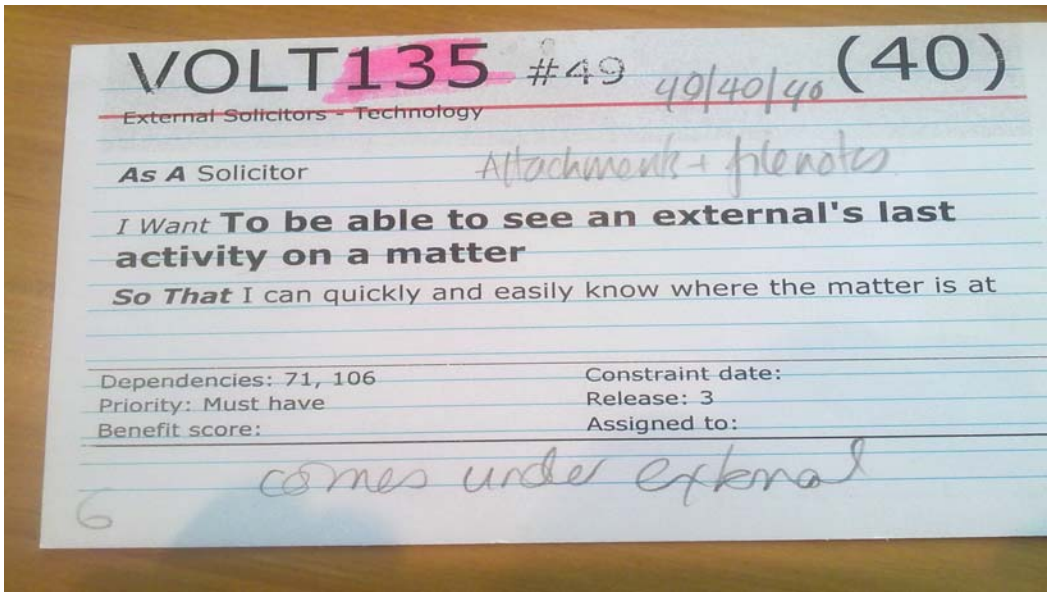


Figure 9 Front of Example Story Card

Shown in Figure 9 above is story card printed out of Jira. It shows the JIRA number (VOLT 135), standard format, the "Must Have" priority, the release number and a note of the dependencies identified. There are also the pencil annotations on the card. This shows the addition information 40/40/40 which shows the breakdown of the effort, the priority within the release, and a note to check the attachments and file notes on the AT wiki.

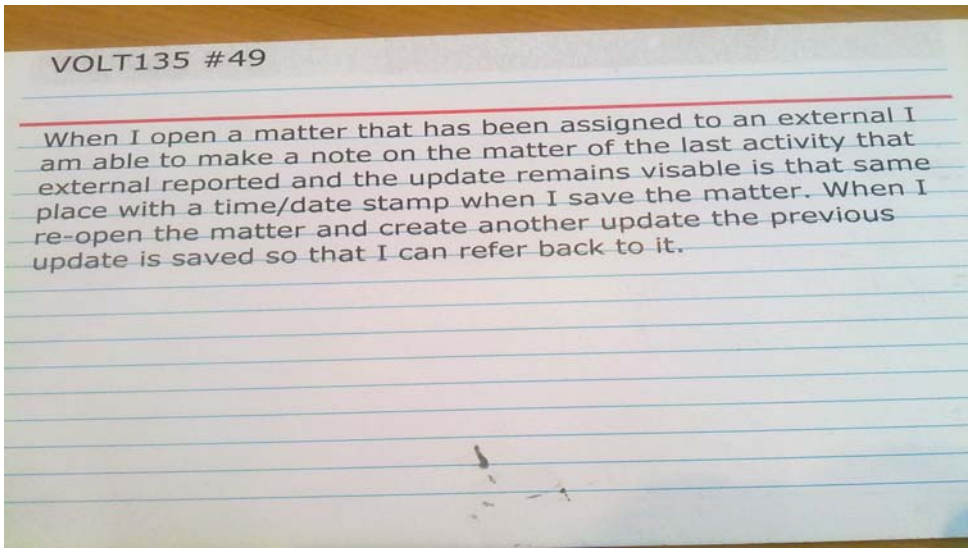


Figure 10 Back of Example Story Card

As can be seen in Figure 10, the acceptance criteria are captured on the back in a narrative form. From this natural language, the tester can make sure that update is visible and date/time stamped on update, and information is updated to the database.

4.2.3 Delivery Phase: Implementing User Stories

As described by interviewees, the Delivery Phase consists of a series of sprints that include planning, development, testing, demonstration and retrospectives. It starts with the first sprint, in which all the selected story cards are placed in the sprint backlog. The Delivery Phase consists of a release that delivers significant functionality. AT uses aspects of the Scrum methodology, using the term “sprints” for iterations. Other Scrum techniques used by the team includes stand-up (or scrum) meetings.

At the end of a sprint the user stories have been transformed into code. Many of the user stories may no longer be separately identifiable in the application. They become aggregated into a higher level application feature. At this point there is little use for the story card and this is taken as the end of the user story process for the purposes of this thesis.

4.2.3.1 Story Wall

As the sprint begins, the story cards are placed on the story wall. This story wall is located near the collocated software development team and should be visible from the team area.

Before the beginning of the sprints, the team decides on how they will format the story wall. This format essentially entails rules that provide meaning to the story cards on the wall. There are some components of the wall that all AT story walls have. For example, there is an order in which the work is performed: Analyst, followed by Developer, followed by Tester.

By creating a story wall with these "phases," the story wall forms a time series in which the goal is to move the story cards from the backlog on the left to the "Complete" (or the AT in-house designation, "Done Done Done") column.

Back Log	Analysis			Development			Testing			Complete
	Waiting	In Progress	Done	Waiting	In Progress	Done	Waiting	In Progress	Done	

Figure 11 Three-Stage Story Wall

However, the participants report differences in how these phases are divided. teams break each of the phases into three sub-phases: "Waiting," "In Progress"

"Done" columns as illustrated in

Back Log	Analysis			Development			Testing			Complete
	Waiting	In Progress	Done	Waiting	In Progress	Done	Waiting	In Progress	Done	

Figure 11. Others report only "Waiting" and "In Progress" sub-phases on their walls, as illustrated in Figure 12.

The difference between these two kinds of walls is that at the point when a team member completes a story card in

Back Log	Analysis			Development			Testing			Complete
	Waiting	In Progress	Done	Waiting	In Progress	Done	Waiting	In Progress	Done	

Figure 11, he or she moves the story card to the "Done" category for that phase. Presumably, a person responsible for the next phase will take the story card out of the "Done" category and place it into the next phase's "Waiting" category. However, teams that have only two sub-phases per phase move a completed story card into the "Waiting" section of the next phase. Effectively there is very little difference. However, it may be that for bigger teams, the number of cards is so great that these buffer sub-phases make a better representation of the sprint.

Back Log	Analysis		Development		Testing		Complete
	Waiting	In progress	Waiting	In Progress	Waiting	In Progress	

Figure 12 Two-Stage Story Wall

Once each team member starts a user story, they move the story card from the "Waiting" sub-phase and move it into "In Progress". While they may do this at any time during the day, most often they update the story wall during the stand-up meeting.

Figures 13 to 16 show photographs of examples of four story walls in use, illustrating the columns and other functional areas.

In addition to the area for stories, Figure 13 shows project artefacts such as the Success Sliders, the Focusing Question, the Social Contract and Team Achievements. One interviewee suggests that having these so visible helped to keep them front-of-mind and influenced some project decisions.

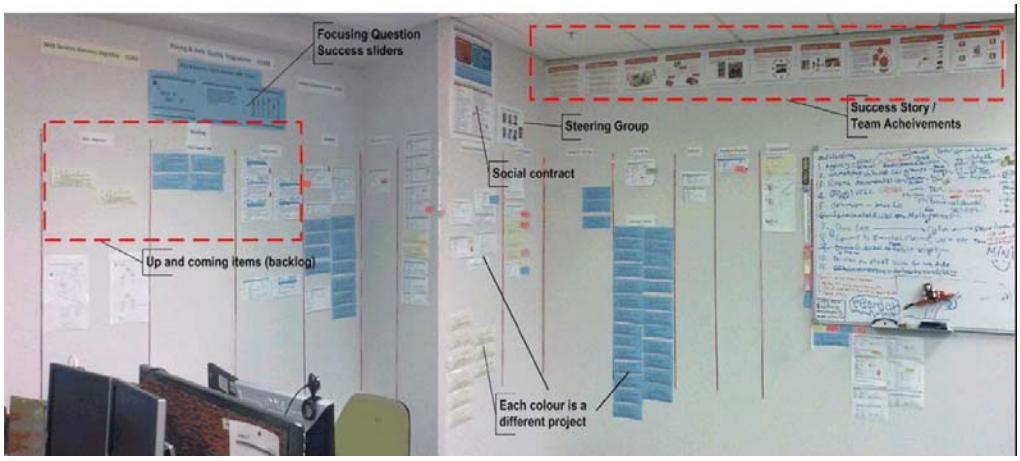


Figure 13 Story Wall 1

The story wall in Figure 14 is notable for the highly coloured story cards used to distinguish the developers assigned to each story. There are sticky labels on the current sprint.

Figure 14 Story Wall 2

The columns “Backlog,” “Analysis,” “Development,” “Testing” and “Complete” are used in the story wall shown in Figure 15 This illustrates the typical columns of a story wall.

Figure 15 Story Wall 3

Figure 16 Story Wall 4

Notably, there are calendars on the right hand side of the story wall shown in Figure 16. This shows that a team may use the Wall to “radiate” the kinds of information they find important.

The business analyst will move the user stories that are ready and place them in the "Waiting" sub-phase. By default, user stories in the "Waiting" sub-phase for Development are assigned to the technical lead, who then assigns the stories to the appropriate developers. The assignments are made in the Jira application and often also on the wall. Inevitably, issues raised in the sprint planning session may require a certain amount of business analyst work. These story cards remain in the "In Analysis" column for the business analyst to work on.

The user stories held in Jira must be updated by status changes to reflect those of the cards on the story wall. Some participants pinpoint this necessity as a problem. One participant spends half an hour after each stand-up session updating the JIRAs. Another points out that that the story wall is more likely to be accurate than the JIRAs; others believe the developers are more likely to update the JIRAs than the story wall and thus the JIRAs are more accurate.

4.2.3.2 Sprint ahead

As mentioned before, while the software developers and testers are working on the current sprint, the business analysts prepare the user stories for the next sprint. One business analyst describes how difficult it is to "get ahead" at the beginning of a release as they have to spend a lot of time helping the software development team during the first sprint. However, by the third or fourth sprint, the business analysts are usually far enough ahead of the developers that they can simultaneously support the sprint and do their own work.

4.2.3.3 Stand-Up Meetings

AT adopts the Scrum stand-up meeting. Every morning the software development team and key stakeholders meet in front of the story wall and discuss the current status of their work. Most participants report that the key stakeholders are involved in these meetings. One of the project managers insists the project owner be present.

The meetings are brief, lasting fifteen minutes to half an hour, perhaps longer. The point of the meeting is to understand the status of the sprint. One developer describes the conversation as being about "yesterday, today and tomorrow". Every team member

describes what they did the day before, what they plan to do that day and what they plan to do the next day.

While at the stand-up meeting, attendees often look at, talk about, touch, move, and update the story cards on the wall. Project managers can monitor the movement of cards through the phases and sub-phases particularly looking for cards that have not moved.

The stand-up meeting is also an opportunity to ask questions and raise issues. If an issue arises that requires further discussion, the people involved in the issue discuss it after the stand-up meeting ends.

4.2.3.4 Wrap Week

One participant states that the AT standard length for sprints is three weeks and other participants concur.

At the end of the sprint there is a wrap week. During this week, there is the showcase, at which the stakeholders are shown the results of the sprint. While the story cards are available, they are seldom consulted because the stakeholders focus on the functionality delivered, which may involve several user stories. The feedback the stakeholders give, however, may result in further user stories. However, the stakeholder tests and signs off the story if it has met the acceptance criteria.

In addition, there is the retrospective, at which the team examines if there are ways they can improve their sprints. Lastly, there is the sprint planning session, in which the user stories for the next sprint are elaborated, prioritised and story-pointed.

4.2.4 Summary

In this chapter the user story process is described in detail, based on an analysis of interview data. The main activities are identified, as are the different roles involved in them. In line with descriptions in the literature, the main activities at AT are user story creation in story workshops, prioritisation of user stories, estimation of user story effort, elaboration of user stories with key stakeholders, and the use of the story wall for planning and monitoring. Although some variations are reported in the implementation of these activities in practice, there is an overall consistency in and acceptance of the

Scrum and Agile techniques. Both the duplication of user stories electronically in Jira and the linking of these to the domain information on the wiki are also described; neither of these are “standard” agile practices.

This detailed description of the AT user story process is the primary input to the analysis using the DiCoT framework, as depicted in Figure 4, resulting in the Distributed Cognition analysis presented in the next section.

4.3 Distributed Cognition for Teamwork Analysis

This thesis takes a Distributed Cognition view of AT’s user story process as a theoretical perspective to understand the organisation of the distributed functional system and the dynamics of the representations and related reasoning in this system. The first part of this section uses the DiCoT framework to develop a Distributed Cognition model of the detailed description of the user story process given in Section 4.2. In the section following that one, the details of the cognitive activities are inferred from the data. Lastly, consideration is given to benefits and challenges uncovered by the DiCoT analysis.

The next Sections, 4.3.1 to 4.3.3, build the information flow, physical and artefact models of the DiCoT framework. In each model, phenomena significant to the user story process are identified.

The information flow theme tracks the significant flows of information and transformations through the representational states of the story card as well as the propagation of representational states through other media.

The artefact theme captures the uses of the artefacts – emphasising the story card and the story wall - throughout the user story process, to build an artefact model.

The physical theme explores the impact of the physical environment on cognitive activities related to the user story process.

Each theme continues with an analysis of the relevant DiCoT principles, and a section on issues uncovered from the data.

4.3.1 Information Flow Theme of AT User Story Process

The information flow theme begins with a discussion of the user story. Examining the definition in Section 2.3.2.2, one can see the user story as the primary "information hub" of the user story process. The information from different stakeholders and the software development team converge around the story card. Through the process of collaboration, decisions are made of the optimal representation of the information and the user story is continually defined and redefined by these flows.

The focus of this theme is on the flow of information through the user story process. This process consists of several formal steps resulting in intermediary representational states. These representational states are information flows themselves in that each state is a further redefinition of the user story. They are also the product of information flows, of collaborating views and information from various domains and perspectives.

The Information Model focuses on these representational states of the story card:

- Early story card – When the user story is created and written as a story card;
- Roughly prioritised story cards – The story cards are divided into the four MoSCoW categories;
- Prioritised story cards – Each story card is assigned a priority;
- Estimated story cards – The development of a detailed estimate of effort;
- Acceptance criteria on story cards – The stakeholder and software development team clarify how to know when the story is done.

As noted before, these are formal steps of the user story process. However, there are informal steps that need to be described, which is why this model begins with a discussion of the user story and how informal processes, namely the need to split a story, change the information flow.

Lastly, the way information flows through the story wall is considered.

4.3.1.1 User Story

The AT user story lifecycle is examined in detail in Section 4.2. The user story is the information that flows through this distributed cognitive system. One representation lives in the conversations and memories of the stakeholders and members of the software development team. It is also propagated to a different medium in a physical story card as well as to JIRA electronic story card. In many ways it functions as an information hub itself.

The information held on a story card and JIRA represents a user story in the abstract. Specifically, the components of a user story on a user story card representation are:

- JIRA number for cross-reference;
- The requirement couched in the standard user story format;
- The priority of the user story;
- An estimate, in story points, of the effort the software development team is expecting to expend in order to implement the user story. In addition, this may be broken down on a role by role basis;
- The acceptance criteria to evaluate when the user story is done.

Other than the JIRA number, each datum is developed through separate processes.

Overview of Information Flow

The user story goes through a series of state transformations. When each datum is added to the story card it becomes a different representational state. Once all data is on the story card, the user story is ready to be implemented.

There are two kinds of transformations the user story makes:

- Transformations due to the user story process;
- The kinds of transformation that occur when someone perceives something wrong.

The transformations due to the user story process

These transformations are both iterative, in that the previously represented information may be re-represented, and incremental in that representations are transformed by constraints (priority, estimate, acceptance criteria) that will also be represented on the story card.

These transformations include:

- The creation of the user story during the stakeholder workshop;
- The prioritisation of story cards into MoSCoW categories;
- The priority of a user story (written on a story card) which constrain the user story;
- Estimates for the user story (with possible breakdown by role) which constrain the user story;
- Acceptance criteria which constrain the user story.

These transformations will be explicitly detailed in the following Sections 4.3.1.2 to 4.3.1.6.

Transformations when someone perceives something wrong

At any time during the user story process, the stakeholders or the software development team can change the user story. One of the most common changes is to split the user story, as described in Section 4.2.2.1. There are several reasons to split the user story:

- The user story is too large to complete in a sprint;
- To ease planning;
- To keep functionality from be “blocked”.

Splitting a story is a comprehensive transformation and will require the other representations (the constraints – priority, estimate and acceptance criteria) to be readdressed if they have already been developed.

In this way, the AT user story process lacks a formal “Split a story” step that Beck lists in his Exploratory stage (Section 2.1.1.1). However, what the AT process recognises is that the need to split a story may arise at many different stages, and they embrace that fact.

These kinds of transformation are outlined below in the “Issues” subsection.

Analysis

Information Movement

The user story in all its incarnations is a way of moving information. In the beginning, information moves from what a stakeholder would like to see to the index card. From that point, the process draws information from sources such as stakeholders and the software development team through each of the representational states of the story card to the implementation of the user story.

