THE ROLE OF CUSTOMERS
IN EXTREME PROGRAMMING PROJECTS

By

Angela Michelle Martin

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the requirements for the degree of
Doctor of Philosophy
In Computer Science

Victoria University of Wellington
2009
ABSTRACT

eXtreme programming (XP) is one of a new breed of methods, collectively known as the agile methods, that are challenging conventional wisdom regarding systems development processes and practices. Practitioners specifically designed the agile methods to meet the business problems and challenges we face building software today. As such, these methods are receiving significant attention in practitioner literature. In order to operate effectively in the world of vague and changing requirements, XP moves the emphasis away from document-centric processes into practices that enable people.

The Customer is the primary organisational facing role in eXtreme Programming (XP). The Customer's explicit responsibilities are to drive the project, providing project requirements (user stories) and quality control (acceptance testing). Unfortunately the customer must also shoulder a number of implicit responsibilities including liaison with external project stakeholders, especially project funders, clients, and end users, while maintaining the trust of both the development team and the wider business. This thesis presents a grounded theory of XP software development requirements elicitation, communication, and acceptance, which was guided by three major research questions.

What is the experience of being an XP Customer? We found that teams agree that the on-site customer practice is a drastic improvement to the traditional document-centric approaches. Our results indicate, however, that the customers are consistently under pressure and commit long hours to the project in order to fulfil the customer role. So while this approach to requirements is achieving excellent results, it also appears to be unsustainable and thus constitutes a great risk to XP projects.

Who is the XP Customer? The initial definition of XP resulted in many people interpreting the on-site customer to be a single person. This research has highlighted that a customer team always exists, and goes further to outline the ten different roles that were covered on the team, which range from the recognised “Acceptance Tester” role to the less recognised roles of “Political Advisor” and “Super-Secretary”.

What are the practices that support an XP Customer to perform their role effectively on a software development project? An additional eight customer-focused practices have been uncovered to supplement the existing XP practices. These customer-focused practices together enable customers to sustainably drive XP projects to successful completion. The practices range from those that specifically focus on interaction (both with the programmer team and the larger organisation) e.g. “Programmer On-site” and “Roadshows” to those that specifically look to the well-being and effectiveness of the customer (e.g. “Pair Customering”) to those that highlight the key steps or activities that need to occur along the way (e.g. “Big Picture Up-Front” and “Re-calibration”).
ACKNOWLEDGEMENTS

“We will surely get to our destination if we join hands”
— Aung San Suu Kyi

Completing this thesis has involved many “hands” and I would like to take a moment to express my gratitude and acknowledge the significant contributions made by so many to the development of this thesis.

Eleven organisations invited me into their XP projects and allowed me to explore their implementation of the on-site customer. Thank you for taking the time to share your experiences and insights with me. It is no exaggeration to say that this thesis would not exist without the contribution of these companies and the sixty individuals who participated throughout the interview and observation process.

A number of people within the agile community have encouraged me in this research. Many have provided introductions to companies who would later become case studies. Many more have allowed me to discuss and reflect on the theory with them as it emerged, and more recently reminded me to complete my thesis before our next meeting. Thank you to each of you, your encouragement over the years has been greatly appreciated.

Research is often considered a solitary activity, and while this is largely true, I have gained much from discussing research ideas and progress with my research peers, including: Rilla Khaled, Pippin Barr, Alex Potanin, Craig Anslow, Elizabeth Whitworth and Sonia Chiasson.

Good thesis advisors are hard to find. I have been very fortunate. Robert Biddle and James Noble have been my advisors, guides and mentors, always supporting me on this journey, and encouraging me to go just one-step further. Robert and James somehow found the patience to review yet another draft paper, chapter, or presentation, and without fail, just as my confidence was lagging they knew just the right words to say to help me carry on. Somehow together we have travelled further than I ever believed possible, thank you.

I would like to acknowledge the enormous contribution made by my father, Graeme Martin. Since I was little he has encouraged me to follow my dreams; nothing was ever impossible as long as I put my mind to it and worked hard. I suspect that this thesis may never have even been started without his encouragement, and it might never have been completed without his regular (at one point daily) supportive phone calls. Thank you, your support over the years, has meant more to me than my words here can convey.

Finally, I would like to thank Dr Peter Komisarczuk (Victoria University of Wellington), Dr Rick Mugridge (Rimu Research Limited) and Professor Helen Sharp (The Open University) for their time and attention in reading and examining this thesis.
# TABLE OF CONTENTS

1. INTRODUCTION ........................................................................................................ 7
   1.1 The Problem ........................................................................................................... 7
   1.2 The Context ............................................................................................................ 7
   1.3 Finding a Proposed Solution ............................................................................... 8
   1.4 Structure of the Thesis ....................................................................................... 8
   1.5 Previous Work Credited ..................................................................................... 9

2. REVIEW OF THE LITERATURE ............................................................................... 10
   2.1 The Agile Manifesto ............................................................................................ 10
   2.2 An Introduction to XP ......................................................................................... 12
   2.3 Software Development Processes Research: A New Direction ....................... 15
   2.4 Users in the Software Development Process .................................................... 17
   2.5 XP Case Studies .................................................................................................. 18
   2.6 Other Agile Requirement Practices .................................................................... 19
   2.7 Traditional Analysis and Requirements ............................................................. 20

3. RESEARCH METHOD: GROUNDED THEORY ..................................................... 23
   3.1 What is Grounded Theory? ............................................................................... 23
   3.2 Why is Grounded Theory Appropriate for this Study? ................................ 23
   3.3 Which Research Perspective Will be Used? ...................................................... 24
   3.4 Who is the Researcher (And Why It Matters) ............................................... 24
   3.5 How Will We Use Literature? ............................................................................ 25
   3.6 How We Gained Access to XP Projects ............................................................ 26
   3.7 How We Collected the Data .............................................................................. 27
   3.8 How We Analysed the Data .............................................................................. 29
   3.9 How We Verified the Theory? .......................................................................... 30

4. OVERVIEW OF THE CASES .................................................................................. 32
   4.1 Project Endeavour, KiwiCorP ........................................................................... 33
   4.2 Project Discovery, RavenCorP .......................................................................... 33
   4.3 Project Atlantis, EagleCorP ............................................................................... 34
   4.4 Project Pinto, FalconCorP .................................................................................. 35
   4.5 Project Speedwell, SwiftCorP ............................................................................. 36
   4.6 Project Advance, HawkCorP .............................................................................. 36
   4.7 Project Endurance, TernCorP ............................................................................. 37
   4.8 Project Aurora, SparrowCorP ............................................................................ 38
   4.9 Project Godspeed, KiteCorP .............................................................................. 39
   4.10 Project Dolphin, RobinCorP ............................................................................. 39
   4.11 Project Calypso, OwlCorP ................................................................................ 40
   4.12 Summary ............................................................................................................ 41

5. BEING AN ON-SITE CUSTOMER .......................................................................... 42
   5.1 Project Endeavour, KiwiCorP ........................................................................... 42
   5.2 Project Discovery, RavenCorP .......................................................................... 45
   5.3 Project Atlantis, EagleCorP ............................................................................... 49
   5.4 Project Pinto, FalconCorP .................................................................................. 53
   5.5 Project Speedwell, SwiftCorP ............................................................................ 57
   5.6 Project Calypso, OwlCorP ................................................................................ 62
LIST OF FIGURES

Figure 1-1: The practices of XP ................................................................. 12
Figure 5-1: Project Endeavour Organisational Relationships ........................................... 42
Figure 5-2: Project Pinta Organisational Relationships ..................................................... 53
Figure 6-1: The Customer Team Roles ........................................................................... 70
Figure 7-1: Customer Practices ......................................................................................... 99
Figure 8-1: Impact of Customer Team Roles on the Experience of Being a Customer ......... 126
Figure 8-2: Impact of Customer Practices on the Experience of Being a Customer ............ 128
Figure 8-3: The Impact of Combining the Customer Roles and Practices on Passion and Overwork ........................................................................................................ 129
1. **INTRODUCTION**

1.1 **THE PROBLEM**

eXtreme programming (XP) is one of a new breed of methods, collectively known as the *agile* methods, that are challenging conventional wisdom regarding systems development processes and practices. The agile methods are specifically designed, by practitioners, to meet the business problems and challenges we face building software today. As such, these methods are receiving significant attention in practitioner literature [1]. In order to operate effectively in the world of vague and changing requirements, XP moves the emphasis away from document-centric processes into practices that enable people [2].

One of the core roles within the XP team is the *Customer* role. Beck and Fowler [3, p.18] describe a good customer as someone who:

- Understands the domain well by working in that domain, and also by understanding how it works (not always the same thing)
- Can understand, with development’s help, how software can provide business value in the domain
- Is determined to deliver value regularly and is not afraid to deliver too little rather than nothing
- Can make decisions about what’s needed now and what’s needed later
- Is willing to accept ultimate responsibility for the success or failure of the project.

Practitioners [4; 5] have questioned the practicality of implementing this role. The XP founders provided little further guidance regarding this pivotal role in the XP team, but Beck and Fowler [3, p.17] do acknowledge the risk associated with this role:

> All the best talent and technology and process in the world will fail when the customer isn’t up to scratch

1.2 **THE CONTEXT**

Information Systems Development methodology researchers [6; 7; 8] have expressed a growing concern that existing methodologies do not meet the needs of today’s business and software development environments. Studies [7; 9; 10; 11] in this area have begun to explore practices in natural settings in order to begin to address these issues, but practitioners have not waited for this research, and have instead introduced agile methods, including XP.

XP relies on significant user involvement. Studies [9; 12; 13; 14] have been conducted which cover the types of user and development team interaction issues and benefits proposed by this involvement.

One of the ways the XP community is establishing a body of knowledge is through the collection of real life experiences concerning the implementation of XP [15]. These experiences highlight a number of issues when projects implement the XP Customer role, including who should be the customer [16; 17], how the role scales to large projects [18], and what should be done if the customer cannot be “on-site” [17; 19].

Many questions remain unanswered both about who the customer should be on a project and how to effectively implement the XP Customer Role. This thesis uses the rich descriptive studies that explore actual practice, as suggested by methodology researchers to address that gap.
1.3 FINDING A PROPOSED SOLUTION

1.3.1 THE PURPOSE OF THE STUDY

This thesis develops and presents a grounded theory of XP software development requirements elicitation, communication, and acceptance, which was guided by three major research questions:

- **What is the experience of being an XP Customer?** This question sought to understand how customers and project teams were implementing this role in practice, and what their experience of this role was, both positive and negative.
- **Who is the XP Customer?** This question sought to clarify if it was a single person or if it was more commonly a team of people who undertook the customer role, and if it was a team, what sort of people made up the team.
- **What are the practices that will support the XP Customer?** This question sought to explore any practices the team used to enhance the effectiveness of the customer role on their software development project.

1.3.2 DELIMITATIONS AND LIMITATIONS OF THE STUDY

This research has produced a substantive theory [20], that is, it is specific to the domain of XP software development projects. We do not attempt to develop a more general theory for all software development requirements elicitation, communication and acceptance. This thesis, however, supports later projects that wish to take a wider scope in the future.

To keep the topic coherent and the scope suitable for a doctoral thesis, no attempt has been made to address other issues surrounding XP, such as the productivity of pair programming.

1.4 STRUCTURE OF THE THESIS

This thesis consists of the following chapters:

- **Review of the Literature.** This chapter introduces XP, concentrating on the role of the on-site customer. It also reviews other relevant literature concerning software development process as well as specific literature concerning XP, including the involvement of users in the software development process, case studies of XP projects, and it briefly covers the more general literature concerning requirements analysis.
- **Research Method: Grounded Theory.** This chapter outlines the research method, grounded theory, which was used to conduct this study, including the perspective selected for the study, the data collection and analysis procedures and the verification steps undertaken to increase validity.
- **Overview of the Cases.** This chapter provides an overview of all eleven projects studied; it also briefly touches on the selection criteria and approach we used to select the eleven projects.
- **Being an On-Site Customer.** This chapter explores the in-depth experience of six projects and their implementation of the on-site customer practice and by doing so it answers the first of our research questions. It considers who the on-site customer was, what happened on the project, how the requirements were gathered, communicated, prioritised and verified, as well as the experience of the person, or as we discovered the team, playing the role of the on-site customer.
- **The (Informal) Customer Team.** In the previous chapters we discover that the on-site customer is not a single individual but a team, typically an informal team. This chapter answers our second research question and outlines the roles that emerged from our interview data concerning who needed to be on the customer team.
The Customer Practices. This chapter answers our third research question and outlines the practices that we discovered teams were using to help improve the effectiveness of the on-site customer.

Conclusions. This chapter provides a summary of the emergent theory on the role of customers in extreme programming, identifies future work, and finally notes the specific contributions made by this research to the field.

The general structure of the thesis was guided by Creswell [21].

1.5 Previous Work Credited

This thesis document has evolved from a pilot project [17], as part of INFO408 at Victoria University of Wellington, which was undertaken prior to the commencement of this PhD study. This earlier work only involves the first of the eleven projects studied in this thesis. Additionally, early drafts of some sections of this thesis were published separately in the agile research literature [22; 23; 24; 25]. In all of the papers Angela Martin was the primary author. In the thesis the terms “we” and “our” refer to Angela Martin, typically in consultation with her PhD advisors.
2. REVIEW OF THE LITERATURE

In this chapter we provide a summary of the initial literature we reviewed to place our research in its initial context. In this thesis we will primarily use literature inductively [26] as outlined in section 3.5 entitled “How will we Use Literature”; we will cover additional literature as we interpret our findings in later chapters of the thesis. So, this chapter covers:

- **The Agile Manifesto.** This section provides an overview of the values and principles of the Agile Manifesto created as part of the 2-day workshop held in February 2001 at Snowbird, Utah.
- **An introduction to XP.** This section provides an overview of the background to XP and the key practices and roles of XP, with a particular emphasis on the practices associated with the XP Customer. It also provides a definition of the key XP terms used in this thesis.
- **Software development processes research – a new direction.** A review of the recent challenges posed by researchers in the area of software development process area and their suggested ways forward.
- **Users in the software development process.** This thesis focuses on the customer role in the XP software development process, and this literature provides a background to this area allowing a deeper exploration of the proposed topic.
- **XP case studies.** A review of recent relevant experiences reporting on the implementation of the customer role.
- **Other agile requirements practices.** Other agile development methods include requirements elicitation, communication and acceptance practices. Some of these practices may help to further explore and perhaps address some of the short-comings identified with the XP Customer Role.
- **Traditional requirements practices.** A review of some of the early and well-utilised literature in this area.

2.1 THE AGILE MANIFESTO

In February 2001 a group of seventeen people\(^1\) met in Snowbird, Utah, for a 2-day workshop [27; 28; 29] to discuss the growing field that was referred to, at the time, *lightweight or light methods*. The result of the workshop, was the new term, *Agile*, to refer to these methods; and the *Agile Manifesto* [30]:

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>  
>  - **Individuals and interactions** over processes and tools
>  - **Working software** over comprehensive documentation
>  - **Customer collaboration** over contract negotiation
>  - **Responding to change** over following a plan
>  
>  That is, while there is value in the items on the right, we value the items on the left more.

We use the writings of three of the attendees of this session Highsmith [28], Fowler [29] and Cockburn [27] to provide further detail behind the manifesto. The introductory sentence to the values of the manifesto emphasize that the authors were:

(a) Practitioners, not merely onlookers making rules for others.

---

\(^1\) Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Greening, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marrick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland and Dave Thomas
(b) Uncovering what works in practice, not inventing practices.
(c) Expecting to continue to evolve their understanding of agile by “doing” and helping others to practice agile.

It is also important to notice the concluding sentence (the sentence after the value statements). There is often the misconception that agile “throws away” documentation, process, contracts, and plans (i.e. the things on the right), but the authors do not suggest this; they simply state their (strong) preference for the items on the left. Cockburn [27] specifically goes on to suggest that there is no “opposite” to agile methods, just alternatives, phrased according to their own value systems; people may disagree with these choices but he hoped the manifesto would lead to constructive conversations concerning those disagreements.