Information Transformation

As the user story is developed, each of the additional types of information transforms the user story from one representational state to another.

Information Hub

The user story is effectively an information hub, being the central locus of the requirement.

Issues

Reasons to transform (or split) the user story:

1) “Epics” or “master stories” are user stories that are too big to complete in one sprint.

Often it is obvious to the analyst that the user stories generated by the stakeholder workshops are too big, as described in Section 4.2.2. This process may occur many times throughout the lifecycle of a user story. The business analyst may initially work through the epics. The following quote from a business analyst illustrates the kind of thinking that takes place:

> *What I’m really trying to get at the first meeting is just to get as much information out as I can then I’ll follow up on and some of those, umm, some of those cards will become ten new stories and that so I also do tend to label the ones that I feel are what I call epics… those are the ones that are seriously at a hugely high level and I need to break them down because they*

> *may, you know, like... we've got some of them that are, that use forty stories comes out of one.* ***Business Analyst***

Developers may perceive the need for splitting from their technical background.

2) To ease planning, user stories must be small enough to show progress on the story wall throughout the sprint, as detailed in Section 4.2.2.1. The representation must be of the correct granularity to show movement across the wall. It is the movement of a card that indicates work is done.

3) Parts of a user story may needlessly "block" or impede the progress of other parts.

These parts could be developed or tested separately to more accurately show the current state of the sprint on the story wall. For example, if a screen needs to be changed and there are several changes to make to a screen, parts of the screen functionality may be more difficult to develop. The bundling of the screen functionality together means the tester cannot proceed with the parts of the story that may be ready for testing.

4.3.1.2 Early Story Card

As described in Section 4.2.2, the output of the stakeholder workshop is the first representational state of a story card.

The medium is an index card or Post-it note. One participant calls this first representation the "framework of the card" to which the stakeholders and members of the software development team "add all the fat".

The representation is the written word in natural language. The information and knowledge required to create the story card involves domain knowledge (sometimes referred to as "role") and an understanding of business value for Company A

Flow of Information

As described in the AT User Story Lifecycle in 4.2.2, a user story usually begins in a stakeholder workshop in which stakeholders brainstorm user stories. Figure 17 illustrates the movement through the intermediate representations to the first representational state.

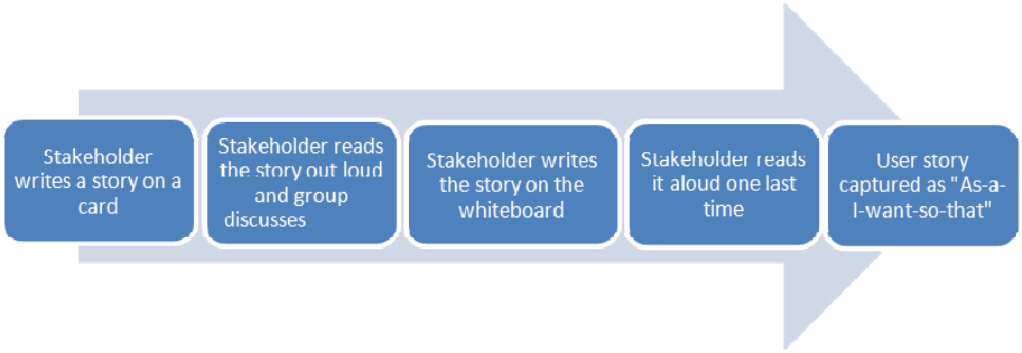


Figure 17 Creation of the User Story

The stakeholder writes the user story down on a story card. The stakeholder uses natural language to write the user story. This first sentence on the story card is a movement of an internal representation in the stakeholder's mind to an external one in written language, possibly in the standard format. If it has been written in the standard format, the representation has also been transformed.

The stakeholder reads the user story aloud. The information moves into the public realm. It is also transformed from written language to a spoken representation. Both should be in natural language, for ease of interpretation by others. The information moves throughout the group of stakeholders

Once the discussion is finished, the original stakeholder often writes the user story on a whiteboard or a flipchart. Alternatively, the stakeholder may place the story card on a wall. The wall, whiteboard or flipchart acts as an information hub, as user stories will accumulate. Thus, the internal representation in the stakeholders mind goes through several transformations, primarily to verbal and written language representations.

Lastly, after the workshop is over, the business analyst will use the original cards or perhaps re-write them on new index cards. The representational state is the user story on a story card: either the original story card written by the stakeholder or a version by the facilitator.

Analysis

Information transformation

Information is transformed from an idea in someone's mind to written language, to spoken language, back to written language.

Information Movement

Information moves from the mind of the stakeholder responsible for the user story to a piece of paper, to a spoken representation, to a publically accessible written representation, to a final version of the first representational state of a story card. The stakeholder writes and speaks. The facilitator may capture the user story on a final story card, or use the original story card written by the stakeholder.

Information Hub/Buffering

A whiteboard or flip chart functions as an information hub, as is a wall on which story cards are attached. They store user stories while the stakeholder workshop continues, but keep the story cards accessible.

Communication Bandwidth

Face-to-face communication with a wide variety of roles aids communication and shared understanding. This broadband communication enables a high degree of error checking by making it easier to understand the intent of the spoken communication.

Issues

One interviewee reports a difficulty when dealing with stakeholders who "talk in circles". As the expression indicates, information that appears to move does not actually do so. Essentially, these stakeholders are not trying to share their understanding.

This interviewee suggests these stakeholders hope that by being obscure, no one else will argue against their stories. Regardless, the interviewee needs to understand what the user stories are. This interviewee resolves this problem by audio recording these sessions.

4.3.1.3 Roughly Prioritised Story Cards

This process is described in Section 4.2.2. Prioritisation is a two-step process. This step is the second representational state of a story card, but the first step of prioritisation.

The medium is a collection of story cards which represents one of the MoSCoW categories.

Flow of Information

The flow of information is from an unordered line of story cards spread down the middle of the table top (Figure 18) to four stacks of cards representing each of the MoSCoW categories. The stakeholders move the cards to the various corners representing the categories. The representational states are four piles of story cards corresponding to the MoSCoW categories.

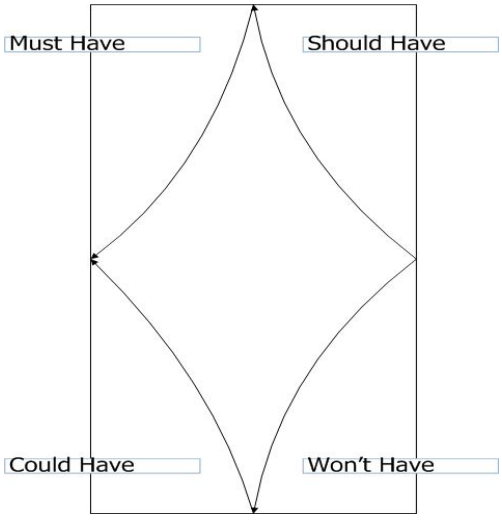


Figure 18 MoSCoW Table Top

While there may be issues about where a card belongs, this step is not as contentious as the prioritisation within a category is. Most interviewees report that it is more inclusive than exclusive, i.e. the number of the highest category "Must Have" is not limited.

Analysis

Information Movement

The stakeholders discuss the relative merits of the user stories, sharing domain information and their perspectives on business values. They pick up cards and move them to corners. As this process is not controlled, different stakeholders may move the cards many times. Perhaps after considering the values of other story cards relative to other cards, a stakeholder may move a card into another corner.

Information Transformation

The story card is transformed into a placeholder in a series.

Communication Bandwidth

Face-to-face communication between the stakeholders with a wide variety of roles aids shared understanding and perspectives. This communication bandwidth also allows all to be aware of the movement of cards. Thus, the movement of cards themselves is an information movement evident by the nature of the process.

Issues

Often, this MoSCoW prioritisation process does not split the user stories into four significant piles. Stakeholders may place most of the user stories into the "Must Have" stack. If so, this step will not have winnowed many user stories out. Not only will the next step probably be more contentious, the cognitive value of the current step is reduced.

4.3.1.4 Prioritised Story Card

As described in Section 4.2.2, this stage is the third representational state of a story card. It is the second of two prioritisation steps.

Flow of Information

The flow of information is from an unordered line of story cards spread down the middle of the table top representing the story cards of the "Must Have" category to a priority written on the story card.

Stakeholders determine the priority (or place in the line) of each story card in turn. Figure 19 breaks down the process in detail. It shows how the process is simplified into a series of binary decisions.

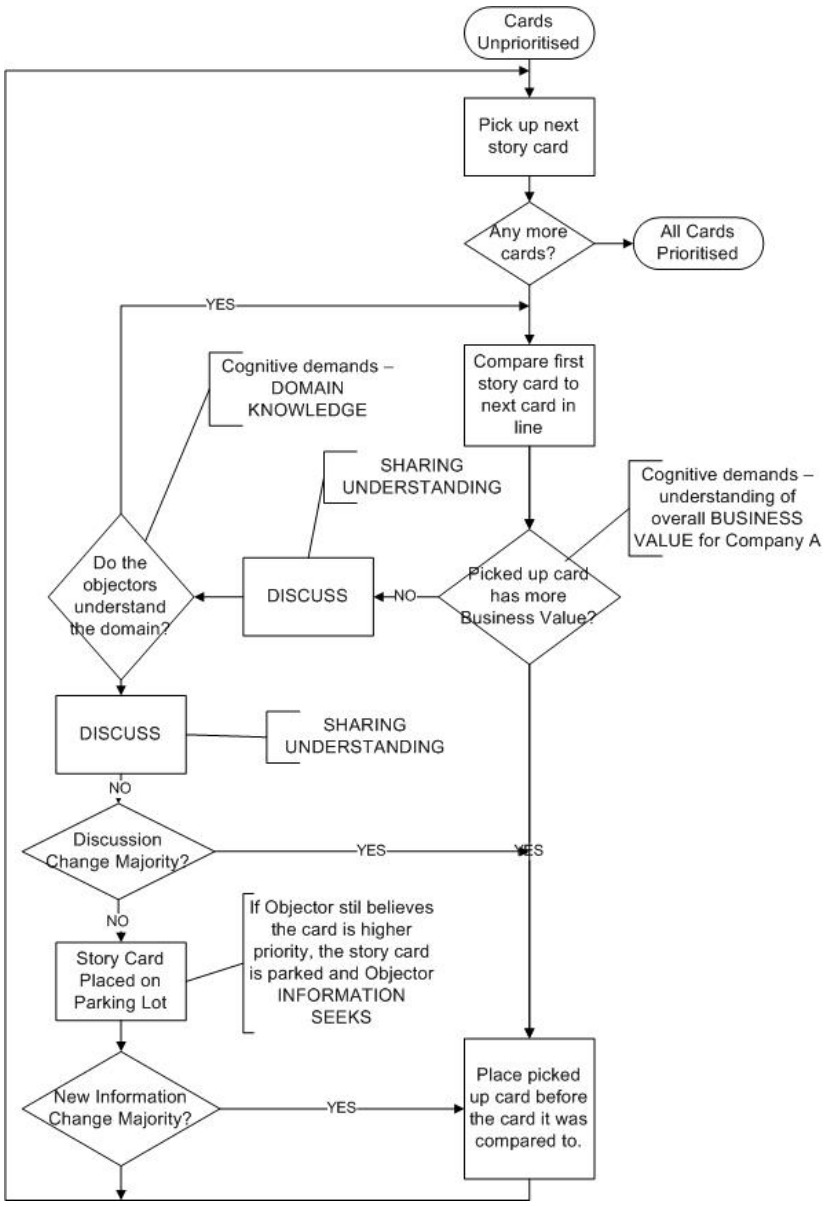


Figure 19 Cognitive Map of Prioritisation

Final representation: The position in the pile – the story card's priority – added to the story card. The information necessary for the creation of priority involves domain knowledge and an understanding of business value for Company A.

Picking up the first story card at the top of the cards spread in a line on the table top, the facilitator holds the card next to the next card in the line and asks if the held card is of higher priority than the card on the table, to which the stakeholders reply “Yes” or “No”.

If there is significant disagreement, the stakeholders discuss the relative value of the cards. Sometimes the problem is that those who disagree about the comparative value of the story card do not understand the domain in which this particular card is value. If they do not, those domain experts who value the card explain the relevance of this particular card to the domain. They share understanding of the domain.

If there is no consensus, the story card may be “parked” on the Parking Lot. As described in the User Story Lifecycle Section 4.2, the Parking Lot is a buffer that allows the dissenting stakeholder the opportunity to find information to prove his or her point. Thus the dissenting stakeholder seeks information to support his or her case and brings it forward, most probably at a later date. In the meantime, the held card is cycled down to the next card in the list. When the new information is presented, if it is not persuasive, the card remains where it eventually was placed.

Therefore, both domain knowledge and the understanding of business value are required to divide the story cards into categories. While most of the stakeholders should understand business value, individuals probably do not understand all domains. At this point, two cognitive activities support the prioritisation process – information seeking and sharing understanding.

An alternative method of prioritisation

A facilitator asks the stakeholders to apply three differently coloured sticky labels that corresponded to different priorities on a list of user stories held on a flipchart.

Analysis

Information movement

Domain information is brought forward by the process. Instead of creating a top-down definition of priority for each domain, information is brought forward when necessary.

In the case of a dissident stakeholder finding support for his or her argument, information literally moves.

Information Transformation

The user story written on the story card is transformed into a prioritised user story. In addition, the discussion and sharing understanding especially of domain knowledge may transform some stakeholders' representation of business value.

Buffering

The table top serves as a short term buffer in which those story cards not in consideration at the moment are held ready but momentarily ignored. This buffer allows the focus of the stakeholders to narrow to only the two story cards in discussion.

The Parking Lot serves as an intermediate term buffer in which user stories may be reconsidered.

Communication Bandwidth

Communication is face-to-face. One has the opportunity to take in the complete bandwidth transmitted by a stakeholder – the facial expression, tone, and body language. This helps the communication remain robust.

Behavioural Trigger Factors

Placing a story card on to the Parking Lot triggers the concerned stakeholder to find information to make his or her case.

In the alternative case, the *perceptual* and *naturalness* principles apply – "you can kinda see it building up". The user stories judged as highest priority have a larger number of sticky labels of the colour so designated against them. It becomes a perceptual judgement. The fact that is the largest number is complies with the naturalness principle. This "build up" is illustrated in Figure 20 on the next page.

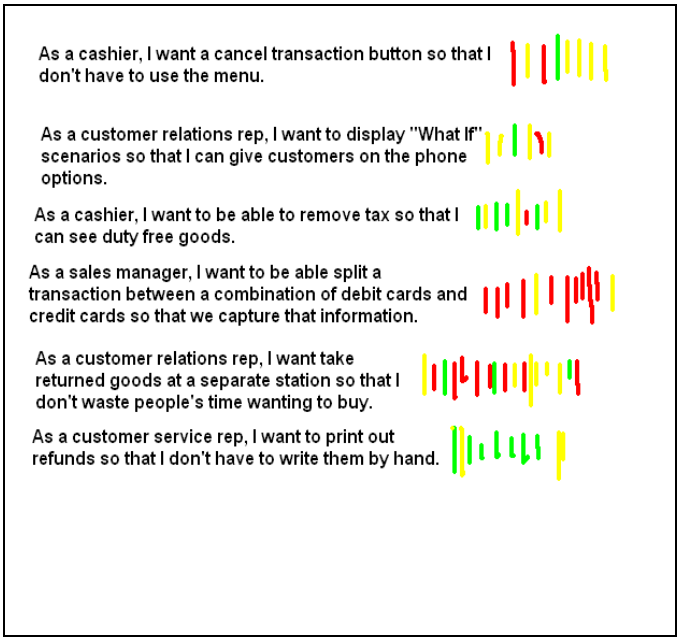


Figure 20 Priority "Building Up"

This is not unlike the visual evidence gathered by Sir John Snow when he looked at a map of the cholera outbreak in London in 1854 (Tufte, 1997, pp. 28-29). Snow saw a concentration of cases in an area and by investigating, discovered a contaminated water pump.

Issues

This prioritisation process can spark many arguments. One participant asserts the need for "really strong facilitation skills" to handle this second stage of potentially contentious prioritisation. Participants often find that the arguments may occur when one stakeholder does not understand a user story. Often stakeholders oppose user stories because they see them as a threat to "their" user stories.

A number of the participants use the "parking lot" to put these issues aside if it appears there will be no resolution. Stakeholders are asked to bring information to prove their point on the relative priority. This technique allows the process to proceed.

4.3.1.5 Estimated story card

The process of story card estimation is described in Section 4.2.2.1. The output of this process is the fourth representational state of a story card.

Flow of Information

This representation consists of an estimate and possibility a matrix of roles and estimated time for the role. For a single user story, members of the software development team select a story poker card representing effort.

The team discusses the various estimates on the poker cards. The team may or may not have developed a shared understanding of the story pointing. If they have, they may quickly estimate. If they have not, they may formally create a lowest common denominator by deciding which user story will take the least effort. They use that as the lowest measure and estimate all others in terms of that user story.

Information necessary for estimates includes software development experience and technical knowledge.

Analysis

Communication Bandwidth

The story poker games manipulates the communication bandwidth by holding back the movement of information – what everyone thinks is the estimate – until a moment of the "reveal". By making it into a game, personalities can neither dominate nor disappear. Choosing, hiding and then presenting the cards require the entire team to "show" an opinion. As Grenning (2002) says, "They all have to be a turtle and stick their neck out when they play their card" (p.1).

Information Transformation

The estimates are transformed into a number from the Fibonacci sequence. A sounder understanding of the meaning of the points emerges over time.

Issues

One participant describes one team who consistently estimated high while another one always estimated low. Once these tendencies came to the attention of AT management, they sent members of other teams to discuss why they estimate the way they do, but despite these discussions, both teams still over- and under-estimated. In the end, AT resolved the problem by integrating members of these teams into other teams.