The first value of the manifesto emphasizes the importance of people and how they interact with each other. People are not merely roles on an organization chart and they cannot be managed or treated as though they are “plug replaceable”, people are individuals and the quality of the individual interactions matter. The authors agreed that they would rather use an undocumented process with good interactions than a documented process with hostile ones.

The second value of the manifesto emphasizes the importance of working software. Their perspective was that the act of thinking and creating the documents could be useful for team understanding but not for checking team progress. This statement ran counter to common industry practice of the day, which counted documentation as deliverables to be marked off on a Gantt chart, to reassure stakeholders that the project was on track. The intention was for teams to create temporary documentation (e.g. models on walls) and only turn that into long-lasting documentation if and when required. The authors suggest we think of the words “just enough” and “barely sufficient” when thinking of documentation on agile projects.

The third value explores the relationship between the people who want the system and those who are building it. This value emphasizes that we are all needed to build good software. Contracts are useful but without collaboration the software may not work as needed or intended.

The final value is about adjusting to fast-breaking project changes. Building a plan is useful; all agile methods plan, but they provide mechanisms for dealing with changing priorities, rather than “hanging-on” to an outdated plan. Agile emphasizes time-boxing with re-prioritization after, or near the end of, the relatively short time-box.

Highsmith [28] and Cockburn [27] both also emphasize that being agile on a 100 person team is different than being agile on a 10 person team. They suggest that perhaps one of the missing values is that there is a need for different ways of working in different situations, be it scale, complexity, or life criticality.

The Agile Manifesto also includes some oft-forgotten principles that provide additional guidance:

- *Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.*
- *Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.*
- *Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.*
- *Business people and developers must work together daily throughout the project.*
- *Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.*
- *The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.*
- *Working software is the primary measure of progress.*
- *Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.*
- *Continuous attention to technical excellence and good design enhances agility.*
- *Simplicity — the art of maximizing the amount of work not done — is essential.*
- *The best architectures, requirements, and designs emerge from self-organizing teams.*
- *At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.* [31]

The values and principles of the agile manifesto provide the common foundations of the agile methods. One of those methods is XP.

### 2.2 An Introduction to XP

>*XP is the most important movement in our field today. I predict that it will be as essential to the present generation as the SEI and its Capability Maturity Model were to the last* – Tom DeMarco [3]

XP is described in a collection of books known as the XP series. The founder of XP, Kent Beck, published the initial text, *eXtreme Programming Explained: Embrace Change* [2]. In this book he proposes a new lightweight method for small to medium sized teams that he claims will allow us to deliver, and to deliver value, in the face of vague or rapidly changing requirements. Kent Beck followed up his initial text with a second edition [32] that was a significant update to his original text. While we will discuss the second edition and its potential implications for the findings of this thesis, the focus of this section will be on the first edition. It is the first edition that has been primarily used by all of the cases we have studied, and it remains widely adopted in practice.

XP consists of a set of values, principles and practices. The values and principles provide the context for the method, but it is primarily the practices that are used to identify and discuss XP. Beck notes that none of the XP practices are wholly new. Instead, the innovation of this method is turning these practices up to extreme levels, grouping them together and ensuring they are practised as thoroughly as possible.

![Figure 1-1: The practices of XP][2]

For example, code reviews have been an established software best practice for some time and the XP practice of *pair programming* turns this up to an extreme level, so every line is reviewed as it
is being written. All production code is developed with two people side by side at the computer. The practice of collective code ownership\(^2\) is supported by “test-first” development, because changes made to the code can be verified by ensuring the tests still run correctly after the change has been made, allowing all team members to change code with confidence.

One of the key aspects of XP, outlined by Beck and Fowler [3], is the clear separation of roles between the business and technical people. The business responsibilities concern the content of the system, that is, what functions will be built and in what order. The technical responsibilities concern how the system will be built and how long it will take to build the system. It is also clearly recognised that the two sets of decisions cannot be made in isolation, as each has a key role in determining the other.

Beck and Fowler [3] recognise that the customer is the pivotal role on the project, however, there is little evidence of this when perusing the above practices, as the practices appear to be developer-centric. The practices that will be the focus of this study are those that directly involve the customer: on-site customer, planning game, short releases, (acceptance) testing and 40-hour week. Each of these customer-focused practices is briefly described in the following paragraphs.

The on-site customer is the person responsible for making all of the business decisions. This person is expected to know the domain well, be able to make decisions, and to be on-site with the rest of the XP team. The on-site customer is responsible for writing user stories and acceptance tests. A user story is a short paragraph description of a system requirement or feature that is:

\[\ldots \text{understandable to customers and developers, testable, valuable to the customer and small enough so that the programmers can build half a dozen in an iteration.} \ldots\] A user story is nothing more than an agreement that the customer and developers will talk together about a feature. [3, p. 45-46]

In XP, projects are planned using the planning game. XP projects are decomposed into small public releases of software, and within each release are iterations. Releases are small, typically two to three months. Iterations are smaller, typically one to two weeks. The planning game is run at both levels at different granularities. The planning game typically occurs as a “face-to-face” meeting, where:

- The customer prioritisates the user stories\(^3\) in business value order.
- The programmers sign up to estimate the user stories they will develop. They estimate the time needed to develop the story by breaking it into tasks. Each task is estimated in ideal time\(^4\) based on yesterday’s weather. The principle of yesterday’s weather is that the best prediction of today’s weather is yesterday’s weather. This applies to task estimation in the following way: compare the new task to a similar task and use the actual figures from the previous task as the new task’s estimate.
- The team uses the principle of yesterday’s weather to determine the amount of ideal time available for this release or iteration.
- The customer confirms the final scope of the iteration or release. This step can include manipulating the stories, including breaking a story into parts, in order to ensure business value is delivered.

\(^2\) Collective code ownership promotes any programmer changing any piece of code, irrespective of whether they wrote it initially.

\(^3\) Note that after the first iteration defects are included in this process as well as user stories.

\(^4\) Ideal time is the time without interruption where you can concentrate on your work and you feel fully productive.
Once the scope is determined, the programmers are able to enter the coding phase. This phase primarily consists of discussing the story with the customer, implementing the story and ensuring the story passes all of the customer’s acceptance tests. An acceptance test proves the function works as expected by the customer.

XP uses short releases to maximise feedback from end-users and other business stakeholders, therefore regular releases are encouraged, as often as every day, but more typically every couple of months. The last customer-focused practice that we will describe in this section is the 40-hour week. This practice recognises that overtime impacts the quality of the software, as tired individuals are more likely to make mistakes, therefore XP recommends overtime is used sparingly, if at all.

A number of XP practices support the coding phase, including: metaphor, simple design, test-first coding, refactoring, pair programming, collective ownership, continuous integration, coding standards and the 40-hour week. These practices, except for the 40-hour week, are outside the scope of this study. The initial text by Beck [2] provides an excellent overview of these practices.

2.2.1 DEFINITION OF XP TERMS

The following section outlines the key XP terms used in this thesis.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>40-hour Week</td>
<td>The 40-hour week practice recognises that overtime impacts the quality of the software, as tired individuals are more likely to make mistakes, therefore XP recommends overtime is used sparingly, if at all [2].</td>
</tr>
<tr>
<td>Acceptance Test</td>
<td>An acceptance test proves the function works as expected. XP emphasises the use of automated testing wherever feasible [2].</td>
</tr>
<tr>
<td>Big Boss</td>
<td>The big boss, often known as the project sponsor, is responsible for the project [2].</td>
</tr>
<tr>
<td>Coach</td>
<td>The coach is responsible for the process as a whole. They work with each of the other roles to help facilitate the correct use of XP [2].</td>
</tr>
<tr>
<td>Customer</td>
<td>A role on the team for choosing what stories the system has to satisfy, what stories are needed first and what can be deferred, and for defining tests to verify the correct functioning of the system [2, p. 177].</td>
</tr>
<tr>
<td>Customer Proxy</td>
<td>This term is used informally within the agile community to mean a person, typically with an IT background, who fulfils the role of the customer but who is not the “true” business representative (i.e. does not work in the domain).</td>
</tr>
<tr>
<td>eXtreme Programming</td>
<td>XP is a lightweight methodology for small to medium sized teams developing software in the face of vague or rapidly changing requirements [2, p. xv].</td>
</tr>
<tr>
<td>Hang-over</td>
<td>“Hang-over” is a term used informally within the agile community to mean incomplete stories: the stories were started within the iteration but were not accepted as complete by the customer before iteration close.</td>
</tr>
<tr>
<td>Iteration</td>
<td>A fixed period of time, typically between 1-3 weeks. At the end of an iteration working software is delivered to the customer. An iteration gives a heartbeat to a project, and so typically will remain the same.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>fixed length of time throughout a project [2].</td>
<td></td>
</tr>
<tr>
<td>Release</td>
<td>A number of iterations form a release. Unlike an iteration a release will not necessarily be a fixed period of time, although it will typically be somewhere between 1-3 months. The reason for the time period to be variable is that the software will be available publicly and so it must form a cohesive set of functionality that provides business value [2].</td>
</tr>
<tr>
<td>Planning Game</td>
<td>The XP planning process. The business gets to specify what the system needs to do and the development team specifies how much each feature costs and what budget is available per day/week/month. The business decides what to include in the iteration once they have this information [2, p.178].</td>
</tr>
<tr>
<td>Programmer</td>
<td>A role on the team for someone who analyses, designs, tests, programs, and integrates [2, p.178].</td>
</tr>
<tr>
<td>Short Releases</td>
<td>Short releases maximise feedback from end-users and other business stakeholders, therefore regular releases are encouraged, as often as every day, but more typically every couple of months [2].</td>
</tr>
<tr>
<td>Tester</td>
<td>A role on the team to help the customer to choose and write functional tests, which provide feedback into the planning process [2].</td>
</tr>
<tr>
<td>User</td>
<td>An end-user of the computer system that is the person who will use the software in their “day-to-day” activities. The XP Customer is expected to represent the business and “know the domain well” but they may or may not be an actual end-user of the computer system [2].</td>
</tr>
<tr>
<td>User story</td>
<td>A user story is a short paragraph(s) description of a system requirement or feature that is understandable to customers and developers, testable, valuable to the customer and small enough so that the programmers can build half a dozen in an iteration. A user story is nothing more than an agreement that the customer and developers will talk together about a feature [3, p. 45-6].</td>
</tr>
</tbody>
</table>

### 2.3 SOFTWARE DEVELOPMENT PROCESSES RESEARCH: A NEW DIRECTION

XP challenges conventional methodological wisdom [2] by claiming that:

- Team members should be generalists and able to undertake the full role of XP development team roles. Thus, it de-emphasises or questions the need for specialist team members such as analysts, architects, programmers, testers and integrators.
- An on-site Customer who develops user stories (short paragraph(s) description of a system requirement). These user stories represent a promise of a conversation with the Programmer to elaborate the requirement. Short iteration cycles result in the customer reviewing these requirements in software almost immediately, enabling the requirements to be explored and refined. Thus, it de-emphasises or questions the value of formal requirements specification documents as an effective representation of user requirements.
- The cost of changing software does not necessarily rise dramatically over time.
- The use of best practices such as code reviews, if turned up to extreme levels, can result in increased productivity and quality.
Therefore, we explored articles within the scholarly literature that also question conventional assumptions.

Fitzgerald [6] discusses the need to re-examine the foundations of our system development methodologies. He uses the software development methodology literature to expose the fact that most of our concepts come from a period between 1967 and 1977. He explores the changes in the business environment and associated software development projects using a mix of literature, surveys and interviews. Of particular interest are the following trends:

- The increase of small scale rapid development projects to meet the needs of fast-paced business change(s).
- The increase of projects that do not use a methodology
- The most popular technique on projects, irrespective of whether the project uses a methodology or not, is prototyping

He concludes that in this field practice often drives theory, so one of his suggestions is that we explore the software development projects today to derive the next generation of methodologies.

Russo and Stolterman [8] also challenge the existing research on software development methodologies. In particular they challenge the underlying assumptions and purpose of the research. They contend prescriptive methodologies rely on a number of assumptions that may not accurately reflect practice and the complexity of the software development environment. They recommend the need to create methodologies that facilitate practitioner reflection and learning. In particular, they suggest rich interpretative descriptions of practice are required to create these methodologies.

Nandhakumar and Avison [7] use a case study to explore the application of a systems development methodology in a large scale executive information system (EIS) development project. They discover in practice that systems development methodologies are too prescriptive to be of use in the day-to-day activities of software development teams. In fact they find that the methodologies are used to present the necessary fiction of everything being under control within the project. They recommend systems development methodology researchers need to research supportive rather than prescriptive processes to assist systems development.

Wastell [11] uses clinical research, a form of action research with multiple cases, to explore a new theoretical perspective in software development processes research, a learning perspective. He contends that many IS failures are due to anti-learning mechanisms that develop due to the inherently stressful nature of software development. He suggests it is essential to create environments that are open and nurturing that will emphasize and promote learning. He suggests recasting software development into a transitional space and to use transitional objects within this process to increase the potential for problem solving learning behaviours, rather than problem avoidance behaviours. An example of a transitional object is bicycle training wheels: the wheels give us the support and comfort as we move from one state to another. Of particular interest to this study is his comment:

> Both IS professionals and users must engage in an intensive learning experience, the former to develop a thorough understanding of the business domain, the latter to reflect on current practices and to acquire an understanding of the potential of IT to transform the way the work is done. [11, p.582]

The study will need to consider the impact of XP principles and practices on the learning process, particularly that of the customer.
Chapter 2: Review of the Literature

There are significant groups, as reviewed above, who agree that the assumptions behind conventional methodologies need to be challenged and explored using rich descriptions of practice.

2.4 USERS IN THE SOFTWARE DEVELOPMENT PROCESS

In order to effectively operate in the world of vague and changing requirements, Beck [2] claims XP moves the emphasis away from prescriptive processes into practices that enable people. XP relies on effective face-to-face and regular communication within the XP team, particularly between the customer and the developers. Hence a small relevant section of the existing literature regarding user and development team interactions was explored. The XP Customer may or may not be an actual user, but they are expected to typically have a business background rather than an IT background, and so many of the issues and insights in this literature may be relevant.

Bostrom and Thomas [12] recognise the importance of effective communication between users and developers in developing computer systems that are on time, within budget and satisfy the needs of users. The authors used action research in a campus administration system to explore effective communication practices. The authors draw our attention to one of the ongoing issues in user and developer communications, the use of framing. They used an example to illustrate the concept of framing, this example is replicated below.

What is the ordering in the following numbers?
8, 5, 4, 9, 1, 6, 3, 2

Most of you were probably looking for a well-formed numeric sequence, a pattern of thinking installed since early childhood. If you were you probably had trouble answering the question since the ordering is alphabetic by first character of the name of the number (eight, five, four etc) [12, p.2]

This example highlights how our backgrounds and assumptions can cause us to frame the question in such a way as to “miss the point”. The study found the use of communication patterns was key to the development of effective communication between users and developers. People needed to be trained to ensure frames were (a) understood and (b) expanded by gentle verbal prodding.

Urquhart [14] used a multiple case research design to review issues in user-developer relationships during the early requirements gathering phases of software development. Three public agencies in Australia participated in the study. Six people were interviewed, representing three relationships, all participants were internal employees. She found that the development of positive working relationships, particularly between departments, is likely to have a large bearing on the success of the software development process.

McKeen, Guimaraes and Wetherbe [33] used a positivist survey research approach of 151 software development projects to discover the key determinants of user satisfaction with software development. The key findings from this research are:

- Effective user-developer communication directly affects users’ satisfaction with the system
- Users influence on the project also directly contributed to users’ satisfaction with the system
- User participation in highly complex systems development is worthwhile
- User participation in low complexity systems development may not be effective

The next two studies specifically concentrate on the role of the user in prototyping methods, particularly the evolutionary prototyping method. Gordon and Bieman [34] collected 39 prototyping case studies (22 published and 17 first hand accounts) and analyzed them to determine commonalities that led to effective use of the prototyping method. One of the areas explored in the study was the role of the user in the prototyping approach. The users were
typically able to participate in the process and explore and evolve their requirements as the system develops. This exploration opportunity however, could lead to users viewing incomplete functionality and becoming concerned about the quality of the end system. Alternatively it can set high expectations concerning the timeframe for delivery, particularly where a ‘smoke and mirrors’ prototype was developed. One of the limitations noted in the study was the ability to draw conclusive results from cases varying in rigor.