Another issue is the differing demands on members of the team. Legacy development has impact on the amount of work the tester must do. For that reason, one participant prefers to put a breakdown of effort by roles on the cards.

4.3.1.6 Acceptance criteria placed on story card

As described in Section 4.2.2.1, adding the acceptance criteria to the story card (or onto JIRA) completes the user story. Therefore, the story card with the requirement in the standard format, a priority, an estimate and acceptance criteria is the final representation of the story card. While it may complete the information needed for the user story to be developed, the team and stakeholders may still change the story.

Information necessary to develop acceptance criteria includes domain knowledge, knowledge of business value for Company A, and technical expertise, particularly in regard to testing.

Flow of Information

Stakeholders specify acceptance criteria based on “so that” of the user story, as one participant puts it. It is the “key driver” of the acceptance criteria,

The software development team helps develop the acceptance criteria into something they can use to create automated tests. Often what they define as the acceptance criteria is high level and may need to be defined in a more atomic, slightly more technical way without losing the stakeholder’s intent.

It is critical the tester understands the criteria because the tester creates the test scripts for the acceptance testing at the showcase.

Analysis

Information Movement

The reasoning behind the “so that” moves from the stakeholder to the software development team.

Information Transformation

When the stakeholder describes what he or she wants, the software development team may need to define a more generic functionality that can provide the stakeholder with what they want, and allow it to be changed. For example, the stakeholder may want a specific colour. The software development team is likely to develop the software to handle all colours, but default the colour to what the stakeholder wants. The software development team and especially the tester may reframe the acceptance criteria in terms with which they can create tests.

Issues

Sometimes stakeholders try to use the acceptance criteria to impose a design solution. In this kind of case, the software development team need to work the stakeholder back to the business value.

4.3.1.7 Story Wall

The story wall process is described in Section 4.2.3. In this section, the information flow through the story wall will be discussed. The discussion of the story wall as an artefact is in Section 4.3.2.1.

Flow of Information

Story cards on the story wall signify something more than what is written on them. As discussed in Section 4.3.1.1, the effort the software development team has spent making sure the stories are “trackable” results in cards that can be “encoded” as a job corresponding to this “trackable” amount of time. Participants perceive this amount of time to be 2 or 3 days.

Therefore, as a result of AT rules, the story card has been transformed from a thing-in-itself to be part of the wall.

As shown in Figure 21, once the sprint begins, the story cards are placed in the backlog column. Each card must move from the backlog to the final category, termed “Complete”. The movement of information along the wall represents the movement of responsibility and work.

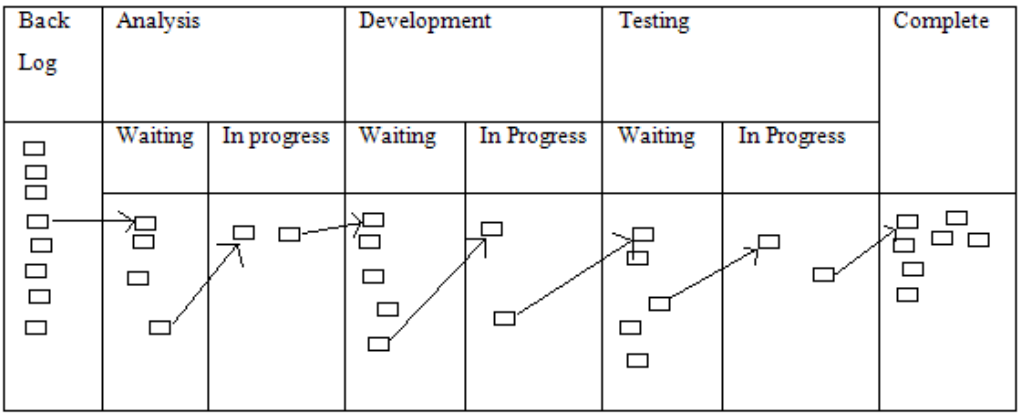


Figure 21 Flow of Information on the Story Wall

Analysis

Information Movement

When a developer moves a story card from "In Progress" to "Waiting" of another phase, for instance, the information moving is different in some respects from the "overt" information movement in a stakeholder workshop or in a prioritisation meeting. Rather, the move itself – the act of moving the card from one column to another - is information. The status of the story card and as a consequence the sprint has changed. The story wall "radiates" this change, as Alistair Cockburn (2007) points out (referred to in Section 2.1.1.1).

Information Hub

Through the daily stand-up meeting, the story wall becomes the information hub with several channels of information – project leader, stakeholder, developer, tester and business analyst – meeting to discuss the project.

Buffering

While the backlog does function as an information buffer, the movement in and out of the buffer is not like the usual information buffer. Information buffers are used to offload information when the inflow is faster than it can be processed. All sprint cards go into the sprint backlog at once and then get taken out in piecemeal fashion. While

there is not an incoming stream of information, the buffer still functions as allowing the user stories to be dealt with in turn.

Behavioural Trigger Factors

While most participants report that the developers and testers primarily use the JIRA project tracking application to understand when the status of user stories has changed, others note that the arrival of a story card in the "Waiting" category for a member of the team triggers an action.

Informal Communication

Several participants emphasize the value of the story wall for advertising the sprint status to stakeholders walking by. This value is that the sprint state is readily apparent for those interested to see. There are stakeholders who come to the stand-up meeting, and there are stakeholders who do not. Even the stakeholders who do not come to the meeting can perceive the sprint state. Some interviewees mention the stakeholders may ask questions based on what they see.

Issues

Not every participant embraces the story wall unreservedly.

> *To me, it's a lot of overhead, I personally don't, it looks good, it makes management happy, that's the way I see the visual board. We could certainly handle it all electronically, we wouldn't ever have to do, to get up and stand together.* ***Business Analyst***

This is a minority view.

4.3.2 Physical Theme of the AT User Story Process

As first described in literature review Section 2.3.2.3, this theme focuses the physical context of the cognitive system. It examines what can be "physically heard, seen and accessed by individuals" that have "a direct impact on their cognitive space" and how they "shape, empower and limit" the cognitive activities in the AT user story process (Furniss, 2004, p. 30).

The AT physical theme consists of several components:

- Story wall – Viewed as part of the environment;
- Team area – Where the team works;
- Large meeting rooms – The kind of room (with its furnishings and equipment) that can accommodate the larger numbers that may be involved in a stakeholder workshop or a showcase.

4.3.2.1 Story Wall as part of the physical environment

The story wall is both an artefact and part of the physical environment. The analysis in this section focuses on its support of cognitive activities as part of the physical environment.

The photographs in Figure 13-16 illustrate a range of AT team story walls.

Use of physical environment

The story wall displays this information:

- Sprint status shown by balance of cards between the backlog and the completed column. If there are too many cards still in the backlog, there may be a problem;
- The work load balance between different phases/roles of the sprint. If the cards are accumulating in one phase, there may be a problem;
- To whom the story cards are assigned;
- If looking at the story wall over several days, an unmoving story card may signify a problem.

Analysis

Space and Cognition

In their team area, AT teams use the physical wall of their environment as a medium for an important artefact, the story wall. As the information on the story wall is not filed or "put away," it is always visible and available, as shown in Figures 13 to 16 in Section 4.2.3.1. If the team has built the wall so that information is evident from afar, the team members can look at the wall and perceive information. Moreover, they will notice when someone moves a card from one column to another on the wall. Thus, by looking at the wall, one can see what the state of the sprint is. When one sees someone move a

card, one sees that the sprint state is updated. Thus, if a developer or a stakeholder seeks information on the sprint state, this cognitive activity is made easier by becoming a perceptual judgement.

Perceptual Principle

Another way some AT teams support information seeking is by coding visual cues into the story wall to represent information. For example, members of the team may adopt a coloured sticky label or an avatar. They place these visual cues on the story cards they have been assigned. In this way, they are taking responsibility for the cards. If one knows the code, one can perceive who is currently working on which card. An example of this kind of wall can be seen in Figure 14.

Another example of the perceptual principle is how one can perceive the status of a story card by its location on the story wall. The story card will be in a phase, indicating which role is responsible for it. It will be in a category indicating whether it is waiting or currently being developed.

These practices help the software development team and the stakeholders to perceive the state of the sprint from afar. Not only does it ease the amount of walking they may otherwise have to do, it changes what necessarily might be a computational cognitive task to a perceptual one. Even with the process of decoding the meaning of the labels and columns, this perceptual process is a lighter process than searching through the Jira application to determine what the state of the sprint is. If there were no sticky labels or avatars (such as in Figure 15 or Figure 16), a project manager must read the card to find out to whom it is assigned (unless there is only one developer, business analyst, and tester on the team). Again, the story wall supports the cognitive activity of information seeking.

Issues

One issue mentioned repeatedly during interviews is a lack of room for the story wall. Many participants report that the story wall may not be in the same area as the team area, or they do not have enough room to convey the information they would like. A lack of physical space makes sharing understanding through it more difficult.

In Figure 13, one can see a story wall built around a corner. However small the corner may be, it may limit the horizon of observation. At the same time, it is possible a team may use the corner in a meaningful way.

4.3.2.2 Team Area

The AT software development teams are collocated in open plan team areas.

While the team area may be collocated, this is not to imply the team members are always in the office. One project manager reports many members often work from home. Likewise, another participant reports that project managers and business analysts are often away from their desks.

Use in Physical Environment

The team work in the team area. In addition, some participants report the stakeholders spend a considerable amount of time in this area throughout the day.

Analysis

Horizon of Observation

Some participants report they can see the story wall from their desks. Such proximity means they can not only look at the sprint status at any time, but if they are at their desks, they also will notice if anyone moves a story card from one category to another, or one phase to another.

Situation Awareness

Teams can easily share cognition through informal means such as overhearing questions, discussions on how to solve problems. One participant confirms: “We work together in our pods, so we’re all talking. We all overhear the same things”.

This participant also states that that the SMEs and project owners “sit with us for a number of hours a day”. They also overhear the team talking about problems and can step in to solve them.

Issues

As noted above, several participants report a lack of room in their team area, especially for the story wall.

4.3.2.3 Large Meeting Rooms

For large stakeholder workshops or showcases, facilitators require large rooms.

Use of physical environment

Several participants discuss the need for large meeting rooms for stakeholder workshops and showcases, and their ideas about the suitability of the kind of rooms necessary for these kinds of meetings.

The use of physical space is important in these rooms to allow stakeholders to break into groups for the initial brainstorming session and then reassemble as a larger group to discuss them, as described in Section 4.2.2. Breaking into small groups helps the stakeholders develop their user stories, practice explaining to the others in the small groups, and hearing what others have to say. Thus, having the room to split into small groups supports cognitive activities such as communication, decision making and sharing understanding.

Several participants also believe an informal atmosphere is important for a stakeholder workshop. One project manager imagines her ideal rooms: "ones in which we can get rid of the tables and just have everyone sitting around in quite an informal, ah, format". This kind of room contrasts with a "classroom" kind of meeting room in which all chairs are facing in one direction. Likewise, it is different from sitting at a boardroom table.

An informal atmosphere signals to the stakeholders that they are to produce information not simply receive it. An information atmosphere also supports the cognitive processes underlying that first brainstorming step of the user story process. This physical environment supports communication between stakeholders, sharing of perspectives and understanding and helps them make decisions.

If the point of the meeting is prioritisation, several participants specify the need for a large board-room type of table, for the prioritisation activities described in Section 4.2.2.

Analysis

Situation Awareness

The participants need an environment in which everyone can not only see but hear each other clearly. This is the kind of environment that allows that "contextual banter and debate". Stakeholders can see and hear each other clearly, instead of hearing someone talk to a facilitator at the front of the room. Through this kind of interaction, the attendees are able to communicate and share understanding. Having being aware of the situation helps decision-making, in that it is obvious whether there are any other views. Being aware of this situation, stakeholders collaboratively build representations of the user stories through the stakeholder workshop.

Arrangement of Equipment

As seen above, several participants report that the arrangement of furniture is important for the group cognitive activities such as communication and sharing understanding.

Space and Cognition

During the prioritisation sessions described in Section 4.2.2, a large boardroom table affords the use of its surface, enabling the facilitator to use it to focus attention on the story cards. While this surface is important for dividing the story cards into MoSCoW categories, it is even more crucial for the prioritisation within a category. By implementing a rough sort algorithm, AT simplifies the task of sorting. Hutchins' colleague at the University of California San Diego, David Kirsh, (1995) points out, "the fewer degrees of freedom an agent has, the simpler the task". Through "intelligent use" of the table top, AT reduces each decision to a binary one. The stakeholders are focused on the task at hand and discouraged from thinking about "their" user stories, as some participants report the stakeholders have a tendency to do.

Issues

None of the participants report any problems with the large meeting rooms.

4.3.3 Artefact Theme of the AT User Story Process

The Artefact Theme examines how artefacts affect the performance of the cognitive system. In this theme the focus is on "artefacts, representations, and environmental affordances" (Furniss, 2004). This theme is developed at length in literature review Section 2.3.2.1.

In the AT context, the focus is on the ways information is stored, transformed and communicated by artefacts. AT's user story process is highly centred on the representations of the user story held on the story card and the story wall. Another significant artefact is the Jira electronic "story card" that corresponds to aspects of both the story card and the wall.

Artefacts identified in this cognitive system are:

- Story Cards;
- Story Wall;
- Graphical Representations;
- AT Cultural Artefacts;
- Sticky labels;
- JIRAs;
- The AT wiki.

The analysis explores what the artefacts are and how they are used in this functional system from a Distributed Cognition perspective. The artefacts covered in this analysis are those in which a clear indication of the physical aspects of the artefact and its use are evident from the data. In addition, although there may be other artefacts, only those uniquely involved in the user story process have been included.

4.3.3.1 Story Card

A story card is a physical manifestation of a user story.

Use of Artefact

The story card is the medium used to represent a user story through the life-cycle. Most participants do not distinguish between the user story and the story card.

The story card is physically used through several processes that change the representational state of the user story. Its tangible nature allows the stakeholders and team to manipulate the cards to affect these transformations. It also serves as a reminder of what the user story is.

The uses of story cards the participants describe include:

- Writing on a card;
- Reading a card;
- Using the story card to discuss the user story;
- Holding one up to talk about the user story;
- Trying to monopolise the conversation by taking the story card(s);
- Sorting the cards into a stack based on high-level feature, category, or domain;
- Sorting the cards into stacks based on priority of the user stories;
- Holding two cards next to each other in order to compare the user stories;
- Moving the cards along the story wall as their status changes.

Members of the software development team and stakeholders use this artefact to store, transform or communicate information. It is useful to note that some of these uses cannot be accomplished if the user story had been written in a software requirements specification.

In many ways, the use of the story card changes what normally would be a serial process – reading written language or listening to spoken language – into a two-dimensional one. When listening to other stakeholders talk about a user story, not only does a stakeholder have to attend to this ephemeral information flow, remember it, but also construct meaning by integrating it with what was said before. Written language is buffered by the paper, whiteboard or flipchart. Regardless, the input process is serial and one must mentally integrate the information on the fly. One must remember and compare information mentally.

Contrast this with the fluency with which the stakeholders can compare user stories on story cards. The cognitive load on memory is reduced: no longer does the stakeholder need to remember what the user story was because it is there on the card. The stakeholder does not have to mentally compare two user stories but look at the two cards and use the mental resources to evaluate the relative merits of the stories.

For example, a group of stakeholders cannot easily break a list of user stories held in a word processing document into four categories. Besides the difficulty of fitting around a single screen, the stakeholders must read the same requirements at the same time and turn the page once everyone is ready. If one has to wait, that stakeholder may lose focus.

The cards enable the stakeholders to focus on a single user story at a time. They enable a stakeholder to take as much time as he or she wants to think about the user story. The cards enable the stakeholders to focus on the task at hand, not constructing a supporting mental model to do so.

Analysis

Mediating Artefact

As reported in literature review Section 2.3.2.1, a mediating artefact is one brought into coordination to *complete* a task. As such, the story card is the central mediating artifact through which a series of representations of what the user wants is propagated through the cognitive system that is the user story process. It only loses relevance as the user story becomes working code.

One participant describes the mediation process.

> *Cards are good, cards are because people have written on them in handwriting, scribbled over their, they become almost a, not a living entity, but that's what they feel like to the stakeholders as well.* ***Business Analyst***

Another participant backs up this assertion of how the stakeholders regard the cards as more than paper. This participant reports how stakeholders try to "take over" a meeting by gathering up all the cards.

The nature of the story card as an artefact allows interesting uses. When sorting the cards into stacks, the stakeholders create a new temporary artefact – a stack sorted by domain or a stack sorted by priority. Thus the story cards mediate in the creation of new information represented by the stack.

One could argue teams could build these lists on a whiteboard. However, there is a tendency to "build downward," to define the headings of the lists first on a whiteboard. As they are working through a category, if the team decides a category should be subsumed into another category, any entries under the first heading must be copied to the new heading. The tangible nature of the cards allows the team members to build upward, mix and merge the piles until sensible categories remain. The categories emerge.

The story card also mediates between past and future for the completion of the task. It reminds the stakeholder or the team member of the conversations they have already had about the user story and also to conversations they need to have about it in the future.

Coordination of Resources

The story card can be used to coordinate the plan for the user story by allowing the team members of the software development team and stakeholders – the components of the cognitive system - to compare what they think the user story is with what is on the card. To some extent, the user story on the card is the definitive representation as the stakeholder wrote it in his or her own words.

Issues

Story cards may be covered with pen and pencil marks. They may become torn and crumpled. Handwriting may be difficult to read. Two participants express a fear of losing a story card.

While user stories may live in conversation, it is possible to forget what the conversation was: "you will scratch your head in front of the story wall," as one participant puts it. This problem will not occur, if, as another participant adds, the user stories are written so well it does not matter whether they can remember these conversations or not.