Baskerville and Stage [9] used action research to explore the software development technique of evolutionary prototyping. A small contact management system for a non-profit organization was the unit of analysis. Baskerville suggests one of the reasons for using evolutionary prototyping is that the technique emphasizes learning, and helps to facilitate meaningful communication between users and developers over concrete artefacts. The expected result is rapidly built flexible software. This study explored an approach to the planning of the evolutionary prototyping iterations and the approach was risk mitigation based. The approach described in this study had a number of elements in common with XP, including the short release cycles, building only what you need now, and the planning game technique (which ensures business functionality is delivered in business priority order).

The studies in this section highlight the importance of the users in the software development process. The communication and interactions between developers and users can significantly affect the outcome of a developed system. The prototyping studies reviewed explored the strengths and weaknesses of the users involvement in an incremental delivery approach. Note that one common misconception regarding XP is that XP is evolutionary prototyping. Ron Jeffries, one of the leading XP practitioners, in a personal communication [35] has explicitly stated that XP is not the same as evolutionary prototyping, suggesting that the practices of XP result in a more disciplined development approach than evolutionary prototyping.

The studies reviewed above provide valuable insights into the involvement of users in the process. However, the combination of XP practices is likely to change the dynamic of the user role and requires further exploration.

2.5 XP CASE STUDIES

XP is new. International conferences, including XP200x, and Agile 200x (which is the conference resulting from the merger of the XP Agile Universe 200x conference and the Agile Development Conference), are focussed on agile development and are establishing a body of knowledge concerning XP [15; 36]. Little research existed regarding the method, and particularly the customer role; however, this section outlines a few of the most relevant experiences available prior to the commencement of this study.

Gittins, Hope and Williams [16] used action research to explore the implementation of XP in a development team at a medium sized application (software) service provider. The company produced enhancements to the existing software every two to four weeks in response to feedback from customers, sales representatives, marketing executives and corporate management. One of the findings of this study was the importance of the implementation of a customer proxy. The term customer proxy refers to the person who sat with the development team as a substitute for the customer. Gittins et al. observed that the proxy in this study understood the customer’s needs in detail and always acted in the customer’s interests. The report does not clarify, however, how the customer proxy developed a detailed understanding of the customer’s needs or the attributes of the person undertaking this role.

Schalliol [18] provides a practitioner report on the experience encountered by a team of eight analysts on a 50-person, multiyear development project that converted to using an XP process. On this outsourced development project, the customer company devoted a team of its employees
full-time to the project. One of the issues reported was the need to handle diverse and competing user requirements inherent in the development of complex systems. The team added roles and procedures to XP that reduced the agility of the process, but allowed the team to handle this issue when it arose. The key aspect of the process change was the addition of a role to facilitate the communication and resolution of this issue. The team also introduced an issue card to allow the prioritisation and planning of this activity to occur within the XP planning processes. The report is a reflection on a real life experience, and does not contain the validity checks expected in case research.

Farell, Narang, Kapitan and Webber [19] also provide a practitioner report on the implementation of an effective on-site customer practice. This report considers the case of two analysts playing the customer role. The report describes the ineffective practices at the start of the project that included the customer being off-site and only meeting the development team once a week for a formal progress report. It then takes us through the journey that allowed an effective on-site customer to be available to the team. The conclusion was that the location of the team is important because customers and developers must work as a single team. Another important conclusion was the importance of feedback in XP. Testing was identified as an activity that must occur early for the project to succeed. Regular reviews of the process allowed the team to identify the issues and to implement correction strategies that improve the final outcome of the project. As with the previous report, this practitioner report does not contain the validity checks required for case research.

Deurson [5] has also begun to raise issues associated with the XP Customer Role. He noted that XP provided a “bag of tricks” for developers but little guidance for a customer to succeed. To begin to address this need, Deurson ran a workshop at XP2001 that aimed to collate a number of guidelines for successful customer involvement. The workshop group agreed that developing user stories with a “so that” section of the story assisted the customer to evolve their long-term goals of the system. The workshop group also agreed that the customer role typically cannot be held by one physical person, rather a team of people is required, termed the customer team. The customer team should include specialist expertise such as acceptance testing as well as domain area specialists, as it is difficult for one person to represent such a diversity of users, from system administrators to senior managers. Another customer team suggestion was for a programmer to pair with the customer, in order to provide effective support for the customer. It was noted that it would be essential for the customer pair to facilitate customer/programmer communication within the team rather than blocking this communication. As with the previous reports, this set of workshop papers does not contain the validity checks required for case research.

2.6 Other Agile Requirement Practices

Other agile methods such as Adaptive Software Development (ASD) [37] and Scrum [38] include more practices to address requirements gathering and analysis. It is interesting to consider the similarities and differences of these methods approach to deciding what to build to help place the XP model in context.

Scrum is often perceived as the project management agile method; an alternative perspective is that it has a very focused and minimalist approach to agile, which emphasises or aims to provide a mechanism to control the chaos surrounding the communications between the business and the project team. Scrum streamlines the requirements communication [38]:

- There is one central prioritised list of all requirements, the product backlog
- The product backlog is controlled by one person, the product owner, however many people contribute to the list
The sprint allows the project team to focus on a frozen set of requirements, while allowing the business to change their requirements after a tolerable delay.

The software is reviewed at the end of each sprint to ensure ongoing feedback to the project team and the business.

ASD [37] is based on complex adaptive theory and describes the mindset and principles required to succeed in developing software iteratively and incrementally. ASD recommends the following for requirements:

- There is a set of artefacts (project vision, data sheet and specification) that ensure a shared project vision exists.
- Collaboration techniques are used to evolve requirements: Joint Application Development (JAD) sessions, Customer Focus Groups (CFG) and finally post-mortems or process improvement reviews.
- A collaboration facilitator role is introduced to focus on “thinking about” and planning collaboration rather than simply “letting it happen”.

There is significant cross-over between these two methods and XP, including:

- The product backlog, user stories and product specifications all ensure there is a centralised list of requirements.
- The on-site customer role, the product owner role and the JAD/CFG sessions all aim to ensure the developers do not need to deal with conflicting requirements, including the prioritisation of requirements.
- The small and regular iterations ensure there is opportunity to learn and to evolve requirements.

So, while there is some difference in the details of the methods, many of the practices could potentially be interchanged. One of the key overall differences between XP and Scrum is their emphasis on different aspects of software development; XP emphasises programming, while Scrum emphasises project management. XP provides reinforcing practices for developers and Scrum provides practices for project managers. ASD concentrates on outlining learning and collaborative techniques and theories for all project members, and introduces a collaborative facilitator role that specialises in collaborative and learning based communication, including the communication between customers and developers. Both XP and Scrum appear to under-emphasise the challenging job facing the on-site customer and product owner and in gathering and prioritising the requirements for the project. ASD begins to address this issue by recognising the need for structured collaborative techniques and roles in this area.

2.7 Traditional Analysis and Requirements

The broad areas of requirements gathering and analysis cover the domain relevant to the XP customer. There are many texts in this area [39; 40; 41; 42; 43; 44; 45; 46; 47], the text of DeMarco is indicative. DeMarco, in his classic text *Structured Analysis and System Specification* [40], defines analysis as:

> Analysis is the study of a problem, prior to taking some action. In the specific domain of computer systems development, analysis refers to the study of some business area or application, usually leading to the specification of a new system. The action we are going to be taking later is the implementation of that system. [40, p. 5]

He discusses the need for a middleman or analyst to be involved in software development; the analyst is there to help to bridge the gap between the users and the development team. He highlights the differences between the programming and analysis activities. He notices that
programming is reasonably straightforward, friendly with minimal inter-personal relationships required, definite and satisfying, contrasted to analysis, which is none of these things. Analysis is not easy; it is frustrating, full of complex interpersonal relationships, difficult and indefinite (analysis often ends once the time in the plan or budget is up, as defining an end-point in any other way is exceedingly difficult). Most notably he compares analysis to “Kissinger negotiating for peace in the Middle East” [40, p. 5 - 7].

DeMarco outlines in his book a number of analytical diagramming and textual techniques that help analysts explore with users and developers the target domain, and what is more, communicate the intended end result of the computer system in a target document. The techniques include data flow diagrams, data dictionaries, structured English, decision tables, and decision trees. He supplements these techniques with a larger toolset such as:

- An early study of the user area (understanding the current environment)
- Using functional decomposition to help break down the problem or solution domain into a number of smaller areas, one of the notable advantages this breakdown provides is the ability to increase the number of people on the analyst team, so that more than one person can work on this activity
- Logical and physical modelling activities

Additionally, he also includes guidance for the analyst and implementation team on items such as acceptance testing, estimating and transitioning into implementation.

Although the lifecycle model he discusses is predominantly a waterfall model, he points out a strong need for feedback between lifecycle stages. Finally, while he is adamant that structured analysis assists us moving forward in this space, he does not attempt to suggest that it now makes it “easy”. He elaborates that analysis still remains a frustrating and difficult task that none the less is “fascinating”.

Object-Orientated analysis and design techniques brought a different way of breaking down or decomposing the problem and solution domain and communicating it. In the 1990’s these techniques standardised on the Unified Modelling Language (UML) [41; 45]. UML provides modelling and documentation techniques such as domain class models, use cases, sequence diagrams, package diagrams, state diagrams and activity diagrams. So, the decomposition models and diagramming techniques moved from the structured techniques outlined by DeMarco to object-orientated ones. However, as Fred Brooks notes, we are still left with the most difficult problem in software development:

*The hardest single part of building a software system is deciding precisely what to build ... Therefore, the most important function that software builders do for their clients is the iterative extraction and refinement of the product requirements. For the truth is, the clients do not know what questions must be answered, and they almost never have thought of the problem in the detail that it must be specified." [48, p. 199]*

Fred Brooks goes on to elaborate in his original essay and his updated 1995 essay that the hoped for silver bullets such as OO, software reuse, and modelling may have helped us improve our productivity but the heart of the problem remains the complexity of software development and defining what to build, and for this there is “no silver bullet”. As he indicates in the quote he believes the key to the complexity problem is iterative and incremental requirements definition.

However, along-side these modelling techniques were lifecycle or process changes too. One of the most notable ones is the Rational Unified Process (RUP) [41; 44], which was closely associated with UML. RUP emphasises an iterative and incremental process to developing software, and recommends or guides software development staff on the UML elements to utilise
at each stage within development to help decide what to build and then later transforming that into an architecture, design and implemented software.

We have touched upon the models we use, and the lifecycle or process that helps guide our use of those models. Another key area to consider is how do we work with users and other organisational stakeholders to develop that knowledge and the associated models. Again, there are many texts [41; 42; 46; 49; 50] that cover, in varying depths, interviewing, observation and workshop facilitation techniques. Gottesdiener [42] provides one of the most recent, in-depth and practical texts on requirements collaboration techniques. She covers practical workshop collaboration techniques that range from getting the right people into the room, preparing for a workshop (e.g. defining the goal of a workshop and an agenda with activities to help achieve that goal), facilitating the actual workshop, closing the workshop and following up after the workshop (e.g. documented output from the session). Her book provides practical guidance to a new requirements workshop facilitator; for example, one of the tips is to use spray mount. Spray mount is spray glue that turns any material into a post-it, which is highly useful when facilitating group discussions, allowing all participants to share in the collaborative note taking or modelling exercises being undertaken in the session. Perhaps most importantly she provides guidance on what models to use, when and who should attend workshops; helping us to fully utilise and apply the modelling techniques we have learnt elsewhere to the task of creating a common or shared understanding of both the problem and solution domain at hand.

Structured and object orientated analysis approaches provide us tools to help us study and communicate a domain. Each of them provides concrete diagramming and documentation techniques, a mental model or decomposition techniques to help us deal with the complexity of the systems we are developing. Additionally associated processes such as RUP provide us guidance on when and how to use these techniques on software projects. Finally, interviewing, observation and workshop facilitation techniques help us to apply these techniques and work with organisational representatives to define the problem and solution domains collaboratively. Given the XP Customer’s responsibility to set the direction or vision of the software the development team should be building, these techniques should assist the XP Customer to do that.
3. RESEARCH METHOD: GROUNDED THEORY

This chapter outlines grounded theory, the research method that was used to conduct this study. This chapter also specifically presents the research perspective, data collection and analysis procedures and the verification steps undertaken to increase validity.

3.1 WHAT IS GROUNDED THEORY?

Grounded Theory is a qualitative research method that emphasises [20; 51]:

- The importance of discovering or generating theory from data, particularly qualitative data from interviews and observation, over testing a theory.
- The importance of theory that can be understood and used by the people under study, in this study the XP Customer, as well as by researchers.
- The use of comparative data analysis to evolve theory; data collection, data analysis and theory evolution are tightly interwoven.

The creators of grounded theory, Glaser and Strauss, have over the years diverged [26; 52] as to the specific nature and practice of grounded theory. We will primarily utilise the original sources of knowledge concerning grounded theory [20; 51; 53]. However, we will utilise Glaser’s later works and opinions rather than Strauss’s to guide our interpretation of these earlier works.

3.2 WHY IS GROUNDED THEORY APPROPRIATE FOR THIS STUDY?

Grounded Theory is a qualitative research method. Creswell [21] provides a number of reasons to guide researchers as to the applicability of qualitative methods to their intended research. The following table outlines these reasons and their applicability to this research.

<table>
<thead>
<tr>
<th>Qualitative research selection reason</th>
<th>Applicability to this research</th>
</tr>
</thead>
<tbody>
<tr>
<td>The research question is exploratory and often starts with why or what. The intent is to describe what is going on. [21].</td>
<td>Our three research questions are formed with the intent of describing what is going on, concerning the day-to-day practice of the customer in XP projects.</td>
</tr>
<tr>
<td>The topic needs to be explored, that is, there is little existing literature or theory available to explain the participants or population of study. [21]</td>
<td>As outlined in our literature review in chapter 2, there is little understanding or guidance concerning the XP customer role.</td>
</tr>
<tr>
<td>There is a need to present a detailed view of the topic, as a wide-angle view will not provide the answers necessary, or a close-up view does not exist. [21]</td>
<td>The intent of this research was to provide the rich descriptions of existing practitioner practices, as recommended by systems development methodology researchers [6; 7; 8; 54; 55]. Rich or close-up views are perceived as the way forward in this research area.</td>
</tr>
<tr>
<td>Phenomenon of interest cannot be studied outside its natural setting</td>
<td>The complex social interactions and processes [8] inherent in a software development project make it difficult to replicate outside of its natural setting without the loss of the richness of the activity.</td>
</tr>
</tbody>
</table>

We have adopted Grounded Theory, a qualitative method, for this research. Grounded theory stands out from the other qualitative methods such as case research and ethnography. Grounded
theory not only richly describes practice but also generates theory that will allow practitioners to understand their situation and learn how to alter it to achieve a desired outcome. Therefore grounded theory was the qualitative method we selected to assist us to answer our research questions.

### 3.3 **Which Research Perspective Will Be Used?**

Research perspectives\(^5\), such as positivist, interpretative and critical provide a researcher with multiple methods of viewing a research problem or topic. Each of these perspectives has its own set of assumptions that result in strengths and weaknesses of the research design approach. Orlikowski et al. [55] challenge the dominance of the positivist perspective in Information Systems (IS) research and suggest viewing the world through one lens is unnecessarily restrictive. There is a growing interest in the use of the interpretative perspective in both IS research generally [56] and IS research methodology research specifically [6; 7; 8; 54].

Orlikowski et al. [55] note the interpretative perspective is most appropriately used to capture complex and dynamic social processes and quote Gibbons:

> The interpretative perspective attempts to understand the intersubjective meanings embedded in social life ... [and hence] to explain why people act the way they do (p. 13)

The interpretative perspective assumes people are inherently complex and that group interactions are complex social practices. It assumes the researcher interprets or shares in these meanings. These assumptions allow the interpretative perspective to explore complex social processes, without attempting to simplify these processes into cause and effect laws of nature.

Orlikowski et al. [55] go on to critique the interpretative perspective. Their critique includes the following deficiencies:

- External conditions, including history, are not examined and could give rise to certain meanings and consequences.
- Conflict and contradictions within the social group are not explored.