> *What we're aiming towards is it's written so well that you can pick it off, because if your BA and your tester isn't around, you can get cracking on it.*
> ***Project Manager***

One of the goals of writing a user story is to make its meaning clear.

4.3.3.2 Story Wall

One participant calls the story wall a "physical, visual map of what we're doing and where we're going". The focus of this analysis is on the wall-as-an-artefact and centres on its use. The software development team couples artefacts – the story cards – with a format to create the story wall, another artefact. The format instantiates the rules by which the cards must move across the wall. These rules effectively support the cognitive activity of information processing.

Use of Artefact

The story wall is an artefact built of other artefacts, most notably the story cards.

These are the uses of the story wall reported by participants:

- Put story cards into the sprint backlog at the beginning of the sprint;
- Move a card from one sub-phase to another on change of status (Waiting, In Progress, Done);
- Move card from one phase to another upon change of status (Analysis, Development, Testing, Complete);
- Move card back to a Waiting sub-phase of a previous phase;
- Look at the wall to understand what is happening in the sprint;
- Stand at the wall during scrum;
- Point to or touch cards while discussing a user story during the scrum.

This story wall as an artefact affords many uses not available if the requirements had been written in a document. The question is whether the affordances offered by the artefacts help the cognitive performance of this functional system.

Analysis

Representation-Goal Parity

A common way of describing the sprint is how the teams demonstrate sprint progress by movement of story cards through the phases and sub-phases on the wall: "…you can track across the storyboard", as one participant says.

One of the participants describes the point of a project manager's job as getting "every member of [my] team to be able to be successful, and to be successful, they get their cards across to [the "Complete" column]". The story wall coordinates the user story development (and indeed the sprint) by making it easy to compare the current state (where the story cards are on the wall) with the "Complete" goal state. It does this by representing chronologically a series of phases, each of which must be completed in order to achieve the goal state. The goal state is when all cards are in the "Complete" column. It is also obvious if any story cards have not reached the goal state.

The principle of representation-goal parity exemplified by the story wall supports the ability to understand the state of the sprint. Instead of having to make a cognitive computation, one can make a perceptual judgment about the state of the sprint.

How does a perceptual judgment contrast with a cognitive computation? Essentially, the way the story wall looks imparts information. Other than remembering the meaning of the columns – which one can see as they are labeled, aiding the cognitive task of recall – and the meaning of the colours used, if any, one does not need to decode representations into other representations in order to understand the sprint state.

As Hutchins (1995b) points out when discussing how pilots ease their cognitive load by marking points on a speed dial, "a memory or scale reading is transformed into a judgment of spatial adjacency" (p. 283). The pilots no longer "read" a speed dial. They no longer look at the speed represented by the dial, remember what that means, and contrast that with their memory of what the trigger point is. The trigger point is when the needle reaches the speed bug.

In the AT example, team members can look up the status of individual JIRAs to build an understanding of the sprint status. To do that, they must know how to use Jira, what the JIRA numbers are (or other identifiers), and perhaps how to generalize that information across a sprint to extract the sprint status. Using Jira requires a heavier cognitive load than merely looking up at the wall and seeing the sprint state.

Create scaffolding

AT teams treat the story wall as their "cognitive ally" (Furniss 2004). They use it to simplify the task of sprint and project management. The story wall remembers the current state of the sprint. It remembers what the user stories are. It remembers who is assigned to each story card.

By standing in front of the wall during stand-up meetings, discussing, touching and pointing to the story cards, the story wall helps the team recall. The story wall helps to show relationships. This cognitive support allows the team to use their limited cognitive resources to communicate and reason with other team members and the stakeholders. Thus, the team members use the story wall as a scaffolding to support recall, sharing understanding and reasoning.

Mediating artefact

The story wall is a mediating artefact brought into coordination for the completion of the task of developing a user story.

Coordination of resources

The story wall reduces the ambiguity of what the sprint state is. It coordinates what everyone's idea of the sprint state with what the collective wisdom of the sprint state is (since the wall is developed by the group). In addition, since the information resource to which the story card corresponds is a task, coordinating the information resources on the story wall coordinates the team.

Issues

Throughout the interviews, the principal concern about the story wall expressed by the participants is whether or not the story wall reflects the state of the sprint. If the participants do not believe it accurately reflects the sprint, the value of the story wall for sharing understanding is limited.

Often this problem manifests itself when a single story card requires development by both Java and legacy developers. Both these types of developers share the same "Development" phase. If one developer finishes work and the other remains on the

story card, the card remains in “In Progress”. Thus, the wall does not reflect the state of Development. As the two types of development are carried on simultaneously, it is not meaningful to break phases into successive “Java Development” and “Legacy Development” phases.

One participant creates two story cards to represent the effort required by both developers. Others solve this problem by using adhesive sticky labels to signify the different developers, as described by the following participant.

> *Often it will start off as one card, because the business won’t know and then we will go and investigate and find out, and sometimes it may mean they will work on the same card…. But yes we have card stickers; they stick their stickers on, so sometimes there’ll be two stickers.* ***Developer***

After finishing the task, the developer takes his or her label off the card. This allows the other developer to continue working. Either of these techniques allows the wall to better reflect the true nature of the sprint.

Another project manager relates a situation in which a developer did not think the story wall adequately reflected the developer’s work. The developer felt that when the tester found a problem in a user story and moved the card back to “Waiting” sub-phase for Development, it looked as if the developer had not done any work on the story. While working on the story wall during scrums, the project manager agreed that it was not reflecting sprint state:

> *But with the wall, when we had our stand-ups in the morning, we realised that if something went through development and then into testing … and there was a bug, if you moved it back into the development column on the wall, it made it look like she hadn’t done it, and it made it look like there was still that work to do…. Well, it just wasn’t a good representation.*
> ***Project Manager***

Together, they came up with a solution of using a sticky label to represent “Tested with bugs”.

Lastly, during the day, if developers finish a story card, they are more likely to update JIRA and not move the story card on the story wall. Thus, the story wall tends to be updated at daily stand-up meetings but not throughout the day.

4.3.3.3 Graphical Representations

Many participants cite the use of diagrams, graphs, and other representations in the story creation process. These diagrams can be developed in collaboration with stakeholders and members of the software development team, or they can be developed by the business analyst and altered with others.

As described in Section 4.2.1, participants mention the use of a variety of graphical representations used while creating user stories:

- "As is" and "To be" diagrams;
- Process diagrams;
- Screen mockups and wireframes.

Use of Artefact

The main use of the graphical representations in AT is to aid sharing understanding between members of the software development team and/or the stakeholders. It also supports problem-solving. However, there are many different kinds of representations. As with the story wall format, the kind of representation used depends upon what the business analyst is used to using.

Participants report these representations often begin by depicting the system as it is and then drawing out from the stakeholders what changes they want. Next, facilitators often move towards either creating a new graphical depiction or altering the original one to show how changes must be made. Participants mention "as is" diagrams and "to be" diagrams for this purpose. Another project manager worked with stakeholders to create a process diagram on a white board of the existing process and then drew "pain points" in a different color to where the stakeholders wanted changes to be made.

As these kinds of representations are unconstrained, the stakeholders and software development team often break down a system representation to develop representations of sub-systems and components, as this participant reports:

> *Some of the artefacts for that can be, ah, pictures so we take photos of whiteboards, documents we'll scribble out things on a piece of paper rather than in Visio. We'll draw up screens, to we'll put that there and the down side here and then if we had that there.* ***Business Analyst***

Diagrams and graphical representations require an abstraction, a redefinition of the representation. Using such a graphical representation changes the nature of the cognitive task. Instead of listening, the others see. As mentioned in Section 4.3.3.1, this change reduces the cognitive load. Listening to someone talk about a user story is a linear process in which the listeners must not only understand what is being said (natural language processing) as it is heard, but what had previously been heard and understood must be buffered and integrated with the new input. A graphical representation does not require the linear information processing.

In addition, a graphical representation is a transformation in which abstraction plays a role. Important aspects of the representation are usually represented more clearly.

Analysis

Mediating Artifacts

Diagrams and other graphical representations are a way of transforming into a different representation the user story, or perhaps how the user stories hang together. The mediating artefacts are used to share understanding and ultimately aid the completion of the task.

Coordination of Resources

A graphical representation is the opportunity to “get everyone on the same page”. The transformation from one representation into another is an opportunity to detect errors. Coordinating the two representations allows one to test whether they understood correctly what the stakeholder meant with the user story. Thus, a diagram can often unearth mistakes, either from the original stakeholder or the misinterpretations of listeners.

Issues

When choosing the kind of graphical representation to use to coordinate the representations and share understanding, the facilitators may discover not all are suitable for every situation. One participant created a green print/blue print, but they were not “into pretty pictures”. They preferred a process diagram.

4.3.3.4 AT Cultural Artefacts

AT have a number of artefacts that are part of their software development process. AT cultural artefacts are created by the team. However, once they are created, they are not changed. They are usually mounted on the wall near the story wall, as seen in Figure 16.

Use of Artefact

Each AT project creates a "success slider" that defines where the emphasis should fall between two constraints that trade-off between each other. AT teams also develop a social charter of behaviour, sometimes called the social contract (similar to Cohn's (2004) "Ground Rules"). Lastly, AT projects have a "focusing question".

The use of these artefacts is to remain present and visible and guide the software development team if they need guidance. The AT Cultural Artefacts support the cognitive activity of decision making.

Analysis

Mediating Artefacts within the *Horizon of observation* – The AT Cultural Artefacts are a hybrid of the Artefact Theme and the Physical Theme. Thus, the principles they most exemplify are mediating artefacts as they are used in coordination with the completion of the user story and horizon of observation, because they become part of the environment of the team area.

In the following transcript excerpt, the participant points out how AT uses these artefacts to support cognition through the software development process.

> *So often you'll be coming across to meetings, but you'll be going across, so you can see a snapshot view as you go past a story wall. You can see really easily if a block has happened or if stories aren't moving across the wall and you're trying to work through something, you're like,*
>
> *'What's the problem?'*
>
> *'John [Not real name] isn't around.'*
>
> *'What's your slider say?'*
>
> *So you can refer to it.* ***Project Manager***

What the developers do through the process of coding and testing has a substantial impact on the outcome of the project. As one of the project managers says, "developers often have to make day-to-day decisions" and the cultural artefacts are there in the team area to guide them and keep them aligned with what the stakeholders consider most important. In addition, the social charter makes everyone aware of the kinds of behaviour expected.

Further, the AT cultural artefacts give the development team permission to act upon their guidance. Instead asking the business analyst, project manager or stakeholder, to move forward on the user story in a certain way, developers only need look at these cultural artefacts. They probably will not even need to read the posters on the wall, but just look at them to remind themselves of what are the guiding principles.

In this way, AT uses these artefacts to guide the developers in the same way Hutchins (1995a) reports Micronesian navigators use the stars: as a representational guide, not computation one (p. 66).

Issues

One participant reports that although the cultural artefacts are on the wall, the developers do not always use them for guidance. This participant reminds them to consult the cultural artefacts on the story wall.

> *So I'm constantly, you know, if you have a stand-up in the morning, for example, and you might have a developer saying, "I'm working on this story but I need to talk to John [Not Real Name] the SME because I'm not sure which way to go," and Bruce is away until next week, and I'm like, "What's the issue?" and it comes down to a quality or time issue, and I say, "What does it say on the sliders?" Time is paramount; time is paramount, so move ahead on that basis.* ***Project Manager***

If the project manager has to remind the developers the cultural artefacts are there for their help, then this may indicate the unchanging artefacts lose visibility that the continually used story wall does not. In this situation, the use of the physical environment may cause these cultural artefacts to become as unnoticed as the environment.

4.3.3.5 Sticky Labels

Sticky labels are used to label artefacts or information temporarily. They are used to distinguish artefacts.

Use of Artefact

Participants report three uses of sticky labels:

- To identify to whom a story card is assigned;
- To prioritise a list of user stories;
- To identify a problematic state on the story wall.

1) Members of the software development team use them to label story cards and take responsibility for the task as described in the excerpt below:

> *When they go to pick up a card and they'll have a picture. Like it'll either be a profile picture or a cartoon caricature, and so that's great because as a PM, you know who's doing what at any one time.* ***Project Manager***

Thus, the use of sticky labels can communicate information. The sticky labels help the story wall radiate information.

2) One facilitator gave stakeholders three differently coloured sticky labels. Each colour represented a priority. This facilitator asked the stakeholders to use the sticky label to indicate the priority of each user story on a list on a flip chart by placing the appropriate sticky label next to it. The facilitator could "see" the priority building up next to certain user stories, as depicted in Figure 20 Priority "Building Up".The use of colour helps make what might be an intensely cognitive task – determining which user stories are of higher priority - into a less taxing perceptual judgment. The facilitator can see which user stories had the most high priority labels next to them.

This is not to suggest that the use of sticky labels is superior to using the four corner MoSCoW sort. Both ease the cognitive load of seeing which stories have higher priority. They also ease the load of prioritising.

However, in contrast to the method with the cards sorted into piles, this sticky label method retains information of dissenting opinions.

As in the first example of the use of sticky labels, this use helps communicate information. Both of the uses also support the cognitive activity of information processing, in that both are instantiated rules used collaboratively.

Analysis

Coordination of Resources

The sticky labels literally tie a resource to the plan, indicating "current state" – who is working on the story card. If using a colour or a graphical representation, this sprint state may be obvious from afar. This characteristic alleviates the cognitive load in understanding the current state of the sprint.

Issues

While one business analyst questioned the value of using a label to differentiate to whom the card is assigned, there were no cognitive issues mentioned.

4.3.3.6 JIRA

AT use the Jira project tracking application as an "electronic story wall", as described in Section 4.2.2. A JIRA helps the process of information seeking by not only storing information for a user story, but connecting an individual user story to associated information stored on the AT wiki (described in 4.3.3.7).

Use of Artefact

When asked what the difference between JIRA and the story card is, one participant emphatically responded that they are the same thing. However, other participants indicated the application allows the software development team to store more information such as "key requirements" and "comments".

Uses of a JIRA:

- To hold data from both the story card and the story wall in electronic format;
- To tie the user story to any documents in the AT wiki;
- To assign user stories to team members;
- To enable push notification;

- To allow reporting;
- To enable the teams to print the user stories out.

Analysis

Coordination of Resources – A JIRA affords many of the same kinds of coordination remotely. One can log in remotely and check the status of individual JIRAs. One team worked from home extensively so they had this expression as one of their cultural artefacts:

> *We had "Jira was King" because a lot of my team work at home. So Jira had to be the most up-to-date in terms of what you had to do next.* ***Project Manager***

Behavioural Trigger Factors – A JIRA holds the same kind of information as does the wall. It holds a status variable and it is assigned to a team member. When either of these changes, the system sends a push notification to the concerned parties.

Issues

There are several issues with JIRAs:

- Synchronisation or coordination of information resources;
- Difficulty showing a hierarchy of user stories and sub-tasks;
- Formatting.

Using Jira introduces a new problem – synchronisation. The story card is created before the JIRA electronic card. Therefore, the information originally loaded into the JIRA is from the story card. Most business analysts delay entering the user stories into Jira project tracking until they have gathered most of the information they need.

However, it is not this kind of information that creates the synchronisation problem. It is the information that JIRAs shares with the story wall – the status and the person to whom it has been assigned. This is the information that participants identify as causing the synchronisation problem.

While creating different representations often helps uncover mistakes, in this case, it sometimes creates them. Error detection happens when two representations are brought

into coordination; one can see the mistakes, if there are any. However, as a JIRA is electronic and the story card is on the wall, one seldom compares the two.

Most participants mention it is difficult to keep both the story cards and story wall synchronised with the electronic records on Jira. Most participants acknowledge that developers – who are at their desks most of the day, sitting at a computer, and logged into the system – use the Jira project tracking system.

The project managers and business analysts, who one participant confesses are never at their desks, prefer the wall.

> *I think the wall, ah, is definitely where a PM will go first, because you walk past it all the time. And it also highlights to you, you know, why, why that developer is on six [story cards]? That's too many.* ***Project Manager***

In theory, team members changing the status of JIRAs or assigning them to another team member should also go to the wall and move the cards. However, it is questionable whether they do both, one or the other, or neither.

One project manager points out that during stand-up, the team updates the story wall, so the project manager knows that it is correct at that moment.

> *I believe this one actually <indicating the wall>, because they are more like, because we have a stand-up every day, and we'll stand right in front of this wall, and every day I say, is the wall up to date?, and they'll all rush up and <gestures moving things around> and I know that's correct, whereas I know damn well they don't go back and update the Jira.* ***Project Manager***

Another problem with the Jira system is that it is what one interviewee calls "flat". It shows neither a hierarchy nor relationships between user stories. Participants report it would be good if they could maintain a master card relationship with the split user stories and the tasks. This would give AT a desirable traceability. It would help show stakeholders the relationship between what is on the wall and their original user stories.

Lastly, several participants mention that using Jira application is not easy because the formatting is, as one puts it, "shocking".

4.3.3.7 The AT wiki

AT has a wiki in which each project team builds an area in which to compile documentation that does not fit on a story card. This documentation can be hyperlinked to individual JIRAs.

Use of Artefact

Originally AT tried to attach this kind of information to the JIRA but they found it did not work well. At some point they introduced the wiki as better repository of project information. Documents placed on the wiki can link directly to a JIRA.

Analysis

Creating Scaffolding

As described in Section 2.3.2.1, people create scaffolding when they use external resources to simplify cognitive tasks. The wiki is such a scaffold. AT can build persistent information about the project. The wiki forms two functions. First, it allows both the stakeholders and the software development team to share documents about the project. Second, it allows the software development team to hand over information about the project to production support.

It simplifies cognition by creating relations between information.

Issues

No participant expressed any issues about the wiki.