One of the research methods available for use within the interpretative perspective is Grounded Theory.

### 3.4 **Who is the Researcher (and why it matters)**

In the interpretative perspective it is assumed the researcher will interpret the data based on their values and beliefs. It is generally accepted that our values and beliefs are shaped by our life experiences [57] and researchers need to provide relevant experiences as part of a piece of research.

It should be noted that in Grounded Theory personal and professional experience is considered a source of theoretical sensitivity. Strauss and Corbin [58] recognise the advantages of professional experience within the domain under study:

> “This knowledge even if implicit, is taken into the research situation and helps you to understand events and actions seen and heard, and to do so more quickly than if you did not bring this background into the research ... the more

---

\(^5\) For the purposes of this thesis, paradigm and perspective are treated as separate concepts. The terms qualitative and quantitative (paradigms) are used to refer to the type of data collected and associated collection methods, not the assumptions inherent in the perspectives (positivist, interpretative and critical) outlined by Orlikowski et al.
professional experience, the richer the knowledge base and insight available to draw upon in the research." [58, p. 42]

Strauss and Corbin go on to elaborate that there are some disadvantages to this experience as well, most notably, that it can prevent the researcher from seeing the “obvious” or “routine”. However, the comparative data analysis techniques outlined in the Grounded Theory literature [20; 51] assists to mitigate these disadvantages. Additionally a number of published Grounded Theories [59; 60] have utilised the professional experiences of the researcher to help compare and contrast collected data.

Hence the following section provides some of my background that is most likely to affect the interpretation of the data. I completed a Bachelors of Computing and Mathematical Sciences at The University of Waikato, majoring in Information Systems in 1993. During the last fourteen years of my professional career I have specialised in outsourced custom development projects in both New Zealand and the United Kingdom; these projects have covered a wide range of software development methods, from the Software Capability Maturity Model (SW-CMM) [61], Rational Unified Process (RUP) [44], Agile, “waterfall” and “code and fix”. I have worked for both the client and the vendor in outsourced arrangements. I have undertaken a variety of project roles including: team lead, information architect, business analyst, systems analyst and developer. Organisations I have worked for include: Innovus, Logica, Synergy, AMR and Associates, ThoughtWorks, Fletchers, Contact Energy, Department of Work and Income, Department of Child, Youth and Family, Insurance Australia Group (IAG), Ministry of Health, Lloyds TSB Insurance, Guardian, Coda, Sky, Leica Microsystems, and Gazprom Marketing and Trading.

3.5 **How will we use literature?**

In a Grounded Theory study, literature is used inductively [20; 21; 58]. That is, the literature is typically presented at the end of the study, as it is used to compare and contrast the findings of the study. Strauss and Corbin [58] outline the reasoning behind this approach as follows:

> “There is no need to review all of the literature beforehand (as is frequently done by researchers trained in other approaches), because if we are effective in our analysis, then new categories will emerge that neither we, nor anyone else, had thought about previously. We do not want to be so steeped in the literature as to be constrained and even stifled in terms of creative efforts by our knowledge of it! Since discovery is our purpose, we do not have beforehand knowledge of all the categories relevant to our theory. It is only after a category has emerged as pertinent that we might want to go back to the technical literature to determine if this category is there, and if so what other researchers have said about it”

[58, p. 50]

Additionally, Glaser [51] notes that literature should also be used by the research analyst to increase their theoretical sensitivity. That is, literature outside of the substantive area of study, particularly well respected grounded theory research in other disciplines, should be read so that the research analyst becomes sensitive to the potential core categories and how the variables relate to these categories.

We have used literature during this research in this manner, with the previous chapter providing an overview and context to our research, and the later chapters where we present our analysis will use literature inductively.
3.6 HOW WE GAINED ACCESS TO XP PROJECTS

Benbasat et al. [56] recommend the two key points a researcher should cover while seeking initial co-operation are confidentiality and benefits. The person responsible for each of the projects studied was approached with these points in mind, and the opportunity for research was discussed and a proposal for the research was presented (please refer to the appendices for the generic proposal). This proposal clearly outlined the agreed outcomes (benefits), scope, and approach to the research project with the company. Once organisational agreement was obtained, the project team were approached to determine their interest in participating in the study.

Participants were informed of the outcomes of the project, and were assured all information would remain confidential, and all results will be reported so as to retain their anonymity. The information provided to the team members is included in the appendices, and outlines the importance of confidentiality and the agreed rules.

The key benefit arising from the research was the opportunity for the organisation(s) and team members to learn and reflect on their existing practice. This learning opportunity provided the organisation(s) and team members with the learning experience more typically obtained via an implementation review. A second opportunity that was offered was the opportunity to be seen as a practice leader, if the organisations chose to be identified in any articles or conference proceedings published. No organisations took-up the second opportunity, in that all organisations retain pseudonyms within this research.

The costs to the organisation(s) were minimal and time-based only. The total time required of each of the participants was between 2 to 10 hours over the course of 1 – 2 years, depending on how active the participant chose to be in this research. The maximum period in any one day for any team member was 2 hours. All interactions were arranged around the team members’ existing commitments, so disruption was minimal.

Locke [52] notes that one of the particular issues with Grounded Theory is the ongoing access to the subjects under study, as with Grounded Theory the researcher will tend to have significant periods off-site doing data analysis prior to returning to collect more data. In this study this difficulty was further emphasised as we were often located in a different country and so further visits were up to a year in the future.

The strategy to combat this issue, as presented by Locke, is to develop and maintain relationships with the participants. We maintained relationships with the key participants within each project, typically the customer or project manager as well as with particularly interested participants. We maintained these relationships with regular communication, including email summaries of research findings to date as well as using any opportunities to use more direct communication such as phone calls or face-to-face catch-ups that arose due to conference attendance or other such activities. For example, some participants from our initial cases have used our experience or networks to help them write papers for conferences, or to become involved in workshops or panel discussions at conferences.

The projects under study evolved over this period as well, some moved into a “maintenance” mode while others remained actively under development, resulting in different opportunities for data collection. We continued to use our network of conference contacts to locate suitable projects for our data collection focus at any given time.

This project obtained Victoria University of Wellington human ethics approval; please refer to the appendices for a copy of the approved application.
3.7 **HOW WE COLLECTED THE DATA**

Creswell [21] and Glaser et al. [20] outline the typical data collection and associated analysis techniques of grounded theory. The data collection techniques include interviews, observations, documentation analysis, physical artefact analysis and visual image capture. The data collection and analysis techniques used in this research project fall within this typical list, and are detailed below. But first we outline our intended role in this piece of research.

3.7.1 **RESEARCHER ROLE**

Lee [62] and Creswell [57] note researchers typically distinguish between four types of research participation in qualitative studies. The key elements that distinguish these types of participation are summarised in the following table.

<table>
<thead>
<tr>
<th>Research participation classification</th>
<th>Research role</th>
<th>Primary focus</th>
<th>Participation in environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete participant</td>
<td>Hidden</td>
<td>Research and Participation</td>
<td>Active</td>
</tr>
<tr>
<td>Participant as observer</td>
<td>Public</td>
<td>Research and Participation</td>
<td>Active</td>
</tr>
<tr>
<td>Observer as participant</td>
<td>Public</td>
<td>Participation</td>
<td>Active</td>
</tr>
<tr>
<td>Complete observer</td>
<td>Public</td>
<td>Research</td>
<td>Passive</td>
</tr>
</tbody>
</table>

For all of the case studies outlined in this research we were complete observers, and took a passive role in the environment. We did notice, despite our passive role, that the act of asking questions and observing teams did have some impact to the environment we were studying. For example, one interviewee realised during the interview as he put pieces together and told his story to us that the reason he had been unable to do a certain activity was because he was already working 60+ hours a week on the project, and so was able to “forgive himself” for not undertaking that activity.

However, we also undertook some activities where we were also direct participants. Our industry-based roles on agile projects as either a business analyst or agile coach provided a grounding and data source for the theory developed and outlined in this thesis. Additionally, we presented the theory outlined in this thesis to over 300 practitioners and academics in the agile community in the form of a popular conference tutorial. During these presentations we were able to gauge the effectiveness and fit of the theory to practitioners. All of these participation-based activities helped to evolve the theory into the state it is today.

3.7.2 **SEMI-STRUCTURED ONE-ON-ONE INTERVIEWS**

Our main data collection method was semi-structured one-on-one interviews. The XP team consists of the following roles: customer, programmer, tester, coach and big boss [2]. Each of these roles will have a view on the implementation of the XP customer role:

<table>
<thead>
<tr>
<th>Role</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>The team member fulfilling this role is the most intimately involved in the implementation of the XP on-site customer requirement. It is imperative their views and experiences are explored.</td>
</tr>
<tr>
<td>Programmer</td>
<td>One of the key tenants [2] in XP is the use of effective face-to-face communication between the customer and the programmers.</td>
</tr>
</tbody>
</table>
Chapter 3: Research Method: Grounded Theory

<table>
<thead>
<tr>
<th>Role</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tester</td>
<td>The tester in XP focuses on helping the customer to choose and write</td>
</tr>
<tr>
<td></td>
<td>functional tests, which provide feedback into the planning process.</td>
</tr>
<tr>
<td>Coach</td>
<td>The coach [2] is responsible for the process as a whole. They work with</td>
</tr>
<tr>
<td></td>
<td>each of the other roles to help facilitate the correct use of XP. The</td>
</tr>
<tr>
<td></td>
<td>person fulfilling this role will provide valuable insight into the</td>
</tr>
<tr>
<td></td>
<td>adoption of XP.</td>
</tr>
<tr>
<td>Big Boss</td>
<td>The big boss, often known as the project sponsor, is responsible for</td>
</tr>
<tr>
<td></td>
<td>the project. The big boss will have a unique perspective on the</td>
</tr>
<tr>
<td></td>
<td>customer role, particularly how competing customer requirements and</td>
</tr>
<tr>
<td></td>
<td>needs were prioritised and whether that was acceptable to the</td>
</tr>
<tr>
<td></td>
<td>organisation as a whole.</td>
</tr>
</tbody>
</table>

The interviewees covered the spectrum of the required roles, over the projects studied. Not all roles were interviewed for each project, however we attempted to gain as many different perspectives of the implementation of the customer role as possible in each setting, and over the course of the research all roles were interviewed.

The interviews allowed us to explore the team member’s view of the XP process, particularly the practices and processes that directly and indirectly impacted the customer. The topics covered included the background of the interviewee, the planning process and their view of any perceived issues and effective experiences encountered on the project. An example of the interview outline used is presented in the appendix.

The interviews typically occurred in a meeting room at the site of the interviewee. The on-site location helped to minimise the disturbance to the interviewee’s day-to-day activities. The meeting room also helped to ensure the interviewee was able to speak freely without concern for other team members’ perceptions of their responses. When a meeting room at the site of the interviewee was not available then a suggested alternative was negotiated with the affected interviewee(s). One example of an alternative used was a quiet café near the interviewee’s location. All interviews were taped and later transcribed in detail. The interviewees were asked to validate both the transcription of the interview and the interpreted findings.

66 people were interviewed; almost all interviews were over an hour in duration. We had a number of follow-up interviews with some people and so a number of people spent over 2 hours with us.

3.7.3 Observation

The interviews were the primary source of data for this research, with observations adding another dimension to the data. The physical working environment and atmosphere was observed, including the tone and content of the customer interactions with other team members. Where possible we also attended planning game sessions or other specific interactions. These observations were used to verify the interview findings. Additionally they provided some insights that were investigated further in follow-up interviews, particularly if the interviewee did not initially bring up the point we noted.

Nine of the eleven projects studied were observed. Most of the projects were observed for only a few hours, but two of the projects were observed over the course of a 3-day period.

3.7.4 Document Review and Analysis

Documents specifically produced or used by the customer, including user stories and plans, were requested and reviewed, as well as any others that were identified by the interviewee as being of
use to this research. These documents aided our understanding of the artefacts produced, in line with the XP suggestions.

These documents were internal project deliverables and as such are of a commercially sensitive nature. To ensure confidentiality we do not show specific examples, just the interpreted point. The team members were requested to validate all findings regarding the documentation.

3.7.5 EXPLORATORY ACTION

In parallel to the other data collection activities for the case studies, we continued to work in industry on agile projects. This provided us with the opportunity to apply the theory that evolved from this research. The day-to-day experience of applying the theory helped to validate and refine the resulting theory, providing more context and examples of use. Additionally, during the last eighteen months of the research study, we moved into the specialised role of an agile coach. As an agile coach we worked with teams, particularly customer teams, to transition to agile, and to undertake their role effectively on projects. The teams were able to utilise the theory outlined in this thesis to help in that transition. This experience assisted in the verification and refinement of the theory. For example, it led to the identification that one of the role names did not work for some team members, the concept behind the role was agreed but the name caused some confusion.

We worked full-time for the six-year duration of this study. However, the last four years were specialised on agile projects, allowing this exploratory action to occur. The last eighteen-months of the study were focused on agile coaching activities, so actively using the theory that had evolved in practice to help transform the XP customer role.

3.8 HOW WE ANALYSED THE DATA

The interview data was transcribed from the tape recordings by a professional transcriber. We then reviewed the transcripts and observation notes line-by-line\(^6\) and created short summary labels for each significant line of the transcripts. This process is called open coding. Open coding began almost immediately as we used this process to determine the next “sample” to take. For example, we may realise a particular question needs to be asked in a follow-up interview or we may need a document to review.

We used comparative data analysis to compare and contrast the open codes, to develop the substantive codes (an abstracted category that fits multiple open codes). We also wrote theoretical memos that relate the codes, drawing out the potential ideas or concepts held within the data. This memoing process assisted the research analyst to focus on the core categories and the inter-relationships between the categories.

This analysis process was iterative and incremental. The term theoretical saturation is used to denote when to stop this process, that is, when no or very few new codes are discovered within the data. Thus, the data collection and analysis phase stopped as theoretical saturation was reached, which occurred upon the completion of the eleventh case study.

We initially coded the transcripts within the margins and used index cards for the memos, as recommended by Glaser [51]. This process provided a tactile method of undertaking the theoretical sorting process. However, as the research progressed this method became unmanageable given the number of transcripts and memos, so we used the software package HyperResearch (version 2.7) [63] to assist with the coding and sorting process. Printed reports

\(^6\) As theoretical saturation is almost reached, the transcripts may begin to be analysed at a paragraph level.
from the package were then used along with the original coded transcripts to continue the theoretical sorting process.

To give the reader some idea of the number of codes generated during data analysis we can provide the statistics for one of the cases (OwlCorp). For this case, we created 547 open codes, and there were 1,250 instances of these codes across the transcripts, resulting in twenty-five substantive categories.

### 3.9 **How we Verified the Theory?**

Creswell [57] notes that internal validity ensures the information is accurate and it matches ‘reality’. Internal validity, as summarised by Creswell, was achieved as follows.

<table>
<thead>
<tr>
<th>Recommended procedure</th>
<th>Implementation in this study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessing multiple sources of information</td>
<td>Multiple data collection methods were employed including interviews, observations and document analysis. Multiple people on the same project were interviewed.</td>
</tr>
<tr>
<td>Auditing the decisions and interpretations given the raw data, by another researcher</td>
<td>PhD thesis advisors reviewed the majority of transcripts and all of the interpreted findings.</td>
</tr>
<tr>
<td>Obtaining feedback from the participants on both the raw and interpreted data to confirm the data has been interpreted accurately</td>
<td>Participants were given the opportunity to review the raw and interpreted data. In each case study at least one of the participants did indeed take up this opportunity and provided feedback.</td>
</tr>
</tbody>
</table>

Creswell [57] notes reliability as the last verification step. Reliability is met when sufficient information to allow another researcher to replicate the study is provided. All information expected to enable this replication to occur has been provided, including researcher biases, data collection and data analysis procedures. Note that no raw data (e.g. interview recordings, transcripts or project documentation) has been provided as this data must remain confidential, but a significant number of anonymised quotes from the interviews have been provided to ground the interpretation for the reader.