4.3.4 Summary

Through the development of three DiCoT themes, one can see how the use of requirements as an artefact embedded in the world enables a high degree of shared understanding and communication. The information flow theme identified the user story card as the central artefact of the user story process. Indeed, throughout the process, it acts as an information hub. The sub-processes of the user story process each result in a representational state of the story card until the story card is ready for the software development team to implement. Each of these representational states adds and delimits the user story to be implemented.

The artefact theme focuses on how the embedded cognition of things-in-the-world draws more cognition outward. The story card and the story wall both are cognition embedded in the world. Other artefacts that add the shared cognition and collaboration of the stakeholders and the software development team are identified.

As predicted, the physical theme is the least developed through the use of interview data. However, this theme still identifies the emphasis on the physical environment supports communication and sharing understanding.

From the DiCoT analysis, the cognitive activities involved in the user story process emerged. The next section will document the kinds of cognitive activities involved in the user story process.

4.4 Distributed Cognitive Activities

In a Distributed Cognition system such as the ones described by Hutchins (1995a, 1995b) and Hollan et al. (2000), cognitive processes emerge from interactions among people, with people and their artefacts and with people and their environment in a functional system.

There are two goals of the functional system enveloping AT's user story process. The first goal is to create and maintain the user story as a software requirement. The process must specify what the software development team should develop. To that end, the stakeholders and the software development team collaborate and share cognitive activities to create and transform representations of what the stakeholder wants.

Second, the user story process supports sprint management; effectively, this function covers the use of the user stories to build software. By using the roughly equal effort cards on the wall it presents a "physical, visual map" of where the team is going, as one participant put it.

The story card and story wall are cognitive artefacts in this functional system. Hutchins (1999) defines cognitive artefacts as things people create to aid, enhance or improve cognition. Evidence from the AT user story process and the DiCoT analysis indicates these artefacts support cognitive activities.

These cognitive activities include reasoning, understanding, problem-solving, decision making, recalling, information seeking, explaining, and information processing, and talking. For the purposes of analysis, these have been categorised as those involving:

- Communication;
- Transformation of representations;
- Coordination.

Talking and information seeking are communicative cognitive activities. One could argue that talking is the glue that holds together the collaborative activity of the user story process. Naturally, in a functional system, communication is not restricted to between people. For example, the story wall radiates information, a communicative function.

Sharing understanding and explaining straddles between communicative and transformative, since whatever one is communicating may either transform one's own understanding, that of another, or transform the group understanding. Sharing understanding is especially evident in cross-functional teams or when stakeholders come from different domains. Explaining, while communicative, can involve transforming representations. One could draw a picture or paraphrase in order to explain.

Referring to Simon's (1996) definition of problem-solving (p. 136), it is easy to see these types of cognitive activity as transformations. Information processing can also be a transformation, in that well-known rules are being followed, such as the agreed upon way cards move on the story wall.

The cognitive activities of understanding and reasoning start to merge with coordination, if coordination of information resources involves transforming with an eye to the appropriate transformation.

Lastly, recall, decision-making and perceiving (perceptual judgments) are cognitive ways of coordinating representations. Thus, the software development team coordinates the representations on their story poker cards to make a decision on the estimates.

There is overlap within these categories. At all points, if these activities are carried out by more than one person, there will be the associated cognitive communicative activity. For example, by communicating, members of the software development team may coordinate information. By drawing a process map on a whiteboard - a transformation of a representation - a stakeholder conveys an idea once again, helping others to coordinate their understanding of what he or she represented in words on a story card. This redundancy aids the robustness of the functional system.

Cognitive Activities Involved in the User Story Process					
Communicative			**Transformative**		**Coordinating**
Information Seeking	Explaining			Understanding	
	Sharing Understanding				Recall
Talking				Decision Making	
			Problem-solving		
			Information Processing		Perceiving
			Reasoning		

Table 7 Cognitive Activities involved in the User Story Process

4.4.1 Communication

Building a software system, as Mike Cohn (2004) writes, "relies on information from the heads of very different people" (p. 3). Evidence from the AT interviews supports this idea. One participant even calls communication the "fundamental aspect" of AT's Agile method.

Software development teams communicate in a variety of ways. Some of these are formal events such as stakeholder workshops, elaboration, sprint planning meetings and stand-up meeting. Communication in these meeting is face-to-face, and the kinds of cognitive activities range across the spectrum.

Throughout the interviews, participants give evidence of the importance of cognitive activities involving communication. These activities include talking, questioning, information seeking, and collaborative problem-solving.

As discussed in Section 4.2.2, the members of the software development team talk about what a particular user story means, and stakeholders talk about the meaning and priority of the user story. Questioning occurs when the software development team asks questions of the stakeholders, either during elaboration or stand-up meetings. Information seeking is often an informal communication, when a business analyst must dig a little deeper behind a user story. It can also often occur during the daily stand-up meeting, when a developer may remind a stakeholder of a document they need.

In the following example, a participant points out that by communicating, the team is able to change the state of the story wall during the stand-up in a collaborative way.

> *So in the morning, Tim might say, I have another hour on this, and we'd say to Bruce, have you finished and he'd be like "Yep," so he's off it, and as soon as Tim's finished, we've all been in that meeting, he moves it.*
> ***Business Analyst***

By using these cognitive artefacts, the team communicates about a story card in front of them. The story card aids recall, helping the team members coordinate representations of information. They can collaborate about changing the representation of the story wall to more accurately reflect the sprint state.

There is also informal face-to-face communication, which tends to be more goal-oriented. Someone may wish to ask a question, clarify a sentence, or impart new information. This person may carry a story card, or he or she may have seen the story wall and want to ask a question. The kinds of cognitive activities are supported by informal communication include information seeking and problem-solving, as described in Section 4.3.1.1.

Other methods of communication include electronic means such as texting, telephone and email. Team members even communicate indirectly through JIRA push notifications. Again, the motivation for these kinds of communication can be information seeking and problem-solving.

Electronic and indirect methods are commonly characterised as having a "lower bandwidth". Instead of perceiving facial expression, body language, and tone to confirm one's interpretation, informal methods restrict the stream of information the people communicate (Hutchins 1995a p. 232). Hutchins and Palen (1997) further elaborate on the importance of gestures and the space taken by the speaker. These different but related communication streams combine into "a complex multi-layered representations in which no single layer is complete or coherent by itself" (Hutchins & Palen, 1997, p. 2). Thus, this example illustrates the need for coordination of representations.

4.4.2 Transformations

Transformations in a Distributed Cognition system are those of representations between media. These transformations can be graphical (diagrams, graphs, pictures), verbal (talking, explaining, elaborating, paraphrasing), physical (moving a story card from one wall category to another), or written (on a story card). These internal and external representations and the transformations through which they go are what Bonnie A. Nardi (1996) calls the "structure" of Distributed Cognition (p. 39).

An example several participants recount is the transformation between numbers on story poker cards to spoken language, to a series of negotiations of what the numbers mean to a final resolution. In Section 4.2, it was described how a team shared understanding to derive a bottom line for their estimation process: they decide which user story requires the least effort and assign it with the lowest of the poker card numbers.

This team communicates to transform what they perceive the effort involved in a user story into a baseline. Using that as a yardstick transforms their internal representations of the effort estimate for all other story estimates in relation to it.

One participant gives an example of the consensus-building process (or at least the attempt) by which a group of individual estimates transform into one estimate as a collective expert opinion.

> *If someone threw a three and the testers threw in a 20, ah, a dev threw down a 3 and the tester threw down a 20, you'd have to discuss that. There's got to be a reason why the testers saying this is a 20, because in our world a 20 equates to 5 or 6 days' work, that's a fundamental lot of testing. Working 7*

> *or 8 hours a day. But you might find that the one line code change impacts a whole bunch of regression testing, so those things come up in the story pointing… You get a gut feeling after a while; we'd pick the 3 or the 5.*
> ***Business Analyst***

This emergent "gut" feeling is the result of the cognitive activity of reasoning distributed through this functional system. The story poker game gives the team a social structure that enables them to quickly pool understanding.

Another participant – a project manager - describes how the stakeholders talked through a process that the project manager captured on a whiteboard. The stakeholders transformed their internal representations into the ephemeral, serial representations of spoken language, and the facilitator changed that representation into a two-dimensional "boxes and arrows" graphical representation. Through the use of different transformations, the stakeholders and the software development team define and constrain the requirements for software development.

These transformations are the essence of cognitive computation in a Distributed Cognition system: "the *propagation of representational state* across a series of *representational media*" (Hutchins, 1995a, p. 117, emphasis in original). Thus, transformations support the cognitive activities of problem-solving, decision-making and reasoning. Once again, one is reminded of Simon's idea of representing and re-representing until the solution is "transparent" (Simon, 1996, p. 132). Simon goes on to say, "if the problem-solving could actually be organised in these terms, the issue of representation would indeed become central".

The user story process utilises representation whenever possible. This framing and reframing is constrained by rules, however, they may be defined. Hutchins illustrates the use of rules with an example - how the symbolic symbol processing of a mathematical theorem is constrained by the rules of mathematics (Hutchins, 1995a, p. 117). Likewise, the user story process is similarly constrained. While taking the stakeholders through various transformations of a current process, a participant reports how eventually, a stakeholder will say, "That's what I'm after". The stakeholder recognises when the representation has been transformed into what he or she wants according to the rules in his or her head.

The stakeholders and software development team work through a series of representational states of a cognitive artefact. As such, collaboration and coordination of ideas is important in the AT user story process.

4.4.3 Coordination

As the representations and transformations form the structure of the functional system, there is a need to make sure each representational state as it moves and transforms through the system retains a fidelity to the previous state. Hutchins (1995a) says: "Representational states are propagated from one medium to another by bringing the states of the media into *coordination* with one another" (p.117, emphasis in original). Nardi (1996) states that by coordinating representations, "individual agents align and share within a distributed process" (p. 39).

The importance of coordination does not imply representations should be exact "translations" between media. The affordances of different media may be exploited to elaborate representations. For example, transforming a written sentence on a story card into a spoken sentence requires the speaker to consider pitch and intonation to convey the message desired. Each of these considerations conveys information, which may further elaborate the meaning. In stakeholder workshops, participants note they find "slightly different meanings" and "subtle nuances" between written and spoken representations. Coordination ensures any changes are purposeful and meaningful, not the result of error.

By coordinating representational states, both the stakeholders and software development team detect errors in these states and rectify them. Thus, coordination of representations improves the results of problem-solving, understanding and reasoning.

4.4.4 Summary

Many cognitive activities flow through and among the stakeholders and software development team. These can be broadly categorised as those involving Communication, Transformation and Coordination. Talking and information seeking are communicative cognitive activities. Sharing understanding straddles between communicative and transformative, since whatever one is communicating may either

transform one's own understanding, that of another, or transform the group understanding.

It is thorough the interplay of these cognitive activities that this functional system resembles a cognitive ecosystem.

4.5 A Distributed Cognitive Ecosystem

Through an analysis of this system through a Distributed Cognition lens, one finds it is effectively what Hutchins (2010) calls a "cognitive ecosystem". This change of emphasis of the system from "functional system" to "cognitive ecosystem" follows the progression of Hutchins' thinking in the years since *Cognition in the Wild* (1995a). Hutchins (2010) advances the idea of a cognitive ecosystem as a "web of mutual dependence among the elements of cognitive ecosystem" (p. 706). Hutchins (2010) points to a culture as an example of a complex cognitive ecosystem.

Through the interviews, there is evidence of a strong AT culture. This web of interdependence can be seen in the cognitive activities.

Re-categorising the cognitive activity categories identified in the Section 4.4 as:

- "Communication" as "Sharing,"
- "Transformations" as "Understanding" and
- "Coordination" as "Making sure what you've shared and understood is correct".

One can see the interdependence of these cognitive activities form a web of dependence. The components of the ecosystem share information; this information is transformed until everyone understands with the coordination of these transformations as a corrective feedback function.

These corrective feedback functions occur during the formal steps of the user story process as detailed throughout Section 4.2: the repeated statement of the user stories, the changing into text, the writing on a whiteboard, elaborating a user story, discussing estimates and acceptance criteria.

There are also informal processes when someone discovers something wrong as described in Section 4.3.1.1. These informal processes are an important part of the corrective feedback function in the cognitive ecosystem. The person who discovers the problem uses these informal channels of communication to address it.

It is here in this view of the system as a web of dependence that one can see the importance of a word repeated throughout the interviews - responsibility. While in the following excerpt, this developer is talking about the stakeholders in the project:

> *[T]hey take responsibility and ownership; they feel they are part of the project. There is no agenda, hidden agenda…* ***Developer***

This responsibility for the outcome permeates the project. Like an ecosystem, in which agents function without a controlling force, at AT, the cognitive ecosystem works best when the components act upon the situation without direction. This ecosystem requires acceptance of responsibility for the successful outcome of the project. When a developer realises a story should be split, or when a stakeholder sees from the wall that a story is blocked, they take action. It is through viewing the functional system as a cognitive ecosystem that one can see how important what has been characterised as the informal channels of communication work to support the development of software at Company A. It is responsibility that drives these informal channels.

The responsibility evident among the AT teams is like the commitment that MacKenzie and Monk (2004) find in their study, and is also one of the characteristics of XP teams that Robinson and Sharp (2004) find.

4.6 Benefits found by Distributed Cognition analysis

Having decomposed the interview data into a user story lifecycle and a DiCoT analysis, it is useful to explore why certain aspects of the AT user story process are successful and consider improvements suggested by this analysis.

4.6.1 In the Information Flow Theme

Information flows through the user story process by interactions between people, artefacts, and the environment. Throughout this process, representations of the user story are being transformed as they move from one medium to another.

Analysis of information flow in the AT user story lifecycle allows one to locate the benefits of redundancy and robustness in the information flow

Robustness

As an information hub, the user stories enable the perspectives of the stakeholders and development team to have an impact on the user story. As the group represents and transforms their individual understandings of the requirement to be captured in a user story, they influence its collective representation on the story card.

By working through these representations together, at each of the steps of the user story process, it makes the user story and the process itself more robust to miscommunications and misunderstandings. The evolution of the user story in this way is more likely to lead to a representation that embodies *shared* understanding.

Redundancy

There is a redundancy of representations of information related to a requirement, as the representations are transformed and propagated from one state to another and one medium to another. In many ways, having many representations and coordinating the information represented is a useful check for errors. By transforming and coordinating several representations, subtleties that were previously elusive can emerge.

Overall, the observation and analysis of information flows in the data show redundancy and robustness, to the benefit of the quality of the shared understanding.

4.6.2 In the Physical Theme

There are several ways AT teams exploit the physical environment to ease their cognitive load. They maintain a representation of sprint state in a manner in which information is easy to interpret and use. They build their environment to encourage informal communication with each other to help solve emerging problems. In the large meeting room, AT adapts the physical environment in order to stimulate flow of information. AT also uses equipment such as flip charts and whiteboards to create a vertical dimension to allow people represent and re-represent what they want to convey.

By taking advantage of the physical environment, AT:

Simplifies information transformation

Within limits, using visual cues can ease cognitive load. If one can perceive certain information by looking instead of reading, it changes a cognitive judgement into a perceptual one. One can recognise to whom a card has been assigned without having to think about it. One can see what the status of a story card is by its position on the story wall.

Reading written language – making a serial transformation of written symbols - is a more involved cognitive task than perceiving a colour. In his article, *How a Cockpit Remembers Its Speed*, Hutchins (1995b) advances that what he calls devices in this context "reconfigure" a cognitive system to reduce "the requirements for scarce cognitive resources". Hutchins states that it is not that the devices help the pilot recall the speed. Rather, the device knows the speed. Paraphrasing Hutchins' title, a sticky label is how the story wall remembers who is working on a story card.

Makes information persistent

AT uses a variety of surfaces to make information and different transformations of representations persistent and visible. These representations help the team to coordinate the representations they are using to create software. In a stakeholder workshop, developing a temporary "dimension" upon which a bigger picture can be displayed – a process diagram, a goal – helps the stakeholder's understanding. One does not have to recall what the representation is if all one has to do is look at the whiteboard.

Multiplies the number of ways information can flow

With people in proximity, communication is eased. Team members can see each other and hear conversations. While they would undoubtedly be willing to share cognitive activities even if they were not collocated, because they are, they identify when they are able to help those nearby. It is through this greater awareness of the problems and questions other team members may face that more understanding is shared.

Likewise, participants believe the informal stakeholder workshops need to create an atmosphere conducive to communication. One participant explains: "You encourage

individuals to speak, so you sorta set that up early on". It is this richness from everybody that is needed in the stakeholder workshop.

Thus, an analysis of the interview data supports the contention that AT teams leverage their physical environment in the cognitive process of software development. In particular, cognitive load is eased by using the physical environment to remember information for the team. The facilitators also manipulate the physical environment to support communication.

4.6.3 In the Artefact Theme

Multiple levels of abstraction

The use of artefacts in the user story process brings the user story out of the heads of the stakeholders and the software development team and makes it into "nearly a living thing". By making the requirements things-in-the-world, AT can exploit the space to support both perception and cognition. The requirements as artefacts have affordances and properties AT can use to create and manage the software process. These properties are both in themselves, the artefacts and the world. As the people at Company A are "embedded in the world," they can use space to simplify the cognitive tasks involved in software development (Kirsh, 1995).

The tangible property of the story card allows it to be used to create new artefacts. Combining the cards in a stack, the stakeholders create meaning. Further, the cards within these stacks can be ordered, creating an even different artefact. This artefact effectively transforms the representational state of the card itself as the card's place in the order is added to the story card.

On the story wall, the story card encodes explicitly a property of the sprint status. Thus, the story cards, in addition to being requirements, create an observable property – the sprint status.

Moreover, with all the other story cards, it becomes a map of the sprint progress. As discussed in the Create Scaffolding section of the story wall, 4.3.3.2, the story card represents the story, represents the current status of the story, and *in toto* the story cards

represent sprint progress. Thus, there are three levels of abstraction evident in the story wall.

4.6.4 Comparison with Benefits in Research Literature

The only other Distributed Cognition analysis involving user stories is Sharp and Robinson (2006b). Their focus is somewhat different in that by doing observation they focus on a single team in contrast to this study's focus on the process.

Sharp and Robinson (2006b) are able to perform the DiCoT analysis for the physical theme in more detail by observation, compared to this study's reliance on interview data. An interested outside observer may find much of interest in the very aspects of the physical environment the participants no longer find remarkable.

For example, through observation, Sharp et al. (2006b) were able to identify the "Arrangement of Equipment" principle in the way the software development team organised their computers. However, for the participant, this arrangement may be unremarkable. If so, it is difficult to imagine how this principle could have been identified from the interview data.

However, in many other ways, the findings of this study are similar to those of Sharp et al. (2006b). They note that there are few mediating artefacts (Sharp et al., 2006b). While this study points out subsidiary mediating artefacts, such as sticky labels and the AT wiki, it agrees that there are two main mediating artefacts of the functional system – the story cards and story wall.

However, Sharp and Robinson state "there is little transformation between media" (Sharp et al., 2006b, p. 83). Technically this may be true in persistent media; however, this study finds the user story process so rich in transformations through ephemeral "media" that they cannot be ignored.

The other Sharp and Robinson (2006b) findings concur with this study:

- Information flows are "simple and open" providing greater situational awareness (p. 84);
- The XP team works in an information-rich environment (p. 84);

Using interview data, this case study confirms the basic findings of Sharp and Robinson's (2006b) ethnographically-informed case study.

Through the Distributed Cognition analysis, it can be seen that the use of the story card and the story wall help the software development teams and stakeholders share understanding and collaborate. These advantages are supported by the affordances identified by the DiCoT principles.

4.7 Challenges found by Distributed Cognition Analysis

Many challenges to the user story process were found in the DiCoT analysis:

- Stories may be too big to complete in a sprint or enable planning;
- Facilitation problems dealing with groups of people;
- Differing demands due to legacy and Java programming languages and environments;
- Lack of physical space for story walls;
- The story wall possibly not accurately representing the state of the sprint;
- JIRAs possibly not accurately reflecting the state of the individual user stories.

These were covered in greater detail under the "Issues" of the analysis of each of the DiCoT themes.

However, there are additional challenges suggested by the data.

Losing Information

Some of the steps of the user story process require groups to collaborate to create a representational state. This may create a challenge in that information may be lost, such as differing views. For example, the group must collaborate to decide in which corner to put a story card. Some may think a card is a "Must Have" while others may think it is a "Should Have". The question arises, is the corner a card in simply where the last stakeholder put it, or is it placed in that position by a group decision? Did strong personalities dominate?

These are the kinds of questions James Grenning (2002) addressed when he invented story card poker. Another method used by one participant also preserves this kind of

information. This participant – a facilitator – gave stakeholders sticky labels to place against user stories listed on a flipchart. This method shows when there is disagreement about the priority. The facilitator may wish to explore that particular user story. Some interviewees report that often stakeholders simply do not understand a user story they oppose. If that is the case, it would be better to discuss what this particular story means.

Distinctions Difficult to Discern

Another minor challenge that was identified related to the choice of properties used to make distinctions. This can be a challenge if the distinctions a team uses are difficult to discern. For example, if a team were to use a large number of colours to denote meaning, it may be difficult to remember what all the colours mean. Likewise, it may be difficult to distinguish colours if they are similar (such as parsing teal and robin's egg sticky labels).

This problem could be potentially much worse for those who are colour blind or have other visual limitations.

Overloading Meaning

The story of how the project manager and developer used a sticky label to adjust the representation of the wall raises an issue. Instead of using the sticky label to signify to whom the story card is assigned, they used it to adjust the status, "Tested with Bugs". While the team this project manager described only had one developer so there was no need to use sticky labels for identification, it is possible some teams may find a need to use sticky labels for more than one function. In order to convey information, each part of the wall is effectively coded. There may be a point at which the number of codes diminishes cognitive performance. If so, it raises an interesting question: How many distinctions can one apply to the story wall before it becomes overloaded?

A possible diagnostic may be to ask a member of another team to look at the wall and explain what the meanings are. Naturally, this person may not understand the underlying meanings the team assigns to the codes. However, if the patterns are recognizable to an outsider, the team has not overloaded the wall.

4.7.1 Comparison with Challenges in Research Literature

One downside of collocation reported in research literature (Koskela & Abrahamsson, 2004) is that the communication between members of the team may disturb those not involved. In particular, he mentions pair programming as distracting. None of the participants report this as a problem.

Interviewees report AT solves the problem with the XP customer reported in literature (Koskela & Abrahamsson, 2004; Martin et al., 2004) by splitting the role. Teams often recruit an SME to be on the team (and collocate) for the duration of the project. This person is the domain expert available to help the team. However, the key stakeholders still come to meetings and make many of the decisions XP customers make, as reported in literature (Martin et al., 2004).

4.8 Summary

The findings of this research provide insight into how one IT division in a NZ company implements user stories. It illustrates how the integrated use of artefacts and environment help teams in AT to communicate and share understanding. By examining how the stakeholders and software development team use user stories through a Distributed Cognitive lens, this thesis builds three models – the physical, information flow and artefact – to better examine the advantages these cognitive activities gain by being in the world.

This thesis builds upon the work of Sharp and Robinson (2006b, 2009) to draw insight into how people share cognition with each other, artefacts and the environment. Unlike Sharp and Robinson, this thesis focuses on the user story process as opposed to the team. In addition, it uses interview data to develop a "light" Distributed Cognition analysis, demonstrating some utility in this and highlighting some limitations.

The Findings and Discussion provide input to the Conclusion in which a final perspective is developed of how the AT software development teams use user stories with their stakeholders.

5 CONCLUSION

5.1 Summary

The growing adoption of user stories and the motivation to better understand their use in practice is discussed in the Introduction (Chapter 1). The documented utility of Distributed Cognition theory in understanding collaborative teamwork is then used to justify its application as a framework for re-interpreting the use of user stories as an artefact in a distributed cognitive system. This background is used to frame the research aims and questions presented in this chapter.

These research questions guide and constrain the literature review in Chapter 2. This chapter describes the research landscape within which this research is situated and provides a strong theoretical background of Distributed Cognition theory. It also confirms a lack of empirical research in the use of user stories other than as part of wider research on agile projects. This chapter also describes the work of Sharp & Robinson (2006b, 2009) which is closely related to the study described in this thesis.

In Chapter 3 the use of a case study with interpretivist underpinnings is justified as a common approach to the exploratory research questions posed. The research design is further detailed in this chapter, with the use of semi-structured interviews to discover the details of the user-story process, identified as the unit of analysis of this study. The DiCoT framework is then described, as the basis for analysing this process and related interactions with the physical, social and cultural worlds. The limits to the understanding of the interactions and emergent structure gained from interview data only is acknowledged in this chapter, and an argument for this position is provided. This is followed up later in Chapter 5 with a reflection on the outcome of taking this position and the subsequent “light” Distributed Cognition analysis.

Chapter 4 presents the findings in terms of firstly a description of the user process based on interviews. The analysis of this process, based on the physical, artefact and information flow themes of DiCoT are presented and significant results discussed.

Then distributed cognitive activities are divided into three categories of communication, understanding and coordination. The analysis of these categories can be further

developed into a web of dependence in which the responsibility assumed by both stakeholders and the software development team enables ensures the robust sharing of understanding.

The thesis concludes that story cards are used through the user story process in a distributed cognition system to help a software development team and the stakeholders communicate, understand and make this understanding and communication robust through coordination of redundant and persistent information.

5.2 The Distributed Cognition System

The aim of this study is to deepen the understanding of user stories in practice. It proposes to develop such an understanding by answering a series of questions focusing on their use in the world. The primary question is:

> *How is a user story involved in software development as an artefact in a Distributed Cognition system?*

This question will be answered by answering the secondary questions and then relating them back to this primary question. The first of these secondary research questions is:

> *What is the software development process and how are user stories involved?*

User stories are a fundamental component of AT's software development process as detailed Sections 4.1 and 4.2. While the first of these sections was a generic discussion of the software development process, Section 4.2 detailed the user story process. AT uses user stories in much the same way described in Section 2.1, the typical user story process. AT creates user stories in stakeholder workshops, sometimes supplemented by interviews and observation. The stakeholders prioritise the user stories. A key stakeholder elaborates the user stories to the software development team, who then estimate user story effort. Lastly, the stakeholder sets the acceptance criteria with help of the software development team. The story cards are used to create the story wall to manage the sprint. Although some variations are reported in the implementation of these activities in practice, there is an overall consistency in and acceptance of the Scrum and Agile techniques. As noted in 4.2.4., the use of the Jira project tracking application and the AT wiki are practices not recognised as part of the "agile canon".

The detailed description of the AT user story process is used as input to answer the second of the secondary research questions:

> *What distributed cognitive activities involve user stories as a cognitive artefact?*

This question is answered explicitly in Section 4.4. As per Table 7 in Section 4.4, the high-level cognitive activities such as talking and information seeking are identified in the user story process description and the DiCoT analysis of the process. Talking is an immediate and direct communication of issues and understandings within a high communication bandwidth. Talking occurs in every stakeholder meeting, elaboration, stand-up and sprint planning session. Talking is the cognitive activity that functions as the glue for this collaborative functional system. Talking moves information and transforms it.

The cognitive activity of information seeking is in evidence when people seek information from each other, artefacts or the environment. Information seeking is goal-oriented. A member of the development team seeks to the sprint state when looking at the story wall. A business analyst creates a spike to determine whether a user story is plausible or a stakeholder seeks information to argue the case to reprioritise a "parked" user story. Information seeking may be through formal or informal channels.

In addition, there are cognitive activities such as sharing understanding, problem-solving and reasoning. These activities revolve around the greater spread of knowledge, information and understanding through the functional system. This transfer can occur when people from different roles and domains discuss the user stories from their perspectives or elaborate on their business value and impact on their domains. They are sharing their understanding.

When team members and stakeholders reason, they may be considering the conflicting impacts of their differing perspectives. When the developers ask the stakeholders to reprioritise a user story because of dependencies, they are reasoning from the basis of their technical expertise. The stakeholders may reason among each other through out many of the negotiation processes.

Problem–solving, on the other hand, may be the cognitive process by which a number of different views are reconciled. If a view is a representation, the coordination of representations is needed to solve the problem. Thus, the differing estimates for a user story must be reconciled through a problem-solving process.

Likewise, individual components of the story card – the priority, the estimate, the acceptance criteria – all are a result of a decision. This process of decision making continues throughout the user story process. For example, while the priority of a user story is determined before it is estimated in detail, once the stakeholders understand how long it may take to develop, they may reprioritise the story.

Lastly, recall and perception are cognitive activities supported by story card and story wall artefacts. Because the card is the user story, one does not need to consult a list in a file to remember what the story is. One does not need to hold in one's mind a number of user stories. The stories are on the cards, and all one has to do is look at them.

By making how things look communicate information, information that otherwise would entail more complicated cognitive activities to access becomes a perceptual judgement. When perceiving a red stop light, a driver knows to stop. Likewise, one can see the state of the sprint by looking at the story wall.

The cognitive activities discovered through the Distributed Cognition analysis were categorised into groups:

- Communicative;
- Transformative;
- Coordination.

There is overlap between the activities in these categories, as described in Section 4.4. However, what characterises them is that they are examples of distributed cognitive activities in this functional system.

In light of the Findings and Discussion (Chapter 4), examining these two secondary research questions looks like examining two side of the same coin.

How do user stories support Distributed Cognition in terms of sharing understanding?

How do user stories support Distributed Cognition in terms of communication?

Specifically, the story cards support the shared understanding of the software requirement during the user story process by providing a single, public information hub that is not only embedded in the world but also loosely coupled with the cognition of a number of people.

Likewise, the user stories support communication by making recall simpler, as in the use of story cards on the story wall during stand-up meetings and during the prioritisation. Through their use on the story wall, these "trackable" story cards show the sprint status.

However, in a more general sense, the AT culture that has developed around the user story process is a cognitive ecosystem as described in Section 4.5. Its robust sharing of understanding relies on the responsibility of its components, the team members and stakeholders, to respond to emerging problems in through both formal and informal channels.

The next research question concerning benefits and challenges are explicitly listed in Sections 4.6 and 4.7:

What are the benefits and challenges of using user stories in a Distributed Cognition system?

Through the affordances detailed in the information flow theme, AT makes its user story process more *robust* and *redundant*. Leveraging the advantages of the physical environment, AT:

1) Simplifies information transfer;
2) Makes information persistent;
3) Multiples the number of ways information can flow.

Through using the mediating artefacts of the story card and the story wall, AT can take advantage of multiple levels of abstraction clearly visible to transform and re-transform problems until the solution is apparent (Simon, 1996, p. 132).

At this point, the researcher can answer the primary research question regarding the use of user stories in software development as an artefact in the software development. The story cards are used through the user story process to help a software development team and the stakeholders communicate, understand and make this understanding and communication robust through coordination of redundant and persistent information. By leveraging the embedded cognition in the story card artefact and the story wall, the cognition distributed through this functional system supports the software development. This answer of the primary research question is the conclusion of this thesis.

5.3 The Light Distributed Cognition Analysis

It is acknowledged in the Introduction that the information gained from interviewees about the user story process will lack certain detail and have a limited perspective by relying on their memory and perceptions. It is expected that this will limit aspects of the breadth or depth of the DiCoT analysis compared to one based on additional observation of the actual process. It was reasoned that the consequent "light" analysis may still provide some useful insights as well as realize the benefits of the lower effort and time required of participants without an ongoing observation component.

Essentially the question is whether interview data, which is essentially the memories of perceived actions and cognition, provides sufficient data to re-interpret the user story process as a distributed cognition system.

As predicted in 1.1, the lightness of this distributed cognition analysis is most keenly exhibited in the model related to the physical theme. The interview data does not provide significant evidence of what the interviewees perhaps regard as the unremarkable everyday world.

For example, the interviews provided much evidence of the self-conscious activities involved in the user story process. Participants talked readily of how they create, maintain and use user stories. In addition, their discussion of those aspects of the process that challenge them often provides impressive amounts of detail. The participants recount much information about physical environment when describing the challenges they meet. Several facilitators spoke at length about the requirements for

meeting rooms – the kind of room, the kinds of furniture, and the tools and supplies necessary.

However, as the participants do not regard their physical environment such as their team areas with a “strange eye,” the kind of detail needed to discover something new in the unremarked aspect of everyday life is inaccessible.

One may design the interview questions to elicit this detail. However, one motivation of the interview question design is to prompt the interviewees to discuss their experience in their own terms, i.e. the use of user stories in practice. There is a limit in trying to draw out observations of the everyday environment. Not only might it interrupt the flow of the interview, but the value of asking questions from a “strange” perspective may result in the interviewer leading the interviewee.

In contrast, the light view of the artefact theme captures more detail. The interview data provides evidence of the use and importance of artefacts in functional system based on AT’s user story process.

The researcher is able to track the movement of the cards throughout the process. There is ample evidence with which to uncover cognitive activities and the DiCoT principles, and determine how the use of artefacts – primarily the story card and the story wall – support distributed cognition throughout the user story process.

The use of interview data provides ample evidence for the development of the information flow model. For example, participants explicitly detailed information movements and transformations. In addition, the use of flip charts and whiteboards showed information buffering. The interview data also shows how the story wall effectively makes otherwise intensive acts of cognition, such as referring to the Jira project tracking application to extract what the sprint status is, into a perceptual judgement. The story wall also acts as a behavioural trigger, signaling when certain activities are ready to begin.

While the entire DiCoT model may not be completely delineated, insights into the benefits of using user stories have been uncovered. It is by examining these benefits that one can see the motivation behind the recommendations.

5.4 Recommendations

For practitioners

This analysis shows the support of using artefacts and the physical environment to simplify communication and sharing understanding. It can also make more robust the exchange of information and the attendant transformations. By doing so, practitioners can develop a functional system that is characterized by robustness and redundancy.

There are benefits AT has been able to gain from the use of artefacts and the physical environment. These benefits do not necessarily accrue only for agile software development or even software development.

The use of the story card as an artefact has enabled AT to simplify decision making and sometimes play games with them, such as table-top prioritization and story poker (Section 4.2).

For researchers

While clearly identified deficiencies may be attributed to a lack of observation, there may be strategies that this research can identify to overcome them.

Recruit participants from all roles

It would have been useful to identify and recruit testers and stakeholders to interview. A wider range of participants potentially provides additional perspectives. If one looks at the evidence provided by the stakeholder workshop facilitators about the use of artefacts, the physical requirements of the large meeting rooms and the challenge of keeping the information flowing, it is obvious that if they had not been interviewed this rich seam of data would have been missing.

It is conceivable that the roles of stakeholder and tester could have provided an equally rich source of data. In particular, the discussion of the acceptance criteria in almost every interview was relatively perfunctory. It is conceivable that if either of the roles primarily involved in this process had been interviewed, they may have provided an equally rich source of information.

Multiple interviews with questions focused on extracting data for each theme

The original interviews focused on the documenting the user story process. If a second round of interviews had been planned and organised, more detail on the physical environment and artefacts may have been forthcoming.

5.5 Contribution to Knowledge

This study extended the work of Sharp and Robinson (2006a; 2009) by using a Distributed Cognition lens to understand the complex cognitive system around user stories. While Sharp and Robinson (2006b) studied an XP team, this study focuses on the user story process. This is a novel aspect of this study.

Thus, the conclusions of this thesis – that the use of artefacts and the physical environment during the user story process helps a software development team and the stakeholders communicate, understand and check this understanding and communication through coordination of information – adds a new perspective on the user story process.

In addition, examining user stories in a New Zealand context adds a different perspective in the research literature. Lastly, this study uses interview data as the primary data analysed, another unique aspect of this study.