Additionally, and perhaps more importantly, Glaser and Strauss [20] draw our attention to the importance of the credibility of a Grounded Theory. Glaser [51] describes a Grounded Theory as having “grab”, it is interesting, people remember it and more importantly use it. He suggests a number of criteria that a Grounded Theory must meet in order to achieve the “grab” quality. A Grounded Theory must:

- **Fit** the data. The reader will be able to assess from the write-up of the theory that the categories have emerged from the data; the data was not forced by pre-conceived ideas, categories or category inter-relationships.
- **Be relevant.** That is, the practitioners in the substantive area under study consider the theory to be useful and meaningful. This is achieved by allowing the core problems and categories to emerge from the data.
- **Work.** The theory should explain what happened, predict what will happen and interpret what is happening, both for practitioners and researchers.
- **Be readily modifiable.** That is, the researcher presents the theory in such a way that it can continue to be evolved by other researchers.
All Grounded Theories should be assessed as to their credibility as outlined above. During the last three years of our six-year study we have presented our research to well over 300 practitioners and academics in the form of a conference tutorial at a number of conferences including: OOPSLA 2005, XP2006, Agile 2006, OOPSLA 2006, Agile 2007, and XP Day 2007. The feedback from this three-hour tutorial has given us an insight into the “grab” of the theory from the perspective of the attendees. Feedback from attendees is positive, with many finding it helped provide them with a much deeper insight into the experience of the customer. Additionally feedback from attendees has helped us evolve our theory so that it has more grab. For example, the names of practices or roles has evolved over the course of this study, as attendees were able to identify that while the concept and the examples presented to illustrate the concept made sense the name was not memorable enough, or conjured up a different concept to what was intended. The resulting theory is much richer and has more “grab”.

This chapter has outlined our research method for this study. The next chapter will provide an overview of all of the cases studied with later chapters outlining the theory that evolved over the course of the study.
Chapter 4: Overview of the Cases

4. OVERVIEW OF THE CASES

The eleven case studies we present in this chapter were selected opportunistically. We utilised our networks within the agile community, working with speakers from the agile conferences to help us select the projects for the case studies. One advantage of this selection approach is that it helped to ensure that we targeted projects that were practicing XP well.

The case studies were collected in four separate data collection phases. The first phase was our initial inquiry into the subject area, and it provided the initial insights into the on-site customer and helped to guide the refinement of our research questions and the next data collection phase. The second phase of data collection continued our deep exploration of the experience of the on-site customer practice. During this phase we specifically targeted a range of different project types, including those in software product development, internal and outsourced projects over a variety of domains and locations within North America. The third phase of our data collection occurred when we had uncovered the initial working model and we focused our attention on exploring different aspects of that model within these case studies, as well as broadening the companies covered into Europe and the UK. In our final phase of data collection we were specifically targeting a project where the on-site customer was not overloaded to further compare and contrast our findings and the theory developing from the first three phases of data collection.

The projects studied came from a variety of companies, from small start-ups to Top 100 companies. A high-level breakdown of the companies involved in the study is provided below:

<table>
<thead>
<tr>
<th>Data Collection Phase</th>
<th>Project (alias)</th>
<th>Company (alias)</th>
<th>Location</th>
<th>Domain</th>
<th>Size</th>
<th>Team</th>
<th>People</th>
<th>Lines</th>
<th>Words</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>Endeavour</td>
<td>KiwiCorp</td>
<td>Australasia</td>
<td>Telecoms</td>
<td>M</td>
<td>11</td>
<td>5</td>
<td>6,600</td>
<td>73,108</td>
</tr>
<tr>
<td>II</td>
<td>Discovery</td>
<td>RavenCorp</td>
<td>USA</td>
<td>Research</td>
<td>S</td>
<td>20</td>
<td>6</td>
<td>6,366</td>
<td>76,835</td>
</tr>
<tr>
<td></td>
<td>Atlantis</td>
<td>EagleCorp</td>
<td>USA</td>
<td>Software Tools</td>
<td>S</td>
<td>16</td>
<td>9</td>
<td>12,711</td>
<td>152,175</td>
</tr>
<tr>
<td></td>
<td>Pinta</td>
<td>FalconCorp</td>
<td>USA</td>
<td>Retail Products</td>
<td>S</td>
<td>60</td>
<td>7</td>
<td>5,809</td>
<td>67,602</td>
</tr>
<tr>
<td></td>
<td>Speedwell</td>
<td>SwiftCorp</td>
<td>USA</td>
<td>Retail</td>
<td>XL</td>
<td>20</td>
<td>9,997</td>
<td>117,298</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Endurance</td>
<td>HawkCorp</td>
<td>USA</td>
<td>Knowledge</td>
<td>S</td>
<td>5</td>
<td>6</td>
<td>3,708</td>
<td>43,145</td>
</tr>
<tr>
<td></td>
<td>Advance</td>
<td>HawkCorp</td>
<td>USA</td>
<td>Knowledge</td>
<td>S</td>
<td>5</td>
<td>6</td>
<td>3,552</td>
<td>40,023</td>
</tr>
<tr>
<td></td>
<td>Endurance</td>
<td>HawkCorp</td>
<td>Europe</td>
<td>Telecoms</td>
<td>M</td>
<td>20</td>
<td>7</td>
<td>3,113</td>
<td>37,004</td>
</tr>
<tr>
<td></td>
<td>Aurora</td>
<td>SparrowCorp</td>
<td>UK</td>
<td>Energy</td>
<td>L</td>
<td>15</td>
<td>4</td>
<td>2,147</td>
<td>24,915</td>
</tr>
<tr>
<td></td>
<td>Godspeed</td>
<td>KiteCorp</td>
<td>Europe</td>
<td>Transport</td>
<td>M</td>
<td>12</td>
<td>5</td>
<td>12,527</td>
<td>143,803</td>
</tr>
<tr>
<td></td>
<td>Dolphin</td>
<td>RobinCorp</td>
<td>UK</td>
<td>Mobile software</td>
<td>XS</td>
<td>6</td>
<td>3</td>
<td>12,527</td>
<td>143,803</td>
</tr>
<tr>
<td></td>
<td>Calypso</td>
<td>OwlCorp</td>
<td>UK</td>
<td>Health</td>
<td>L</td>
<td>20</td>
<td>9</td>
<td>69,118</td>
<td>810,504</td>
</tr>
</tbody>
</table>

7 One company, for legal reasons unrelated to our research, decided to withdraw from the study after the interviews were complete. All data collected was destroyed and was not used in this thesis.
8 All projects have been named after exploration or research ships.
9 All companies have been named after birds.
10 Companies were classified by the number of employees they employed: XS < 15 people, S < 1,000 people, M < 10,000 people, L < 100,000 people and XL 100,000+ people.
11 Team sizes varied throughout the life of the project, these numbers are typically the peak numbers. For more detail please refer to the individual case descriptions within this chapter.
12 The number of people interviewed concerning each project. One person was interviewed concerning two projects, and they have only been counted once in the total count.
From each of these companies, one agile project was studied. These projects covered a mix of technologies including internal applications, software products, web and mobile developments. We provide a high level overview of all of the projects in the following sections of this chapter. Note that the first five projects and the last project have a more in-depth description in the next chapter; we treat these projects as our primary case studies, and the remaining projects as supplementary case studies.

4.1 Project Endeavour, KiwiCorp

Project Endeavour was an outsourced custom development project, the purpose of the project was to build a content management system. The project involves three organisations:

- KiwiCorp, a large New Zealand company, was the customer organisation
- DevCorp, a large consultancy, was the development vendor
- BureauCorp, a large software services company, was the infrastructure vendor

Project Endeavour was seen as a success by both KiwiCorp and DevCorp, and part of that success was attributed to the use of XP. KiwiCorp had attempted project Endeavour twice previously, but each previous attempt was unsuccessful. The project was approximately fifteen months in duration, and at its peak the project team had eleven full-time experienced team members.

All of the team members were new to XP. DevCorp had recently used XP successfully on a similar project. However, none of the team members on the earlier XP project transitioned to this project. There was strong support in the team for adopting XP; the team gained an understanding of XP by reading the XP books, sharing XP experiences with other practitioners of XP and also by regularly reviewing their progress with the method throughout the project.

The project team was at the DevCorp building, approximately a 20-minute walk from where the KiwiCorp customer was located. The team area was open, brightly coloured and had a number of humorous pictures scattered throughout the visible planning and tracking charts. The customer attempted to spend one to three days a week with the team at this location, depending upon her other commitments.

We conducted five 1-hour interviews with both KiwiCorp and DevCorp project participants, including:

- From KiwiCorp: the Customer
- From DevCorp: the project manager, the team lead, a developer and the account manager.

The interviews were all one-on-one semi-structured interviews, and occurred near the completion of the project [32].

4.2 Project Discovery, RavenCorp

RavenCorp was a United States based start-up company. RavenCorp was formed in mid 2000, with the intention to move a piece of scientific software developed in an academic environment into a commercial environment; this software was codenamed Discovery. Discovery was to be their key product and at the time of the interviews was still under development\(^\text{13}\) with a team of eight staff, although at its peak there were almost twenty people working on Discovery.

\(^{13}\) Since the interviews the product has been released, and sometime later (years) another company purchased RavenCorp. The product was still part of the purchasing companies product suite at the end of 2007.
Discovery, a piece of scientific software, was comprised of two main components, a complex graphical user interface and a set of complex scientific computational algorithms. The software product was due to be released in the first quarter of 2004 and at the time of the interviews was on track for completion. Initial external client feedback, based on product demonstrations, was positive.

The founders of RavenCorp were not software developers, and although they had significant experience in developing simulation based programs for scientific research, they had very little formal software background or experience. RavenCorp recognised, as the focus moved from proving the concept to building a commercial product, that a software development process would be beneficial to the development of a quality commercial product. At the end of the first quarter of 2002 RavenCorp decided to move to XP. The founding members of the organization made this decision. RavenCorp followed what they considered to be best practice guidelines in moving to XP, including bringing an experienced XP coach full-time onto the project.

RavenCorp created a specific XP working area for Project Discovery, bringing all of the team members together in one area. All of the team also retained their individual offices as well as the communal working area; however, the team seldom use their offices. The environment was friendly with a pleasant buzz and had a traffic light to show the state of the current build, a wall given over to progress charts and story cards, and a collection of toys for the programmers. A project wiki also existed which provided external visibility as well as a virtual communal workspace. The RavenCorp senior management worked closely with the project team and as such, a trusted and productive working environment appeared to exist.

We conducted six 1-hour interviews with Project Discovery team members, including interviews with the coach, the customer, a scientist, the tester and two developers. The interviews were all one-on-one semi-structured interviews, and occurred near the completion of their first release of the software product.

4.3 **PROJECT ATLANTIS, EAGLECORP**

Project Atlantis was a successful project undertaken by EagleCorp. EagleCorp is a software company in the United States that specialized in products that support application development and application development management.

The project started in mid 2002, with the intent of creating the next-generation of EagleCorp’s flagship product, Atlantis 7.0. Atlantis 7.0 was a complete re-development of the existing product, Atlantis 6.x, and moved the product onto a new technical architecture. The interviews to collect the data for this case study occurred just as Atlantis 7.0 was moving into internal Quality Assurance (QA). At the time of the interviews the project had been underway for approximately eighteen months and had sixteen experienced full time staff. All indicators were that the project was considered a success\(^\text{14}\), particularly with regard to the quality of the product.

Atlantis 7.0 was EagleCorp’s first XP project. The decision to move to XP was made after the architecture and the product feature set had been defined, approximately six months into the project. EagleCorp followed what they considered to be best practice guidelines in moving to XP, including the use of a local XP mentoring company, for both initial training and ongoing advice.

EagleCorp created a specific XP joint working area for Project Atlantis. The managers moved out of their offices and the programmers moved out of their cubicles into this collaborative

\(^{14}\) Some time (years) after the interviews the Atlantis product was purchased by another company, in 2008 the product remained a core product offering at that company
working area. The working area contained desks on wheels and whiteboards. The desks on wheels allowed the programmers to personalise their working environment as well as allowing the programmers to easily re-orientate their work area to work collaboratively in pairs. The EagleCorp executive team endorsed these changes, which were not standard EagleCorp practice, as well as supporting these changes financially.

Another aspect of the environment was that the responsibility for the product definition, quality and implementation was clearly divided across three line managers: a product manager, a quality assurance manager and an engineering manager respectively. It was also evident that the executive team trusted each of these managers. The direct line managers also placed this trust in each team member. For example, no one was placed on the team who was not acceptable to each team member. Any team member was able to state they would not work with the proposed individual and no reasons were required. A strong team culture was in evidence, particularly within the engineering component of the team. A standard practice was for the team to have lunch together at noon, and other less regular social events also occurred, such as trips to the movies.

We conducted nine 1-hour interviews with Project Atlantis team members, including interviews with the project manager, vice-president of product development, architect, technical writer, testing manager, two developers, the product manager and the user interaction designer. Additionally we conducted three 1-hour follow-up interviews with the product manager, user interaction designer and architect. The interviews were all one-on-one semi-structured interviews, and occurred near the completion of their first release of the software product. Finally, the team was observed over the course of three days during an iteration.

4.4 PROJECT PINTA, FALCONCORP

Project Pinta was an outsourced XP project involving:

- **RCCorp**, a large organisation based in Europe, is the customer organisation
- **ManageCorp**, an international management consulting, technology services and outsourcing company, is the outsourced vendor for all of RCCorp’s IT functions and plays the role of proxy customer
- **FalconCorp**, a large international software product company based in the United States of America, is the development vendor.

Project Pinta was not a typical custom build project, as the intent of this project was to develop a product that will both meet the needs of RCCorp and become a product within FalconCorp’s product suite. FalconCorp has not seen project Pinta as a success, and part of the concerns with the project has been the use of an agile development method. The project had taken longer than the original bid schedule, currently more than three times the original bid estimate, and at the time of the interviews, had yet to be accepted by RCCorp. The project can be divided into two stages, Stage I covers the original bid estimated period and Stage II covers the remaining time.

The team had less than six months to cover almost 30 functional areas, so management had quickly scaled the project up to more than 60 people across several XP labs. The labs were structured so that each lab had a customer representative, two of the labs had product managers from FalconCorp and two of the labs had business analysts from the European office of ManageCorp. The ManageCorp analysts had been involved in writing the initial high-level requirements that had been accepted by RCCorp. No representatives from RCCorp were on the project.

The team had quickly established a pace with weekly iterations. At the end of each week the team produced working software that implemented the prioritised stories. It had quickly become
obvious that all of the functionality would not be completed by the required deadline, despite everyone’s best efforts. However, FalconCorp had committed to deliver the functionality by this deadline, and so the project team experienced a lot of dysfunction and pressure, with many issues being left unresolved in order to achieve their deadline.

The demonstration to RCCorp went well, so well that FalconCorp laid off over two thirds of the original staff, as they were now entering “bug-fixing” mode. During the first two months of Stage II, new project management was appointed in FalconCorp. These new managers quickly assessed that the project required significantly more work than “bug-fixing”. The final decision was to treat Stage I as a throw-away prototype and re-write the entire product. The contract was renegotiated and at the time of the interviews FalconCorp expected to complete the project in mid 2004.

The data that grounds this section was collected in a series of seven 1-hour interviews with FalconCorp interviewees. The interviews included: the coach, developers and business analysts from FalconCorp only, no ManageCorp or RCCorp people were interviewed.

4.5 **PROJECT SPEEDWELL, SWIFTCORP**

Project Speedwell was an internal software development project at SwiftCorp. SwiftCorp is a very large North American corporation, one of the leaders in the retail industry. The purpose of this project was to rebuild one of their internal “bread and butter” operational systems. This project crossed borders with part of the team in the United States and part of the team in Canada. Additionally the team of approximately twenty people was a mix of internal software development staff and external consultants or contractors.

Project Speedwell was the second XP project SwiftCorp had undertaken. The first pilot project had recently completed successfully and SwiftCorp were in the process of rolling-out XP to more projects. The initial team from the first project, including their coach, were rolled onto a related programme of work. The project had large working areas in the United States and Canada. We observed the working area in the United States, and it included story boards, and a progress tracking meter called the geig-o-meter, which was devised by the team, and was the basis of an “in-joke” concerning its name. Video conferencing facilities were readily available to allow both of the teams to communicate regularly.

Speedwell was focusing on replacing one of the core selling systems across all of the SwiftCorp’s stores. Speedwell had been under way more than six months when we visited the project. At the time, the project was perceived as on-track but some of the interviews suggested there might be some issues up ahead. A few months after our time with the team the project was cancelled. A brief follow-up email exchange with the customer coach occurred regarding the reasons for cancellation. The decision flowed out of SwiftCorp’s decision to outsource all software development to an external software development organisation.

We conducted seven 1-hour interviews with Project Speedwell team members, including interviews with the IT project manager, the customer coach, the developer coach, the on-site customer, and three developers. Additionally we conducted two 1-hour follow-up interviews with the customer coach. The interviews were all one-on-one semi-structured interviews, and occurred in the middle of their project. Finally, the team was observed during an iteration planning game.

4.6 **PROJECT ADVANCE, HAWKCORP**

HawkCorp is based in the United States and is a knowledge management company that builds web based knowledge management services and tools. Project Advance was a project to develop
a development library that other development teams at HawkCorp were to utilise. The project had a core team of four full-time developers and one part-time tester. The team was passionate about their Advance library and how it would be used, they were also passionate about being the pilot adoption XP project for the organisation. Two of the senior developers had used XP in previous companies and were keen to introduce it to HawkCorp.

The first release of Advance took five to six months and involved two developers working on it full-time. This first release replaced an existing library and improved the performance of the library significantly, one aspect of the implementation moved from taking six hours to three minutes, and so Advance 1.0 was well received by the development team. Advance 2.0 was four months underway when we interviewed the team, and had a team of three to four developers working on the project.

One of the most interesting aspects of this case, compared to the other cases, was that their customer was a “developer”, rather than a business user, as the end-use of Advance was for development. The team were not able to locate an on-site customer, and instead created an advisory group built from the architects and the lead developers from the teams that would utilise their framework.

The development team lead worked with this advisory group closely throughout the project to create the single voice of the customer. The engineering manager, who like the team, was passionate about both the framework and the pilot use of XP on this project, supported the development team leader throughout this process. The advisory council was a critical component of their implementation of the customer role, however, it took the team a number of months to determine the right people to involve in the council, moving from a general invitee list to a more coordinated and long-term invitees list. They also discovered that typically an iteration would have a particular focus and so they would work closely with a representative from that development area for that iteration.

We spent a day with the Advance team, and interviewed the development tools engineer, the development team lead, two Advance developers, the tester and a developer who would use Advance 2.0. At the time of the interviews people seemed comfortable with the direction and productivity of Advance 2.0. However, the final outcome of Advance and HawkCorp’s XP adoption is not known as the main contact at HawkCorp left a few months after the interviews.

4.7 PROJECT ENDURANCE, TERNCORP

Project Endurance was an outsourced project involving:

- TernCorp, a large national telecommunications provider in a European country.
- PackageCorp, a software package vendor customised their package to meet the business needs of TernCorp. PackageCorp was located in a different European country to TernCorp
- ConsultCorp, a large international consulting firm with agile experience helped TernCorp manage the project

The goal of Project Endurance was to bring all of the elements of the TernCorp online presence together into one platform, including the business-to-business and business-to-consumer aspects of their electronic commerce sites. The online division of TernCorp were driving the project. They spent almost twelve months developing a business case and making initial investigations into the possibilities prior to commencing Project Endurance. The online division was a young, and fast moving business unit within TernCorp and they were particularly keen to try XP or Agile on this project, as they perceived it as a much better fit with their culture. Their experience of the IT organisation was as a large traditional business unit who did not embrace change within the market place at the pace that they needed to in the online division.
Chapter 4: Overview of the Cases

The project commenced with an initial 2-week envisioning phase, where the business representatives worked with some of the IT representatives to develop the initial use cases of the system to be implemented. The project then moved to two-weekly iterations, and the PackageCorp programmers were brought into the same location as the TernCorp and ConsultCorp team. The first release of the project was four months later, and the development team was approximately twenty people in total.

One of the issues brought out during the interviews with this team was the difficulties encountered between the TernCorp and PackageCorp definitions of business concepts and terms. The PackageCorp programmers had a domain knowledge of terms that did not align with the TernCorp’s understanding of the same terms, so it took a few months before they were able to develop a shared understanding of terms and the scope of those items.

It was not clear at the time of the interviews as to whether this project was considered a success. From the TernCorp Online business division it was a success, they had the functionality they wanted and had had the opportunity to react to change as the market demanded. The more traditional units within TernCorp however viewed the project as a failure. The process did not fit their accepted ways of working and through some misunderstandings that we will explore in later chapters the project was a month late. TernCorp had no intention of using XP or Agile in any further projects.

We spent a day with the Endurance project team, and interviewed the IT project manager, two of the business unit representatives, the ConsultCorp project manager, and two of the developers on the project. We were also able to review some of the project artefacts such as the use cases, initial project plans and tracking charts. We interviewed the team a few months after the initial release of the Endurance project, the project team was working on the second release at the time.

4.8 **PROJECT AURORA, SPARROWCORP**

SparrowCorp is a large international company in the energy industry. AgileCorp is an international consulting company that worked with SparrowCorp to deliver Project Aurora. Project Aurora affected the UK and European offices of SparrowCorp, and the project team was based in the United Kingdom. Project Aurora was an internal logistical system that looks after the distribution and transportation of energy throughout Europe. One of the key goals of Project Aurora was to introduce some consistency in the way this was handled throughout the European countries, rather than the specific individualised processes and rules that currently existed.

The initial team of two business analysts and a project manager spent six months working on the vision of the new business process before the larger development team joined the project. The complexity and worldwide geographic dispersal of business representatives (eleven different geographic locations with specific process and systems) made this initial envisioning phase of the project quite long, compared to some of the other projects we have studied. The project team were also uniform in that the eleven different regions “hated” the idea of a common process and preferred their own processes specific to their region.

The remaining eighteen months of the project were in development, with a team of approximately fifteen people. Over the course of this period there were five releases, with the first release targeting three regions, the second release targeting another three regions and so on. At the time of the interviews the third release had occurred and the fourth release was approximately six weeks away. The business analysts involved in the original six-month stage of the project worked as on-site customers for the development phase. They divided their time between the development team and travelling to visit the end-users around Europe. The development team used all of the standard XP practices throughout development. On the whole the developers were comfortable working with the business analysts as their on-site customer.
Overall this project was seen as a resounding success and formed the cornerstone of a long-term relationship between SparrowCorp and AgileCorp. We interviewed four team members, including the project manager, one of the business analysts and two developers over the course of several days.

4.9 **PROJECT GODSPEED, KITECORP**

KiteCorp is based in Europe, and is a large national transport company. Project Gods speed was the long-running (more than three years) online shop provided by the company to its customers. The whole team, at the time of the interviews, including the business representative was approximately twelve people. The team moved to agile partway through the project, and over time the team members had changed. For example, the project manager we interviewed had been with the team for the previous eighteen months.

The team had been assigned a business representative from the company, who represents the business goals and decides the needs of the users. The on-site customer was on-site with the team two days a week, and on-site with the business unit three days a week. The original on-site customer had taken maternity leave three months prior to the interviews, so the on-site customer we interviewed had only been responsible for the project for the last three months.

The on-site customer worked closely with the strategic advisor for the business unit, and together with the team and the steering group they created rolling 6-month plans for Project Aurora. From these plans the on-site customer was able to develop detail in the form of user stories for 3-weekly iterations. The team also had access to people from the original GodSpeed team, and the current support people for the online shop.

We spent a day with the Godspeed team, and interviewed the on-site customer, the strategic manager, the project manager and two developers. The developers and customers were finding the pace of XP quite tiring and stressful. They still, however, believed XP was adding value to their project and so wanted to continue its use. At the time of the interviews GodSpeed was perceived as a success and the organisation was happy with the impact to the organisation’s bottom-line that they were attributing to the recent online shop changes.

4.10 **PROJECT DOLPHIN, ROBINCORP**

Project Dolphin was a mobile phone application development project undertaken by RobinCorp, for a client organisation, MobileCorp. RobinCorp is a small software development house based in the United Kingdom that specialises in mobile phone software development. MobileCorp is a large multi-national mobile phone company, the key contacts for RobinCorp were based in a European country.

Project Dolphin was a long-running project, and the interviews occurred at around the 18-month mark. At its high-point the project had six developers working on it, but at the time of the interviews, it was typically staffed with only two to three part-time developers. The software was released and perceived as a success. One demonstration of that perception is that MobileCorp continued to contract with RobinCorp for additional work, both for Project Dolphin and new projects.

RobinCorp found XP to be an effective development method, and over the three-years they had been using it they had customised it to reflect their own commercial realities and people, but they still practiced most of the standard XP practices. When they introduced XP, they had hired an XP coach to help them adopt and tailor XP to their specific environment.

RobinCorp developed an alternative customer practice that revolved primarily around their account manager for the vision of the software and their project manager for the detail of the
software. The account manager and project manager would work closely with MobileCorp executives and client project managers respectively to evaluate what needed to be built. They would provide a layer of interpretation and detail into the software requirements. The project manager emphasised the transparency of this process with the development team. For example, all electronic written communication would be available to the team in a central repository. Developers would also attend sessions (both phone and in-person meetings) with the client representatives, but the project manager remained the central touch-point. This strategy was perceived as effective for them and the client, and their ongoing relationship would seem to support that perception.

We spent a day with this small XP team, and had the opportunity to interview the project manager, two developers and their tester, approximately 4 hours of interviews. The team enjoyed working with XP, and all loved working for this small specialised software development house in an idyllic location within the United Kingdom.

4.11 PROJECT CALYPSO, OWL CORP

This section tells the story of Project Calypso, a ‘successful’ project undertaken by OwlCorp. OwlCorp is a world-wide company working in the domain of health. OwlCorp builds systems to support its employees to work in their domain. The main development centre for Project Calypso was based in the United Kingdom, with a second team of developers based in Europe. End-user representatives were primarily based in the United Kingdom but some part-time representatives were involved from the Europe and United States offices of OwlCorp.

Project Calypso was the third project in the same space for OwlCorp over the last five years. The previous two projects had been cancelled, the first of which had used a traditional document-centric method, and the second had used XP, where OwlCorp partnered with a consulting firm that specialises in agile software development. Both projects were cancelled for a number of reasons that in effect amounted to insufficient budget for the envisaged scope. So Project Calypso III started life with significant tension between IT and the business. Additionally, a large loss of project memory had occurred as most IT and business people involved in the previous two versions of the project had left the company.

Calypso III had approximately twelve developers, three testers, two business representatives and three project managers and the project lasted approximately one year. The IT group were keen to develop some part of the Calypso project to help restore the business management’s faith in their department. So, a few months after Calypso II was cancelled they began work on a Calypso III beta, a significantly cut-down version of Calypso II. The beta was presented to the steering group and Calypso III was given approval to proceed. OwlCorp created a Calypso project working area, and the IT and business staff (two full-time end user representatives and a business project manager) moved into the space (located within the IT department office area) and worked together.

The group began forming around an initial agile training boot-camp and 2-weeks of story-writing workshops that would form the basis of their release plan. The team formed close working relationships during the project despite the “rocky” start. The interviews to collect the data for this case study occurred just as Calypso III was under-going final UAT; the system was released approximately one month later. All indicators, at the time of the interviews, were that the project was considered a success, however, it had been a very rocky journey, and there was still significant issues between senior IT management and senior business management, despite the good relationships developed on-the-ground between IT and business representatives. Although this project was perceived as successful, ongoing software development activities were outsourced.
We conducted nine 1-hour interviews with Project Calypso team members, including interviews with the executive manager of the affected business unit, the two end-user representatives from that business unit, a business project manager from that business unit, the architect, the IT project manager, the IT team leader and two developers. Additionally we conducted two 1-hour follow-up interviews with the two end-user representatives. The interviews were all one-on-one semi-structured interviews, and occurred near the completion of their first release of the software product. Finally, the team was observed over the course of two days during an iteration.

4.12 **SUMMARY**

In this chapter we have provided a high level overview of the eleven projects studied as part of this research. The projects have covered a mix of domains, project and company sizes, successful and challenged projects as well as geographic locations. The projects were collected in four separate data collection phases, with the first two concentrating on initial theory generation and the later phases concentrating on refining and elaborating that theory.

Please note that for the purposes of this thesis we have classified the first five cases obtained during data collection phase I and II as well as the last case obtained during data collection phase IV as our primary cases. These cases (Endeavour, Discovery, Atlantis, Pinta, Speedwell and Calypso) provide the data that forms the basis of the theory concerning the on-site customer. The remaining cases will be treated as supplementary cases, and will be used throughout the thesis to provide further illustration or examples of the theory discovered from the primary cases.
5. BEING AN ON-SITE CUSTOMER

In this section we will present six in-depth case studies that outline how the on-site customer role has been implemented. Five of the cases presented were project success stories at the time of the interviews, one of which was cancelled a few months after the interviews. The case presented in section 5.4, however, even at the time of the interviews, was considered a seriously challenged project, and as such is presented in a slightly different format, ensuring the potential reasons for ‘failure’ are highlighted.

Quotes from the interviewees are used to support our description and interpreted findings. All quotes have been modified to retain the anonymity of the organisations and participants involved. Quotes have also been modified at times to enhance readability. These changes are clearly marked with square brackets. Quotes appear as indented italicised text following the same conventions as Gittons, Hope and Williams [16].

5.1 PROJECT ENDEAVOUR, KIWICORP

Project Endeavour was an outsourced custom development project and involved a number of organisations as depicted below. The purpose of the project was to build a content management system.

Both KiwiCorp and DevCorp saw project Endeavour as a success. The project was approximately fifteen months long, and at its peak the project team had eleven full-time experienced team members.

5.1.1 WHO WAS THE CUSTOMER?

The identified customer on the project was a records librarian from KiwiCorp. She had extensive domain experience of both the functionality required for the system and in “getting things done” in KiwiCorp. The customer was responsible for determining what to build, ensuring it was built, and managing the outsourcing relationships. The customer was supported on this project by:

- A KiwiCorp senior manager, who assisted the customer to prioritise the requirements and to drive decisions affecting the project through at a senior management team level.
- A KiwiCorp acceptance testing team, who were responsible for testing the stories.
- A DevCorp analyst who was responsible for writing an initial draft of a user story for the iteration based on the initial functional specification and conversations with the customer.
5.1.2 WHAT HAPPENED?

KiwiCorp had attempted project Endeavour twice previously, but each previous attempt was unsuccessful. In fact, the KiwiCorp customer representative noted that when her manager was handed this project to ‘sort-out’:

“We felt we could probably only do it if we used [DevCorp] ... because we had such a lot of confidence in them based on previous experience ... [later in the interview] ... we knew what they were capable of, ... they could actually deliver what they said they could ... a key value of working with [DevCorp is that] it was a joint effort.”
— Customer, KiwiCorp

Initially the project was a traditional waterfall project and was divided into three phases, planning (deciding what to build), development (building the application) and implementation (user acceptance testing, training and roll out). The planning phase focused on gathering requirements using standard workshop techniques and involved a series of user workshops that the business users attended.

At the end of the planning phase, KiwiCorp and DevCorp decided to use XP for the development phase. The requirements gathered during the planning phase were used as a basis for the XP user stories. The XP iteration process was followed, except that the customer was not always on-site with developers, but she attempted to spend fifty percent of her time at the same location as the developers. The implementation phase was retained as this approach meshed with the existing practices of KiwiCorp and BureauCorp.

5.1.3 HOW WERE THE REQUIREMENTS GATHERED, EXPRESSED AND PRIORITISED?

The user stories allowed the customer to adapt or evolve the initial requirements as they watched the system develop and learnt what does and does not work or matter. Both the customer and development team members commented on the approach:

“If we’d had to come up with requirements to the [nth] degree right at the beginning, I think it would have been very difficult because I hadn’t worked with intranet content management systems so a lot of the time I was sort of learning as I went and once ... I saw the bare bones thing, it was much easier to make more detailed decisions for the next level.”
— Customer, KiwiCorp

The team did not follow the standard XP process for documentation because the user stories were based on a functional specification. KiwiCorp formally accepted the functional specification at the completion of the planning phase. No formal change control was implemented to track the changes between the user stories and the functional specification. Neither was there a formal agreement that the user stories superseded the functional specification. The DevCorp team noted this approach, particularly in an outsourcing arrangement, could result in a scope debate between the organisations. However, they did not find this to be a problem for themselves, as they had developed a good collaborative working relationship, and were operating under a time-and-materials contract.