5.6 Limitations

Of the possible limitations and weaknesses cited in Research Design and Implementation Section 3.4.4, two are potential limitations in this research.

First, the Participant Range Problem is a limitation. There is a large diversity of stakeholder types in AT and it would have been difficult to get stakeholders not only from the parent Company A, but the external customers as well, including end users. Having such a broad range would have been optimal. As noted in Section 5.4, in this thesis it would have been optimal to have gained the perspectives of at least one stakeholder and tester. However, recruiting these roles would likely entail a campaign of relationship building that is beyond the scope of a Masters thesis.

In contrast to those parts of the user story process in which the participants were engaged on a regular basis, those components dealing with the stakeholder and tester are less vivid. It would have been interesting to hear the stakeholder's perspective on elaboration. Likewise, the tester may have a different view of the story poker game. The one process that primarily revolves around the stakeholder and tester – creating acceptance criteria – is significantly less vivid.

The second limitation is the Self-Selection Problem. One participant discussed how the teams' project manager "breaks" the user story process by not using the story wall. It would have been interesting to hear are the limitations and challenges of the story wall from this person's perspective. Evidently they are great enough that the project manager did not want to use it.

There are potentially others in AT who also do not follow AT's Agile methodology. Unfortunately, they may not be motivated to participate because 1) they are not interested in talking about Agile and 2) they do not want to voice their opinion. These opinions may have provided evidence of how the use of user stories actually inhibits communication and the sharing of understanding of software requirements.

5.7 Future Work

This thesis has studied an agile software process "in the wild" in a "light" Distributed Cognition analysis. As this study focused only on three DiCoT themes, the results of this "light" Distributed Cognition analysis motivate the researcher to further this study with such an analysis of all five DiCoT themes, including Social and Evolutionary.

Another possibly fruitful avenue to explore would be an extension of this "light" Distributed Cognition analysis to fields beyond agile software development. Teams that share cognitive activities in pursuit of a goal may be ideal for this kind of analysis when observation is not possible. For example, a storm chasing team may react so quickly to events that unless one videotapes them (which may be difficult in a vehicle), it would be very difficult to capture what is happening. By the use of semi-structured interviews, a researcher could construct how a group of people forecast specific meteorological phenomena.

REFERENCES

Abrahamsson, P., & Koskela, J. (2004). Extreme programming: a survey of empirical data from a controlled case study Symposium conducted at the meeting of the Empirical Software Engineering, 2004. ISESE '04. Proceedings. 2004 International Symposium on Empirical Software Engineering, Redondo Beach, CA, USA. Retrieved from 10.1109/ISESE.2004.1334895

Abrahamsson, P., Salo, O., Ronkainen, J., & Warsta, J., &. (2002). *Agile Software Development Methods Review and Analysis* [VTT Publications].

Aiello, G., Alessi, M., Cossentino, M., Urso, A., & Vella, G. (2007). RTDWD: Real-Time Distributed Wideband-Delphi for User Stories Estimation. In N. Guelfi & D. Buchs (Eds.), *Rapid Integration of Software Engineering Techniques* (Vol. 4401, pp. 35-50): Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-71876-5_3. doi:10.1007/978-3-540-71876-5_3

Aranda, J., & Easterbrook, S. (2006). Distributed cognition in software engineering research: Can it be made to work? Symposium conducted at the meeting of the Supporting the Social Side of Large Scale Software Development - CSCW Workshop '06, Banff, AB.

Beck, K. (2000). *Extreme Programming Explained: Embrace Change*. Boston, MA: Addison-Wesley.

Beck, K. (2002). *Extreme Programming Explained: Embrace Change 2nd Edition*. Boston, MA: Addison-Wesley.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., ... Thomas, D. (2001). *Manifesto for Agile Software Development*. Retrieved from http://agilemanifesto.org/

Bell, T. E., & Thayer, T. A. (1976. *Software requirements: Are they really a problem?* presented at the meeting of the Proceedings of the 2nd international conference on Software engineering, San Francisco, California, United States.

Blandford, A., & Furniss, D. (2006). DiCoT: A Methodology for Applying Distributed Cognition to the Design of Teamworking Systems [Dicot]. In S. Gilroy & M. Harrison (Eds.), *Interactive Systems* (Vol. 3941, pp. 26-38): Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/11752707_3. doi:10.1007/11752707_3

Brooks, F. P. J. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. *IEEE Computer*(April 1987), 10-19.

Bugayenko, Y. (2009). Method for Software Cost Estimating Using Scope Champions. In F. Bomarius, M. Oivo, P. Jaring, & P. Abrahamsson (Eds.), *Product-Focused Software Process Improvement* (Vol. 32, pp. 59-70): Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-02152-7_6. doi:10.1007/978-3-642-02152-7_6

Butler, S. J., Hope, S., & Gittins, R. (2001). How Distance Between Subject and Interviewer Affects the Application of Qualitative Research to Extreme Programming Symposium conducted at the meeting of the 2nd International Conference on Extreme Programming and Flexible Processes in Software Engineering, Sardinia, Italy.

Cheng, B. H. C., & Atlee, J. M. (2009). Current and Future Research Directions in Requirements Engineering. In *Design Requirements Engineering: A Ten-Year Perspective* (pp. 11-43). Retrieved from http://dx.doi.org/10.1007/978-3-540-92966-6_2

Chubov, I., & Droujkov, D. (2007. *User stories and acceptance tests as negotiation tools in offshore software development*. presented at the meeting of the Proceedings of the 8th international conference on Agile processes in software engineering and extreme programming, Como, Italy.

Clark, A. (2008). *Supersizing the Mind: Embodiment, Action, and Cognitive Extension*. Oxford, UK: Oxford.

Cockburn, A. (2004). *Crystal Clear: A Human-Powered Methodology for Small Teams*. Upper Saddle River, NJ: Addison Wesley Professional.

Cockburn, A. (2007). *Agile Software Development: The Cooperative Game*. Upper Saddle River, NJ: Addison Wesley Professional.

Cohn, M. (2004). *User Stories Applied for Agile Software Development*. Boston: Addison-Wesley.

Cohn, M. (2006). *Agile Estimating and Planning*. Upper Saddle River, NJ: Prentice Hall Professional Technical Reference.

Delgadillo, L., & Gotel, O. (2007, 15-19 Oct. 2007). Story-Wall: A Concept for Lightweight Requirements Management Symposium conducted at the meeting of the Requirements Engineering Conference, 2007. RE '07. 15th IEEE International

Dingsøyr, T., Dybå, T., & Abrahamsson, P. (2008. *A Preliminary Roadmap for Empirical Research on Agile Software Development*. presented at the meeting of the Proceedings of the Agile 2008, doi:http://dx.doi.org/10.1109/Agile.2008.50

DSDM Consortium. (2008). *DSDM Atern - The Handbook*. Whitstable Kent, UK: Whitehorse Press Limited.

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology, 50*(9-10), 833-859.

Easterbrook, S., Storey, M.-A., & Damian, D. (2008). Selecting Empirical Methods for Software Engineering Research [CSC2130S]. In F. Shull, J. Singer, & D. I. K. Sjøberg (Eds.), *Guide to Advanced Empirical Software Engineering* (pp. 285-311). London: Springer London. Retrieved from http://dx.doi.org/10.1007/978-1-84800-044-5_11. doi:10.1007/978-1-84800-044-5_11

Flor, N. V., & Hutchins, E. (1991). Analyzing Distributed Cognition in Software Teams: A Case Study in Team Programming during Perfective Software Maintenance [dcog]. In J. Koenemann-Belliveau (Ed.), *Empirical Studies of Programmers - Fourth Workshop* (pp. 36-64). Norwood NJ: Ablex

Furniss, D. (2004). *Codifying Distributed Cognition: A Case Study of Emergency Medical Dispatch*. University College London, London.

Furniss, D., & Blandford, A. (2006). Understanding emergency medical dispatch in terms of distributed cognition: a case study. *Ergonomics, 49*(12/13), 1174-1203.

Furniss, D., & Blandford, A. (2010. *DiCoT Modeling: From Analysis to Design*. presented at the meeting of the CHI 2010, Atlanta, Georgia USA.

Gallardo-Valencia, R. E., Olivera, V., & Sim, S. E. (2007). Are Use Cases Beneficial for Developers Using Agile Requirements? Symposium conducted at the meeting of the Fifth International Workshops on Comparative Evaluation in Requirements Engineering

Gallardo-Valencia, R. E., & Sim, S. E. (2009, 1-1 Sept. 2009). Continuous and Collaborative Validation: A Field Study of Requirements Knowledge in Agile Symposium conducted at the meeting of the Second International Workshop on Managing Requirements Knowledge (MARK), 2009

Geering, J. (2007). *Case Study Research - Principles and Practices*. New York, New York: Cambridge University Press.

Gomez, A., Rueda, G., & Alarcón, P. P. (2010). A Systematic and Lightweight Method to Identify Dependencies between User Stories. In A. Sillitti, A. Martin, X. Wang, & E. Whitworth (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (Vol. 48, pp. 190-195): Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-13054-0_17. doi:10.1007/978-3-642-13054-0_17

Goodwin, C. (1994). Professional Vision. *American Anthropologist, 96*(3), 606-633.

Grenning, J. (2002). *Planning Poker or How to avoid paralysis while release planning*: Renaissance Software. Retrieved from http://www.renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf

Hansen, S., Berente, N., & Lyytinen, K. (2009). Requirements in the 21st Century: Current Practice and Emerging Trends. In *Design Requirements Engineering: A Ten-Year Perspective* (pp. 44-87). Retrieved from http://dx.doi.org/10.1007/978-3-540-92966-6_3

Haugen, N. C. (2006, 23-28 July 2006). An empirical study of using planning poker for user story estimation Symposium conducted at the meeting of the Agile Conference, 2006

Hibbs, C., Jewett, S., & Sullivan, M. (2009). *The Art of Lean Software Development*. Sebastopol, CA USA: O'Reilly Media, Inc.

Hirschfeld, R., Steinert, B., & Lincke, J. (2011). Agile Software Development in Virtual Collaboration Environments. In C. Meinel, L. Leifer, & H. Plattner (Eds.), *Design Thinking* (pp. 197-218): Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-13757-0_12. doi:10.1007/978-3-642-13757-0_12

Hollan, J., Hutchins, E., & Kirsh, D. (2000). Distributed cognition: toward a new foundation for human-computer interaction research. *ACM Trans. Comput.-Hum. Interact., 7*(2), 174-196. doi:http://doi.acm.org/10.1145/353485.353487

Hutchins, E. (1995a). *Cognition in the Wild*. Cambridge, MA: MIT Press.

Hutchins, E. (1995b). How a cockpit remembers its speeds [doi: DOI: 10.1016/0364-0213(95)90020-9]. *Cognitive Science, 19*(3), 265-288.

Hutchins, E., & R. A. Wilson & F. C. Keil. (1999). *Cognitive Artifacts* [The MIT Encyclopedia of the Cognitive Sciences]. Cambridge, MA, USA: The MIT Press.

Hutchins, E. (2010). Cognitive Ecology. *Topics in Cognitive Science, 2*(4), 705-715. doi:10.1111/j.1756-8765.2010.01089.x

Hutchins, E., & Palen, L. (1997). Constructing meaning from space, gesture, and speech. In Lauren B. Resnick (Ed.), *Discourse, tools, and reasoning: essays on situated cognition*. New York, NY: North Atlantic Treaty Organization. Scientific Affairs Division.

IBM. (2011). *IBM Rational Unified Proces*. Retrieved from http://www-01.ibm.com/software/awdtools/rup/

Jeffries, R., Anderson, A., & Hendrickson, C. (2000). *Extreme Programming Installed*. Upper Saddle River, NJ, USA: Addison Wesley.

Joint OSD/Services/Industry Working Group. (1993). *MIL-STD-499B DRAFT*. Washington, D.C.: Department of Defense.

Kaner, C., & Bond, W. P. (2004). Software engineering metrics: What do they measure and how do we know? *Methodology*, *8*(6), 1614-1881.

Keith, C. (2010). *Agile Game Development with Scrum*. Boston, MA USA: Pearson Education, Inc.

Keränen, H., & Abrahamsson, P. (2005). A Case Study on Naked Objects in Agile Software Development. In *Extreme Programming and Agile Processes in Software Engineering* (pp. 189-197). Retrieved from http://dx.doi.org/10.1007/11499053_22

Kirsh, D. (1995). The intelligent use of space. *Artificial Intelligence*, *73*(1-2), 31-68.

Koch, A. S. (2005). *Agile Software Development - Evaluating the Methods for Your Organisation*. Boston, MA Artech House.

Koskela, J., & Abrahamsson, P. (2004). On-Site Customer in an XP Project: Empirical Results from a Case Study. In *Software Process Improvement* (pp. 1-11). Retrieved from http://www.springerlink.com/content/780ac3qvf7a92gw1

Kuhn, T. S. (1970). *The Structure of Scientific Revolutions* (Second ed., Vol. 2). Chicago, IL, USA: University of Chicago Press.

Latour, B. (1988). Visualisation and Cognition: Thinking with Eyes and Hands. In H. Kuklick (Ed.), (Vol. 6, pp. 1-40): Jai Press. Retrieved from http://www.bruno-latour.fr/articles/article/21-DRAWING-THINGS-TOGETHER.pdf. doi:citeulike-article-id:2235407

Layman, L., Williams, L., & Cunningham, L. (2004). Exploring extreme programming in context: an industrial case study Symposium conducted at the meeting of the Agile Development Conference, 2004 Retrieved from 10.1109/ADEVC.2004.15

Leffingwell, D. (2011). *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. Boston, MA USA: Addison-Wesley.

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., ... Zelkowitz, M. (2002. *Empirical Findings in Agile Methods*. presented at the meeting of the XP/Agile Universe, Chicago, Illinois.

MacKenzie, A., & Monk, S. (2004). From Cards to Code: How Extreme Programming Re-Embodies Programming as a Collective Practice. *Computer Supported Cooperative Work (CSCW)*, *13*(1), 91-117. doi:10.1023/b:cosu.0000014873.27735.10

Mar, K., & Schwaber, K. (2002). Scrum with XP. *InformIT.com*. Retrieved from http://www.informit.com/articles/article.aspx?p=26057

Martin, A., Biddle, R., & Noble, J. (2004). The XP customer role in practice: three studies Symposium conducted at the meeting of the Agile Development Conference, 2004

Miles, M. B., & Huberman, A. M. (1994). *Qualitative Data Analysis*. Thousand Oaks, CA USA: Sage Publications, Inc.

Miranda, E., Bourque, P., & Abran, A. (2009). Sizing user stories using paired comparisons. *Information & Software Technology*, *51*(9), 1327-1337.

Moon, A. M. (2008, 4-8 Aug. 2008). "Come Together, Right Now" - How the Songs of The Beatles Helped our Product Owners and Teams Live in Harmony Symposium conducted at the meeting of the Agile, 2008. AGILE '08. Conference

Morreale, V., Bonura, S., Francaviglia, G., Centineo, F., Cossentino, M., & Gaglio, S. (2006). Goal-Oriented Development of BDI Agents: The PRACTIONIST

Approach Symposium conducted at the meeting of the Intelligent Agent Technology, 2006. IAT '06. IEEE/WIC/ACM International Conference on

Moser, R., Abrahamsson, P., Pedrycz, W., Sillitti, A., & Succi, G. (2008). A Case Study on the Impact of Refactoring on Quality and Productivity in an Agile Team. In B. Meyer, J. Nawrocki, & B. Walter (Eds.), *Balancing Agility and Formalism in Software Engineering* (Vol. 5082, pp. 252-266): Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-85279-7_20. doi:10.1007/978-3-540-85279-7_20

Myers, M. D., & Newman, M. (2007). The qualitative interview in IS research: Examining the craft [doi: DOI: 10.1016/j.infoandorg.2006.11.001]. *Information and Organization, 17*(1), 2-26.

Nardi, B. A. (1996). Studying Context: A Comparison of Activity Theory, Situated Action Models, and Distributed Cognition. In B. A. Nardi (Ed.), *Context and consciousness: activity theory and human-computer interaction*. Boston, MA: The MIT Press.

Norman, D. (1993). *Things That Make Us Smart*. Reading, MA, USA: Addison-Wesley.

Object Management Group. (2009, 13 October 2009). *UML Resource Page*. Retrieved 14 October 2009, 2009, from http://www.uml.org/

Orlikowski, W. J., & Baroudi, J. J. (1991). Studying Information Technology in Organizations: Research Approaches and Assumptions. *Information Systems Research, 2*(1), 1-28.

Parsons, D., Ryu, H., & Lal, R. (2007). The Impact of Methods and Techniques on Outcomes from Agile Software Development Projects. In T. McMaster, D. Wastell, E. Ferneley, & J. DeGross (Eds.), *Organizational Dynamics of Technology-Based Innovation: Diversifying the Research Agenda* (Vol. 235, pp. 235-249): Springer Boston. Retrieved from http://dx.doi.org/10.1007/978-0-387-72804-9_16. doi:10.1007/978-0-387-72804-9_16

Poppendieck, M., & Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit*. Boston, MA USA: Addison Wesley.

Potts, C. (1993). Software-engineering research revisited. *Software, IEEE, 10*(5), 19-28.

Read, A., Callens, A., Nguyen, C., & de Vreede, G. J. (2011). Generating User Stories in Groups with Prompts Symposium conducted at the meeting of the Americas Conference on Information Systems (AMCIS) 2011 Proceedings - All Submissions, Detroit, MI, USA. Retrieved from http://aisel.aisnet.org/amcis2011_submissions/332

Robinson, H., Segal, J., & Sharp, H. (2007). Ethnographically-informed empirical studies of software practice. *Information and Software Technology, 49*(6), 540-551.