The customer noted that one issue they had encountered with the user stories and communicating the requirements with conversations was knowledge transfer. For example, the end-user training material development required an understanding of the system typically obtained from the functional specification, but the functional specification no longer reflected the system functionality. No documentation existed that specified the system to the level of detail required by the trainers. Cockburn [64] discusses this issue and suggests it is best to determine a need for the documentation rather than simply producing it. This suggestion indicates new adopters of XP
should consider their knowledge transfer requirements (conversations or documentation) and plan accordingly.

The other change to the process was that the customer did not write user stories and instead the stories were based on the functional specification. The project manager emphasised the time it took the development team to write user stories. He went on to discuss the potential impact of relying on the customer to create the stories:

“I know strictly speaking that’s the clients job if you read the books but I actually did that to sort of streamline the process a little ... I don’t think the client would ... have had enough stories prepared on the day iteration kicked off so we’d have delayed start[ing] the iteration and I think ... delay[ing the] iteration is the slippery slope – [and against the] .. whole point ... of time blocks and time approaches”
— Project Manager, DevCorp

The project manager did not clarify his concern further. Possible interpretations include:

- The time available to the customer on this project was insufficient to include this task which is supported by the customer’s comment that she needed more time to effectively contribute to requirements development and testing.
- The difficulty (maybe the technical skills required) to write usable user stories for developers as suggested by Beck and Fowler [3].

It is interesting that no interviewee noted that the development team writing the user stories and the customer reviewing and confirming the stories was an issue or deficiency in the process.

Finally, one of the key roles of a customer is to make business priority decisions about what functionality is delivered when. During the interview the customer demonstrated her strength of feeling regarding prioritisation:

“I hate it, I hate that word prioritise ... [and later in the interview ]... Usually I’ve done [the prioritization] myself and in conjunction with [my manager], when I can’t bear the responsibility anymore”
—Customer, KiwiCorp

The customer also learnt the importance of not assuming all functionality would be delivered on the high level plan, and so in hindsight believed she should have started prioritising from the start of the process:

“We thought we’d bitten off what we could chew but we worked out that our eyes are bigger than our stomachs – it works for IT too ... so I guess one way of doing it would have been to ... do the prioritising right at the time – what can we actually leave till last? And actually leave that till last.”
—Customer, KiwiCorp

This point concurs with the XP philosophy of delivering functionality in business priority order. However, this study also suggests that it is a learning process for the customer.

5.1.4 WHAT WAS THE CUSTOMER’S EXPERIENCE?

The customer was in no doubt about her preference to use XP in the future:

“Overall – I love this approach to development and I’d certainly like to use it again in any future projects I am involved in”
—Customer, KiwiCorp
However, she also notes later in the interview:

“I think we needed some extra roles basically. We probably needed about three of me ... it's been my life for about a year ... look at these grey hairs”

— Customer, KiwiCorp

Although the customer would have liked to devote 100% of her time to being the on-site customer and working with the programmers, she also needed to work with the end-users and business stakeholders in KiwiCorp. Representing thousands of end users is a significant task and to accomplish it well she needed to be on-site with the users. Finally approximately 50% of her time was spent managing technical integration issues between DevCorp and BureauCorp.

5.1.5 SUMMARY.

Project Endeavour was a successful implementation of XP, in an outsourced environment, where both organizations were keen to use XP on their next project. This project was of the recommended size and the spirit of XP was retained, although the implementation of some practices varied.

The customer on this project was responsible for all business decisions, including the prioritization decisions. Although she was not located on-site with the development team, she spent approximately 33% of her time with the team. The customer recognized that spending more time with the developers would have been ideal. However, she also needed to work directly with the users and managers she was representing in her organization as well as managing the two outsourced vendors on this project, so she did not have more time to spare. Finally the customer did not write the user stories, a DevCorp analyst always wrote the first draft, from that point on the customer modified the story.

One of the key issues uncovered in this case was the overwork associated with the customer role, with the customer identifying the need for “three of me”.

5.2 PROJECT DISCOVERY, RAVENCORP

RavenCorp is a company based in the United States. Project Discovery started in mid 2000, with the intention to move a piece of scientific software developed in an academic environment into a commercial software product. The initial project development lasted over three years and at its peak there were almost twenty people working on Discovery.

5.2.1 WHO WAS THE CUSTOMER?

The identified customer on the project, at the time of the interviews, was one of the founding members of the company and is a scientist. He had been involved in developing the original academic software and as a result had a close professional relationship with the founding directors of RavenCorp.

The customer has been on the project, full-time, since the founding of the company four years earlier, and throughout this period he had the role of the XP customer. He also had the role of programmer, particularly within the first two years of product development. Although, at the time of the interviews, he no longer had the role of programmer, he had, however, become the development manager within the company.

The customer was supported in his role by:

- One of the founding directors of the company, who was responsible for the initial vision of the product and working with potential buyers to better understand their requirements.
A scientist with a similar background to the customer, who was responsible for using Discovery to produce product demonstrations for potential buyers. This responsibility ensured he tested the system as an end-user on realistic production scenarios and provided this feedback to the project team.

A small team of research scientists, who were responsible for researching the patented computational algorithms that form the back-end of the product.

An experienced software tester, who implemented an automated acceptance testing environment and worked with the customer and the research scientists to develop a suite of acceptance tests.

Finally it should be noted that for the first two years he shared the role of customer with another founding scientist, until that scientist left the company. They worked together closely and each had key areas of functionality they had been responsible for.

5.2.2 WHAT HAPPENED?

Planning the Product. Discovery was a new product, there was no existing product to replicate and much of the product was based on cutting-edge research discoveries made by RavenCorp scientists. Planning the full scope of the product at the start of the project was not feasible or potentially even desirable. For the first three years, including the first year of the team using XP, very little long-term release planning occurred, instead the team focused on evolving the product based on their discoveries, iteration-by-iteration.

At the end of 2003 that changed, and the customer spent a significant portion of his time in 2003/4 planning the scope of the product for a 2004 release date. Although there was a significant portion of discovery in this project, the release planning could have occurred earlier in the process, the customer noted:

“In the early days ... there was always too much to do ... we were basically working to have enough stories for the next iteration ... let alone have enough stories where you could actually have sort of a larger view ... [of] a few iterations or a release.”

—Customer, RavenCorp

His solution was to work the release planning process in parallel to the iteration planning process, rather than incorporating an intense release planning process into the team’s schedule. The customer worked on gathering and expressing the release user stories and publishing them to the project wiki. The team found one long release planning game frustrating, and so instead the customer has created many smaller sessions to prepare for a release planning game. He ran one hour sessions twice a week with the programming team to review and estimate the release user stories. This information allowed him to review and prioritise the scope of the release with the RavenCorp stakeholders. This approach meant that over the course of a year he slowly “got-ahead” and established a release plan. His intimate knowledge of the domain, as a research scientist, was potentially a factor in the success of this strategy.

Evolving the Product. Initially the team followed a standard XP iteration process, with the customer deciding on and writing the iteration stories a couple of days before the iteration planning meeting. The release planning process allowed this process to evolve over the course of a year so that the time spent in iteration planning meetings was drastically reduced:

“In the early days the meetings were... like four hour meetings and they just dragged on and I think everyone would lose their patience at a certain point”

—Customer, RavenCorp
The small regular meetings allowed the customer and programmers to discuss the stories in small batches, resulting in the iteration planning meeting simply being a refresher discussion, followed by the tasking process.

One of the other variations to the process was the involvement of the research scientists in the acceptance testing process. The research scientists developed the algorithms in the product and were in the best position to accept the implemented algorithm. An automated acceptance-testing framework built by the team allowed the research scientists to write tests to test their algorithms. RavenCorp followed all other aspects of the standard XP process.

**Releasing the Product.** Discovery, at the time of the interviews, was still some way off being released as a shrink-wrap product. However, RavenCorp did release the product regularly, every two to three iterations, to the internal RavenCorp stakeholders, including the end-user representative and the founding director. These releases allowed the team to receive end-user feedback, as the end-user representative used the release to create product demonstrations for external (potential) clients.

### 5.2.3 How were the Requirements Gathered, Expresssed and Prioritised?

The customer was an expert in his domain, and a team of RavenCorp domain experts supported him, so determining what to build was not difficult for the customer:

> “We are building a scientific application and my domain experience is such that I understand what the application is supposed to do ... [and later] I think that for me, coming up with the ideas for “what we need to get done” or “what this story should be about” ... [has] not been too challenging for me ... [however] one of the things that has been frustrating for me is how detailed to make the stories”

> – Customer, RavenCorp

He worked with the programmers for two years to evolve the level of detail that he placed in the user stories. However, he noticed the appropriate level of detail was still a concern when writing the story. He struggled with finding the level of detail and decomposition that ensured the programmer was able to engage and understand the purpose of each story, while still ensuring the customer did not make design decisions when he decomposed the functionality. Finally, one aspect of story writing that he had discovered was critical to the quality of the product was the inclusion of exceptions and error conditions for each story.

The customer noted that one prioritization issue the team encountered, in hindsight, was spending too much time in the early iterations on infrastructure functionality rather than implementing business functionality. It had been tempting to get a solid technical foundation under the team prior to moving onto business functions, but in hindsight he had realised the importance of the XP phrase “you aren’t going to need it” (YAGNI) and hence to only build what is required to implement the current user story. He further suggested that delivering business functionally quicker would have engaged the stakeholders even earlier to assist in the shaping of the product. It also would have helped to ensure, when the hard trade-off decisions were being made later in the project, that business functionality took precedence over the technical infrastructure.

### 5.2.4 What was the Customer’s Experience?

The customer enjoyed his role on the project, he was very passionate about the product, and felt that XP had added “just enough” rigour to their development process. However, he had found juggling his multiple roles and responsibilities at the company wearing:

> “For the entire time I have been at the company ... I’ve always worked at least 70, probably more on an average of 80 [hours a week] ... I don’t even mind it, its
like what I do ... I don’t worry so much about how much time I’m putting in because it doesn’t even really register for me, but yeah, all these different responsibilities can be a little bit wearing."

—Customer, RavenCorp

The level of domain understanding, and potentially the personal attributes, required to be the customer, also make the customer a valuable resource for the company. As a result the customer is too valuable a resource for the company to be able to simply focus on the product:

“... so it’s really been too much responsibility ... on myself and [the other customer] ... I think that was a large contributor to the problems we have had in the past ... we weren’t getting the stories written up at a fast enough pace ... well I’ve got to write the stories and think about product planning but I’ve [also] got to manage the team and I’ve got to think about this and that ... that’s just too much to deal with ... balancing all those things can be difficult, [it would] definitely be nice to be able to just focus on one thing like just focus on the product”

—Customer, RavenCorp

The customer also noted that part of the time commitment is due to the interactive nature of the on-site customer role:

“It may take you an entire afternoon to do something that’s like an hour of work because you’ve been interrupted and had you know 20 minute discussions a number of times through the afternoon with the developers and its all valuable stuff ... then you look back and you’re like man did I get anything done today ... so it is a bit deceiving in that respect I think”

—Customer, RavenCorp

So given the interactive nature of the customer role, the customer typically structured his day so that during the normal 40 hour working week he was available to the programmers, and the rest of his work occurred outside of those hours. Interestingly, this had meant that many of the development team, including the coach, were not aware of the workload on the customer.

5.2.5 SUMMARY

XP was perceived has having worked well with the “research and development” nature of Discovery. In fact, Project Discovery was another example of the successful implementation of XP. Once again, the project fell within the recommended size for XP projects and the team had implemented the values, principles and practices with little variation from the standard XP outlined by Beck [2].

The customer was available on-site to the team, and in fact prioritized this interaction above his other duties. He worked with the key stakeholders in order to provide a single voice to the development team, and be in a position to make all of the required business decisions. He undertook all of the typical customer tasks including writing stories and testing stories, although he had some help with the later task.

One non-typical aspect with this customer was that he had at times also held a programmer role on the project. He started off programming scientific algorithms but as he began to focus on the visual interface scientists would use he up skilled to include graphical user interface development in his skill-set. His dual roles blurred the boundaries of the business and technical decision-making process. However, it seems likely that other XP projects may encounter this need, particularly in start-up organizations and also products where the final users of the product are programmers.
The RavenCorp product manager fulfilled all of the requirements of the on-site customer role. However, the time commitments required of him to fulfil his on-site customer role requirements may not be sustainable for other product managers in a similar situation.

5.3 PROJECT ATLANTIS, EAGLECORP

EagleCorp specializes in the development of products that support application development and application development management. The aim of Atlantis 7.x was to redevelop their flagship product 6.x in a new technology with an emphasis on usability. At the time of the interviews the project had been underway for approximately eighteen months and had sixteen experienced full time staff. All Atlantis 7.x staff were co-located in the same office in the United States.

5.3.1 WHO WAS THE CUSTOMER?

The identified customer on the project, at the time of the interviews, was a product manager. The product manager had over fifteen years of software development experience covering a spectrum of roles including, programming, application support, testing and product management. The majority of this experience was in the same area as Atlantis, that is, products that support developers and development managers. The product manager had only been with EagleCorp approximately twelve months prior to joining the Atlantis team.

The product manager was responsible for the product definition and vision as well as the detailed functional requirements. He, however, was supported on this project by:

- An experienced user interface designer, who developed the user interface guidelines and mock-ups. A decision was made by the senior management in EagleCorp to add an experienced user interface designer to the project, as one of the key improvements to Atlantis 7.0 was to be the user interface.
- An experienced technical architect who was responsible for all non-functional requirements, including design and code quality. He also played the role of programmer when required.
- An experienced technical writer who was responsible for writing the user guide.
- An experienced Quality Assurance (QA) team of four quality engineers and one manager, who were responsible for the final full system and integration testing of the product.

Finally it should be noted that at the start of the project there was a product management team, consisting of four product managers. Approximately four months after the decision to move to XP, EagleCorp downsized, and cut the product managers on this team down to one. The development and testing staff were also reduced, both company-wide and on this specific project.

5.3.2 WHAT HAPPENED?

Planning the Release. The first activity that occurred on this project, for the product manager, was the definition of the product. The outcome of this phase was a product plan, an agreed and prioritized set of functionality for Atlantis 7.0. Running in parallel to this activity was a technical stream, focused on deciding and proving a new technical architecture.

Once the product plan was agreed and the architecture defined, EagleCorp decided to move to XP. The team had an XP release plan, courtesy of this initial work, and so immediately began XP at the iteration planning stage.

Approximately four months into the twelve-month development process, this release plan was revisited. At this point enough data had been gathered to recognize that the original release plan was unachievable. EagleCorp needed to make a decision to cut scope or increase the timeframe. To assist this decision, the team stopped the XP iterations in order to assess and estimate the
remaining functionality; this process took approximately a week. The EagleCorp executive team chose an option that involved both cutting scope and increasing the timeframe.

Evolving the Product. Over the course of the project the iteration process evolved. The resulting refined process meant that for the product manager and the user interface designer, the two-week iteration started approximately a week before the iteration planning meeting. During that week the focus was on developing the user stories and user interface mock-ups to present at the iteration meeting.

At the iteration planning meeting, the product manager would walk through all of the user stories and user interface mock-ups with the programmers and elaborate as required to answer the programmers’ questions. Once the programmers felt comfortable that they understood the user stories enough to estimate them, the programmers estimated all of the user stories. Finally the group agreed the scope of the iteration based on the product manager’s prioritisation decisions and the previous iteration’s velocity.

During the next two weeks, the programmers would implement the stories. If at any point they required clarification of the user story they would discuss the user story with the product manager for functional clarifications, or the user interface designer for user interface clarifications. At the end of each day the product manager, user interface designer and engineering manager would review and test each of the completed user stories. Accepted stories would be marked as completed. The engineering manager would discuss the unaccepted stories with the programmer pair concerned, and outline the items that needed to be complete before the story would be accepted.

At the next iteration planning meeting the previous iteration would be reviewed for any issues and the team also spent time evolving or improving their process during the same meeting.