Robinson, H., & Sharp, H. (2004). The Characteristics of XP Teams. In J. Eckstein & H. Baumeister (Eds.), *Extreme Programming and Agile Processes in Software Engineering* (Vol. 3092, pp. 139-147): Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-24853-8_16. doi:10.1007/978-3-540-24853-8_16

Robinson, H., & Sharp, H. (2005a). Organisational culture and XP: three case studies Symposium conducted at the meeting of the Agile Conference, 2005. Proceedings Retrieved from 10.1109/ADC.2005.36

Robinson, H., & Sharp, H. (2005b). The Social Side of Technical Practices. In H. Baumeister, M. Marchesi, & M. Holcombe (Eds.), *Extreme Programming and Agile Processes in Software Engineering* (Vol. 3556, pp. 100-108): Springer

Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/11499053_12. doi:10.1007/11499053_12

Robinson, H., & Sharp, H. (2010). Collaboration, Communication and Co-ordination in Agile Software Development Practice. In I. Mistrík, J. Grundy, A. Hoek, & J. Whitehead (Eds.), *Collaborative Software Engineering* (pp. 93-108): Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-10294-3_5. doi:10.1007/978-3-642-10294-3_5

Rodríguez, P., Yagüe, A., Alarcón, P. P., & Garbajosa, J. (2009). Some Findings Concerning Requirements in Agile Methodologies. In *Product-Focused Software Process Improvement* (pp. 171-184). Retrieved from http://dx.doi.org/10.1007/978-3-642-02152-7_14

Rogers, Y. (1997). *A Brief Introduction to Distributed Cognition*. Brighton, UK: Interact Lab, School of Cognitive and Computing Sciences, University of Sussex.

Rogers, Y., & Ellis, J. (1994). Distributed cognition: an alternative framework for analysing and explaining collaborative working. *Journal of Information Technology, 9*(2), 119-128.

Royce, W. W. (1970. *Managing the development of large software systems: concepts and techniques*. presented at the meeting of the Proceedings of the 9th international conference on Software Engineering, Monterey, California, United States.

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14*(2), 131-164.

Salo, O., & Abrahamsson, P. (2004). Empirical Evaluation of Agile Software Development: The Controlled Case Study Approach [REFERENCE]. In *Product Focused Software Process Improvement* (pp. 408-423). Retrieved from http://www.springerlink.com/content/QL4CWUA10AGHV6PC

Schwaber, K. (2004). *Agile Project Management with Scrum*. Redmond, WA USA: Microsoft Press.

Seaman, C. B. (2008). Qualitative Methods [CSC2130S]. In F. Shull (Ed.), *Guide to Advanced Empirical Software Engineering* (pp. 35). Berlin: Springer Berlin.

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

Sharp, H., & Robinson, H. (2006a). A Distributed Cognition Account of Mature XP Teams. In P. Abrahamsson, M. Marchesi, & G. Succi (Eds.), *Extreme Programming and Agile Processes in Software Engineering* (Vol. 4044, pp. 1-10): Springer Berlin / Heidelberg. Retrieved from http://dx.doi.org/10.1007/11774129_1. doi:10.1007/11774129_1

Sharp, H., & Robinson, H. (2008). Collaboration and co-ordination in mature eXtreme programming teams. *International Journal of Human-Computer Studies, 66*(7), 506-518.

Sharp, H., Robinson, H., & Petre, M. (2009). The role of physical artefacts in agile software development: Two complementary perspectives. *Interacting with Computers, 21*(1-2), 108-116.

Sharp, H., Robinson, H., Segal, J., & Furniss, D. (2006b). The Role of Story Cards and the Wall in XP teams: a distributed cognition perspective Symposium conducted at the meeting of the Proceedings of AGILE 2006 Conference (AGILE'06)

Simon, H. A. (1996). *The Sciences of the Artifical* (Third ed.). Cambridge, MA USA: MIT Press.

Sjoberg, D. I. K., Dyba, T., & Jorgensen, M. (2007, 23-25 May 2007). The Future of Empirical Methods in Software Engineering Research Symposium conducted at the meeting of the Future of Software Engineering, 2007. FOSE '07 Retrieved from 10.1109/FOSE.2007.30

Sohan, S. M., Richter, M. M., & Maurer, F. (2010). Auto-tagging Emails with User Stories Using Project Context. In A. Sillitti, A. Martin, X. Wang, & E. Whitworth (Eds.), *Agile Processes in Software Engineering and Extreme Programming* (Vol. 48, pp. 103-116): Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-13054-0_8. doi:10.1007/978-3-642-13054-0_8

Sykes, J. B., & J. B. Sykes. (1986). *The Concise Oxford Dictionary of Current English* London: Oxford University Press.

Thornton, S. (2009). Karl Popper. In E. N. Zalta (Ed.), *The Stanford Encyclopedia of Philosophy*. Retrieved from http://plato.stanford.edu/entries/popper/

Tufte, E. R. (1997). *Visual Explanations - Images and Quantities, Evidence and Narrative*. Cheshire, Connecticut: Graphics Press.

United Kingdom Office of Government and Commerce. (2011a). ITIL training - the definitive resource

United Kingdom Office of Government and Commerce. (2011b). Prince2 - The definitive PRINCE2 project management training resource.

van Lamsweerde, A. (2000. *Requirements engineering in the year 00: a research perspective*. presented at the meeting of the Proceedings of the 22nd international conference on Software engineering, Limerick, Ireland. doi:http://doi.acm.org/10.1145/337180.337184

Wake, W. C. (2000). *Extreme Programming Explored*. Boston MA USA: Addison-Wesley Professional.

Wake, W. C. (2003). *INVEST in Good Stories, and SMART Tasks*. Retrieved 05 June, 2011, from http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/

Walsham, G. (1995). Interpretive case studies in IS research: nature and method. *European journal of information systems, 4*(2).

Woodside, A. G. (2010). *Case Study Research: Theory, Methods, Practice*. Bingley, UK: Emerald Group Publishing Limited.

Wright, P. C., Fields, R. E., & Harrison, M. D. (2000). Analyzing human-computer interaction as distributed cognition: the resources model. *Human-Computer Interaction, 15*(1), 1-41. doi:10.1207/s15327051hci1501_01

Yagüe, A., Rodríguez, P., & Garbajosa, J. (2009). Optimizing Agile Processes by Early Identification of Hidden Requirements. In *Agile Processes in Software Engineering and Extreme Programming* (pp. 180-185). Retrieved from http://dx.doi.org/10.1007/978-3-642-01853-4_24

Yin, R. K. (2003). *Case Study Research: Design and Methods* (Third ed., Vol. 5). Thousand Oaks, CA USA: Sage Publications.

Zhang, J., & Norman, D. A. (1994). Representations in distributed cognitive tasks [doi: DOI: 10.1016/0364-0213(94)90021-3]. *Cognitive Science, 18*(1), 87-122.

APPENDIX A - Distributed Cognition for Teamwork Framework

Artefact Theme	*Mediating Artefacts* – artefacts help support activities or the completion of a task.
	Creating Scaffolding – offloading into the environment cognition to help simplify cognitive tasks.
	Representation-Goal Parity – representations that more closely resemble the cognitive need or goal are more powerful.
	Coordination of Resources – coordinating resources to help with a task.
Information Flow	*Information Movement* – the way information moves through a system can determine its representation and have different functional consequences
	Information Transformation – changing the representation of information, filtering, abstracting information.
	Information Hubs – central foci where different information channels meet.
	Buffering – allowing new information to wait when it may interfere with an on-going activity.
	Communication Bandwidth – different kinds of information are more effective i.e. have greater "bandwidth".
	Informal Communication – can play an important functional role.

	Behavioural Trigger Factors – when agents are driven by external factors.
Physical Theme	*Space and Cognition* – using space to aid cognition by distilling meaning into physical groupings, for example.
	Perceptual Principle – how representations help cognition by clearly mapping to what they represent.
	Naturalness Principle – representations are more effective in aiding cognition if they are more easily recognizable.
	Subtle Bodily Supports – using the body to retain or help cognitive processes.
	Situation Awareness – cognition is helped when it is accessible.
	Horizon of Observation – what can be seen or hear by a person.
	Arrangement of Equipment – access to equipment and thus information.

Based on Furniss, 2004; Blandford and Furniss, 2006; Sharp, Robinson et al., 2006

APPENDIX B - Semi-structured Interview Questions

Context	What is your job title? What would you describe as your main responsibilities in a software development project?
	About how many years have you been involved in software development projects? About how many years have you been using user stories in software projects?
	In the project you are currently involved in, about how many team members are there? Is this size typical of the projects you have been involved in?

Goal to share enough understanding to create user stories (elicitation/verification)	**Can you please describe the overall process of how a user story is created?** How is the client involved? Who else is involved and what is their involvement or goal(s)? (How are you involved?) How is the information from the client obtained? (Represented? Meetings? Who is involved? Physical layout of the room? Whiteboard? Are there any other ways the user story is modelled? Is there any other way a user story is represented? Are there any other intermediate representations/artefacts? How is the information from the client analysed and used to make a user story? (Who when where why) What other information is used to create user stories How is the physical representation of a user story created (who, when where why?) How is the content of the user story verified with the client? (Who, when where why?) Who gets a copy of the completed user story-(how when where why? Any standard ways of doing things? Templates?
Goal to manage change in understanding and reflect this in user stories	Are User-Stories ever updated/changed? Why? Describe the overall process for this -from the trigger to the change being incorporated. Who is involved? How, when, where, why? (What information is used to make the change? How is this represented? How is it shared? What other artefacts involved?) .

Goal to make use of user stories to design, plan, develop	What use is made of the user stories? (Who, when, how why?) What information from user stories is used in planning? ….designing? ….coding …testing …other activities?? What information from user stories is used to create other artefacts? Info on user stories-how used? User story states?

Conceptualisation of US as artefact and its use **(practitioners overall perceptions)**	What do you think the main purpose of a user story is? (In your role?) Different at different times of the development lifecycle? What are the most important pieces of information on a user story? And what are they used for? (Do you use them for) during different phases of the lifecycle. Who (what role) do you think a user story is most important to? Why? Do you think user stories contribute to evolving and sharing understanding of the business requirements? How?

APPENDIX C - Consent to Participation in Research Form

Consent to Participation in Research

Project title: ***Use of user stories in a software development team***

Project Supervisor: ***Jim Buchan***

Researcher: ***Karen Chance***

- I have read and understood the information provided about this research project in the Information Sheet dated 1 September 2010.
- I have had an opportunity to ask questions and to have them answered.
- I understand that user stories may be referenced (but not their specific content) as evidence of the use of user stories in this team.
- I understand that notes will be taken during the interviews and that they will also be audio-taped and possibly transcribed.
- I understand that I may withdraw myself or any information that I have provided for this project prior to completion of data collection, without being disadvantaged in any way.
- If I withdraw, I understand that all relevant information including tapes and transcripts, or parts thereof, will be destroyed.
- I agree to take part in this research.
- I wish to receive a copy of the report from the research (please tick one): Yes○ No○

Participant's signature: .....

Participant's name: .....

Participant's Contact Details (if appropriate):

……

……

……

……

Date:

Approved by the Auckland University of Technology Ethics Committee on 8 November 2010, AUTEC Reference number 10/261

Note: The Participant should retain a copy of this form.

APPENDIX D - Participation Information Sheet

Participant Information Sheet

Date Information Sheet Produced: 1 September 2010

Project Title: Use of user stories in a software development team

An Invitation

My name is Karen Chance. I am a student from AUT University, currently doing a research thesis as partial fulfilment of a Master of Computer and Information Sciences degree. I would like to invite you to participate in my research into the area of requirements elicitation and specification. In particular this research relates to understanding how user stories are used in a software development team, and contribute to sharing understanding.

Since this research involves understanding current practice, an essential element is partnering with expert practitioners in this area of software development. As expert practitioners in this area, you are invited to share *your* perspective and contribute to the body of knowledge in this area.

Please note that your participation in this research is voluntary in nature, and you may decline or withdraw your participation without any adverse consequences. None of the participants are identified nor will the information gathered be used to hamper, hinder or harm your career.

The following questions and answers are intended to address the most common questions that the participant may ask about this particular research project. If you need further information, feel free to contact the researcher, Karen Chance. My contact details can be found at the end of this document. It is recommended that you use e-mail to reach me.

How was I identified and why am I being invited to participate in this research?

This research project will gather data from interviews. As a member of the software development team or the customer using user stories you have been identified as potential interview participant.

What is the purpose of this research?

The purpose of this research is to investigate and better understand the main factors that make user stories an effective practice in software development. The primary focus of this research is to understand these factors from examination of artefacts and current practitioners' perceptions.

Understanding current practice and perceptions will suggest possible process improvements and lead to the design of better support tools. This will ultimately contribute to faster and more accurate software development.

At the end of this research a report summarising the main results will be made available to you if requested. Furthermore, it is expected that papers may be published in academic journals relating to this particular research project, with all information kept anonymous.

What are the benefits?

As well as adding to the body of knowledge and influencing practice in this general area, the insights gained from this study will be made available to yourself and your colleagues and it is hoped that the knowledge gained will be useful for improving the practice in your organization.

How will my privacy be protected?

All of the materials related to the participants' information (consent form, tape, and interview notes) will be stored at AUT in a locked cupboard for at least 6 years. After that the material will be destroyed.

It is not anticipated that a transcriber will be involved transcribing the recorded interview. If for some unforeseen reason a transcriber is required, they may be given permission to transcribe the recorded interview session only after they sign a confidentiality contract.

The data from the interviews will be anonymised and analysed for principles and insights that are independent of the interviewee's identity. Furthermore, demographic data will be coded and the data stored in a separate place so that the identity of each participant will be separated from their responses.

If participants decide to withdraw from this research project for any reason before the completion of data collection, it is guaranteed that all of the materials relating to their interview will be destroyed as soon as practicable after your request.

In addition, your employer will not hear or see the content of my research. The only people who will have access to your data the researcher and research supervisors.

Finally, as per the confidentiality agreement signed by the researchers and your company, the information obtained from the interviews will only be used for the purposes of this research and won't be shared with any other companies. All of the interview data will be available only to the researchers identified (student researcher, principal supervisor and secondary supervisor).

What opportunity do I have to consider this invitation?

Due to time restrictions in undertaking the fieldwork for the research, we would ideally like to have notice of your agreement within a week of you receiving this invitation.

How do I agree to participate in this research?

To follow up on this invitation to participate in this research, please confirm your acceptance by email to karcha93@aut.ac.nz. You will also confirm your consent to participate in the interview by signing the Participant's Consent just prior to the interview.

Will I receive feedback on the results of this research?

If you would like a report summarising the results of this research, please tick the appropriate box on the consent form.

What will happen in this research?

If you accept this invitation to participate, you will be interviewed on a one-to-one basis by the researcher. This will be a loosely structured interview where you will be asked some open-ended questions related to your experience of user stories. The interviews will be held at your usual work place(s) or any neutral place if requested. The researcher will take some notes for later analysis and also record the interview as a memory aid for clarification of the interview notes taken. The analysis will involve coding and anonymising the data to identify trends and themes that provide insights to practitioners' perceptions of user stories. Note that it is anticipated that the recording of the interview will not be transcribed in full.

At the end of this research a report summarising the main results will be made available to you if requested. Furthermore, it is expected that papers may be published in academic journals relating to this particular research project, with all information kept anonymous.

What are the discomforts and risks?

- During the interview session there is a possibility you may feel uncomfortable about sharing your point of view about the project operations.
- You may feel uncomfortable that your employer will know who is participating in the study and who has selected not to take up the invitation.
- You may feel uncomfortable about having your interview recorded.
- You may feel uncomfortable that your colleagues or managers may overhear what you may say during the interview.

How will these discomforts and risks be alleviated?

- In order to alleviate the first area of possible discomfort, you will be reminded of our assurance of anonymity and confidentiality of all interview data at the start of the interview process. You may choose not to answer specific questions, and you can also withdraw from participating in the interview at any stage. You can also request that your interview data be withdrawn from the study before the completion of data collection.
- The second possible area of discomfort will be addressed by stressing the voluntary nature of participation to both you and your company. We understand the time pressures faced by the company and you, as its employee, and understand that it is not always feasible or practical to participate in such studies. While your employer will know you have been approached, participation or non-participation will not be specifically recorded or communicated apart from the need to organise specific time and dates for your interview.
- Recording of the interview is not a perquisite of conducting the interview. Before the interview begins you will be asked for permission to record the interview, or not. You will be reminded that you can request that the recording be stopped or wiped at any stage of the interview.
- A soundproof room will be requested for the interviews at the company premises, or, at your request, the interview will be conducted at a neutral place away from work.

What are the costs of participating in this research?

Time is the only cost to you. The interview will take around one hour of your time.

What do I do if I have concerns about this research?

Any concerns regarding the nature of this project should be notified in the first instance to the Project Supervisor,

Jim Buchan
Senior Lecturer
School of Computing and Mathematical Sciences
Auckland University of Technology
Private Bag 92006
Auckland 1142
New Zealand
Phone: + 64 9 921 9999 x 5455
Email jim.buchan@aut.ac.nz

Concerns regarding the conduct of the research should be notified to the Executive Secretary, AUTEC, Madeline Banda, *madeline.banda@aut.ac.nz*, 921 9999 ext 8044.

Whom do I contact for further information about this research?

Researcher Contact Details:

Karen Chance
Master of Computer and Information Science Lab,
School of Computing and Mathematical Sciences
Auckland University of Technology
Private Bag 92006
Auckland 1142
New Zealand
Phone: + 64 9 921 9999 x 5410
Email karcha93@aut.ac.nz

Project Supervisor Contact Details:

Jim Buchan
Senior Lecturer
School of Computing and Mathematical Sciences
Auckland University of Technology
Private Bag 92006
Auckland 1142
New Zealand
Phone: + 64 9 921 9999 x 5455
Email jim.buchan@aut.ac.nz

Approved by the Auckland University of Technology Ethics Committee on 8 November 2010 AUTEC Reference number 10/261