Releasing the Product. EagleCorp has a Quality Assurance (QA) team that focus on full integration testing of the software products. The QA team was overloaded throughout the development process with Atlantis 6.x testing commitments, and as such was only brought into the process near the end of development. As a result the QA team was not present at either the iteration planning or daily acceptance meetings, and missed these undocumented “conversations”, which increased the difficulty of their job. However, the QA team was pleased with the initial quality of the product and was keen to be involved in the “conversations” throughout the next release and adapt their QA processes to XP.

Another aspect of releasing a product is the development of user documentation. The technical writer noted the XP process significantly aided her task. She was able to write the user manual based on both the user stories and working software. In a typical waterfall process she must write a user guide based on a specification that is quickly out of date. She then faces a significant rework phase once a working product is available for her to explore, typically only a week or two prior to shipping the product.

5.3.3 HOW WERE THE REQUIREMENTS GATHERED, EXPRESSED AND PRIORITISED?

As outlined in the previous section, the first step facing the product managers was to define the Atlantis 7.0 feature set. The inputs into determining the release feature set included the existing product functionality as well as specific customer requests. The product manager described this activity that took a few months to complete:
“We [did] a lot of cross-functional, cross-decision making meetings. We talked to the executive team, we [worked] with the rest of the organisation to define requirements ...we had ... broad-brush organisational agreement [to the prioritised feature set].”

—Product Manager, EagleCorp

The resulting feature set was a product plan, and it listed approximately 50 features with an associated priority of 1, 2 or 3. The top priority features were to form Atlantis 7.0, with the remaining features to be implemented in later versions. Atlantis 7.0 included some new features but did not include all of the features of Atlantis 6.x.

Once the product plan had been agreed, the product managers turned their attention to developing the user stories:

“One thing that helped us [write good user stories], is that [around] the same time [we started writing user stories we] started with [the] persona driven approach ... from our UI designer, so we ... had hypothetical users of the system, so we [insisted that all users stories] must be based on these three ...identified ... primary users. So that’s helped us [to move from] I want this and that feature, [to explaining] what this person wants [to do in the system]”

—Engineering Manager and Coach, EagleCorp

The “persona” approach stems from a method for user interface design suggested by Cooper [65]. The personas of the three primary users of Atlantis were based on market and sales research, and included details such as the end-user’s name, age and job role. This technique ensured the prioritisation process focussed on the features required by the primary personas. This technique also resulted in a good sized piece of concrete functionality for a programmer to implement that could be tested, although this outcome was not immediate, but evolved as the product manager and programmers worked together.

In parallel to the writing of user stories was the development of the user interface mock-ups. The user interface designer developed a user interface guideline document that included detailed items such as the space between fields, as well as a high level picture of the user interface metaphor. The user interface designer then worked with the product manager, defining the user interface mock-ups for the user stories within the current iteration.

In the above description it may appear as if the product manager worked in a vacuum. However, the product manager throughout this process interacted with a mixture of internal and external stakeholders.

In the case of internal stakeholders, such as the sales department, application support department and executive team, the product manager presented the product as it evolved in a regular monthly meeting. This meeting met a dual purpose of both reporting progress as well as providing a venue for the internal organisation members to evolve or re-prioritise functions.

In the case of external stakeholders, EagleCorp established a Customer Advisory Council (CAC), comprised of their key set of accounts. The CAC typically met once a quarter. Two of these meetings were face-to-face meetings over a two to three day period, and the other two meetings were an hour or two long and called “webinars”. Important to these meetings was the emphasis on two-way communication:

“...for me, from a PM perspective, [these sessions are] very valuable because [they] help me understand what some [of our] key customers are doing, some of the key issues that they have ...which parts of the application [they] don’t ... use, which parts of the application need major changes before [they] could find them valuable...
... [although not all of our customers are represented] it helps me get a very good understanding of ... where some of the real major holes are and how I should ... spend my engineering dollars [to build] the next generation [of] tools”
—Product Manager, EagleCorp

The ability to present Atlantis 7.0 as it currently stood at each of these meetings greatly enhanced EagleCorp’s ability to obtain concrete feedback on the product direction from the end-users, thus was completing the two-way communication loop with external end-users of the product.

5.3.4 WHAT WAS THE CUSTOMER’S EXPERIENCE?

There was a consensus in the team, from all perspectives including senior management, programmers and the customer, that the on-site customer practice was a drastic improvement over the traditional document-centric processes. EagleCorp attributed the success of Project Atlantis, at least in part, to the predictability provided by XP. Firstly, the predictability of dates. The two-weekly iteration cycle allowed EagleCorp to plan and execute their development significantly more accurately than they had in the past. Secondly, the predictability of quality. The quality focus provided by test-first development and rapid customer acceptance testing have enabled EagleCorp to increase the quality of Atlantis 7.0. Finally the predictability of flexibility, otherwise known as “embracing change”:

“We as a business [would] be foolish if we said, no we can’t take any new things. [XP enables us to ensure] what we are releasing is going to meet our customer’s needs as of today, not ... what they were several months ago”
—Vice President of Product Development, EagleCorp

However, the changes that have caused this improvement have also had some unintended, and perhaps unrecognised, impacts:

“The iteration processes are quite exhausting. Every two weeks you’re basically doing another product release. ... So it’s a very, very intensive process to keep up and maintain... [From] the first year of doing it I think it worked really well but, from a longevity perspective, I don’t know how long it will be possible to keep [this pace] up”
—Product Manager, EagleCorp

The product manager went on to compare the workload to that of a traditional process, where a document is “handed over the wall”. In the traditional process although there were “peaks” there were also corresponding “troughs”, when the responsibility is passed to the development team. In XP, the consistent pace resulted in no “troughs” for the customer. Not only were there no troughs, but the type of work was significantly more intense than writing a document:

“You know, you can work 40 hours on a document or you can spend four hours in an intense meeting where you are getting hammered with questions, you know, you’re getting issues you’re having to face and deal with in a four hour meeting and feel more tired from the four hour meeting than the 40 hours of document development. So it creates a different kind of pressure I think on the PM. It makes ... again the iteration meetings and the requirements to deliver every time you go into these iteration meetings, ups the pressure, ups the intensity and just being prepared for these things is the key. So, you know, you’re also acutely aware of the fact that you have this large group of people who are sitting in this meeting costing the organisation a lot of money. If you haven’t got it right, something isn’t thought through correctly...”
—Product Manager, EagleCorp
So although there had been a significant improvement as the result of the on-site customer role, the practice of sustainable pace had been sacrificed.

5.3.5 SUMMARY

Project Atlantis was an almost perfect example of an XP implementation, it was the recommended size and it retained the spirit of XP, its values and principles, as well as the practices of XP.

The product manager fulfilled the role of on-site customer. He worked on-site with the development team, wrote the user stories, accepted the user stories, and worked with both internal and external stakeholders to ensure he was able to make the business decisions for the project. The small iterations allowed the product manager to use the working software to both report progress to internal stakeholders as well as obtaining feedback and evolving the product with both internal and external stakeholders. EagleCorp tailored XP in three significant ways that impacted the XP customer role.

The first significant modification was the introduction of user-interface design principles into the process. Additional documentation was introduced, including a user-interface guideline as well as mock-ups of all of the screens. Finally, all of the user-stories were modified to implement the persona user interface technique, which they discovered had the pleasant side effect of providing a good decomposition technique for user-stories.

The second significant modification was the introduction of supporting roles for the customer, including the user-interface designer, a technical writer, an acceptance team and an architect. All of these specialists assisted the customer to define and refine the product based on their specialist skills.

The third significant modification was the introduction of structured techniques to manage the interaction with internal and external stakeholders. These techniques included regular meetings ranging from “webinars” to “face-to-face” meetings.

However, despite the success of the project and these customer-supporting initiatives, there remained a concern over the long-term sustainability of filling the on-site customer role.

5.4 PROJECT PINTA, FALCONCORP

Project Pinta was an outsourced XP project involving three organisations as depicted below in Figure 5-2. At the project team’s peak over 60 people worked on the project in the FalconCorp labs. FalconCorp does not consider project Pinta a success.

![Figure 5-2: Project Pinta Organisational Relationships](image-url)
5.4.1 WHO WAS THE CUSTOMER?

Project Pinta was not a typical custom build project, as the intent of this project was to develop a product that would meet both the needs of RCCorp and become a product within FalconCorp’s product suite. As such there were four on-site customers:

- Two business analysts from the European office of ManageCorp. The ManageCorp analysts had been involved in writing the initial high-level requirements that had been accepted by RCCorp.
- Two product managers from FalconCorp.

No representatives from RCCorp were directly on the project. However, ManageCorp analysts were able to interact with RCCorp representatives, although in order to co-locate the ManageCorp analysts with the team, all of this interaction was remote. RCCorp and ManageCorp representatives had the product demonstrated to them at key milestones.

Finally, a number of business analysts and testers from FalconCorp informally supported the four on-site customers.

5.4.2 WHAT HAPPENED?

Project Pinta was not perceived as a success by FalconCorp, and part of the concerns with the project has been the use of XP. The project took longer than the original schedule, at the time of the interviews, more than three times the original estimate, and had yet to be accepted by RCCorp. The project started out as a typical “death march” project:

“Everyone that was on it said it was doomed for failure [but] we were going to make it work anyway”
— Business Analyst, FalconCorp

The team had less than six months to cover almost 30 functional areas, so management had quickly scaled the project up to more than 60 people across several XP labs. The team established a pace with weekly iterations. At the end of each week the team produced working software that implemented some but never all of the prioritised stories. It quickly became obvious that all of the functionality would not be completed by the required deadline, despite everyone’s best efforts. However, FalconCorp had committed to deliver the functionality by this deadline, and refused to adjust their original plans. The unintended outcomes of how this unwelcome knowledge was handled are outlined below.

The outsourced vendor, particularly in an aggressive fixed price project such as this one, must manage scope creep. FalconCorp managed the scope by not exploring exceptions or evolving the requirements:

“We had the client on site but we didn’t make use of them in [the] way that we should [as] we didn’t want to raise any questions that would lead to gaps in the requirements that we would ultimately be responsible for ...and so we built [to the specification] without asking questions ... [our aim was to build a product that] we could check off and say ... we made our deadline, we deserve our payment”
— Business Analyst, FalconCorp

In fact, the level of mistrust and organisational self-protection rose to such a level on this project, that the ManageCorp analysts were removed from the labs:

“There were roadblocks that were on both sides [that stopped] real communication. We had the client over here and I remember the first week that I
was here they were talking about kicking one of the [ManageCorp] people out of the labs because they were a spy ... they were going back and reporting what was going on in the labs... You'd think that was part of what they were for but, you know, they thought they were giving a bad impression back to the client ... and so they actually kicked them out of the labs at one point.”
— Business Analyst, FalconCorp

The result of this organisational climate occurred at the end of this phase of the project when members of the team presented what was intended to be the final demonstration of the software to RCCorp in Europe:

“It was kind of a sales pitch because they [RCCorp] didn’t really test it. We demonstrated how it worked with some scenarios instead of letting them just hammer anything, because if you hammered on anything, you [would] have big huge gaping holes”
— Business Analyst, FalconCorp

The demonstration to RCCorp went well, so well, that FalconCorp laid-off over two thirds of the original staff, as they were now entering “bug-fixing” mode. During the next two months, FalconCorp appointed new project management to Project Pinta. These new managers quickly assessed that the project required significantly more work than “bug-fixing”. The final decision was to treat the previous system as a throw-away prototype and re-write the entire product. The contract was renegotiated and FalconCorp expected to complete the project in mid 2004.

During the course of the interviews, the participants reflected on their experiences with XP and outsourcing. These reflections are outlined and discussed in this section.

**Scope Definition.** A non-negotiable area for FalconCorp as part of the contract renegotiations was an up-front requirements gathering and documentation stage. As an indication of the size of the gaps discussed, one of the documents expanded the initial requirements document from a 30-page document to a 250-page document.

Schwaber [38] writes that agile development, in this case Scrum, is not a silver bullet for fixed-price contracts. In fixed-price contracts an up-front requirements gathering phase is required to accurately estimate scope. It is interesting that FalconCorp did not believe the initial system developed gave them sufficient knowledge and that an additional significant up-front requirements activity was commissioned. During the interviews potential reasons were uncovered, including (a) significant product knowledge was lost when the majority of the staff from the early development project were laid off and this was compounded with XP documentation practices and (b) there was a perceived failure of XP by management and so waterfall processes were seen as ‘safer’ in a contracting arrangement.

**Development Process.** Pinta has yet to complete but the interview participants’ are clear that the project is now much closer to a typical waterfall process than an XP process. One participant reflected on this change:
“XP [if you go straight from the book] is useless from my perspective ... if you have a fixed scope, a fixed date, well ... how well is that spec going to play in that space – not very, because it relies on the ability to manage that and to change that and impact that over time as realities come out. Whereas with a contractual arrangement, you can’t, you have very little wiggle room to be able to shift that and manage that and change that, and so XP finds itself significantly hampered and you have to start shifting it more towards Waterfall over time, because you just had to deal with the realities of the fact that you must get certain things out in certain timeframes.”

— Development Coach, FalconCorp

It appeared, at the time of the interviews, that XP will end up taking the blame for the problems encountered within this project in FalconCorp. While it seems from this case that there are issues with using XP on fixed price contracts without an up-front requirements gathering and documentation phase, perhaps there is more to this case.

Kern, Willcocks and Heck [66] outline a typical outsourcing issue called the Winner’s Curse that may play a significant role in this case. The Winner’s Curse occurs where the winning organisation or person has over-estimated the value of the object being bided on, typically due to inaccurate assumptions made during the bidding process. Inaccurate assumptions in IT outsourcing can range from the Client Organisation’s political climate causing long turn-around times through to functional scope complexity, such as the one-line functional bullet item in the original specification that becomes a complex sub-system over the course of the project. The end-result, the organisation or person makes a loss, or at least no profit on the transaction. The Winner’s Curse is a prevalent issue in IT outsourcing.

In IT Outsourcing the ‘Winning’ organisation will then typically concentrate on cost-cutting strategies, such as placing inexperienced staff on the project or focussing on the letter of the contract scope not the spirit, to reduce their loss. The end result for both parties is a loss situation, as the resulting software is unlikely to meet the client’s needs. Project Pinta exhibits all of the signs of the Winner’s Curse.

5.4.3 WHAT WAS THE CUSTOMER’S EXPERIENCE?

In this case study, unlike the other case studies, XP was not perceived as an improvement over the traditional way of working. Given the document-centric and bitter relationships that arose between the different organisations this is perhaps unsurprising.

The customer’s experience was indeed bitter, they were located in a different country for months, and at one point they were removed from the labs and treated as “spies”. They had a demonstration of the functionality and thought all was on-track, only to be advised two-months later that the project needed to be started again from scratch, taking well over twice as long as originally advised. Although the internal FalconCorp product owners were not treated as spies, they faced a similar disillusionment concerning what was delivered and the length it took to receive a usable product. From their perspective XP had been a dismal failure, as the system developed using XP was “thrown-away” and the team started again from scratch, including defining the requirements using traditional methods.

In this project the whole team was working at an unsustainable or “death-march” pace, including the on-site customer.
5.4.4 SUMMARY.

This story focused on some of the issues faced by companies in an outsourcing environment. One of the key issues highlighted was the need for an up-front scope definition phase for fixed price contracts, irrespective of the development process. The second issue discussed was the Winner’s Curse; that of being the successful bidder for a contract that will result in a loss, or very little profit, for the winning vendor. The Winner’s Curse is a prevalent issue in IT outsourcing, irrespective of the contract type or development process, and significantly impacts the project environment.

Developers who find themselves with the Winner’s Curse should consider carefully what they want from an agile process. One could argue that XP could have mitigated the impact of the Winner’s Curse if relationships could have been built on the ground between the customer and developer teams, but again, the climate of mistrust typical of projects in this situation makes this difficult. Additionally XP brought to the fore the scope shortfalls within the first couple of iterations, but the climate resulted in these warnings being ignored.

5.5 PROJECT SPEEDWELL, SWIFTCORP

Project Speedwell was to rebuild one of their “bread and butter” retail sales systems at SwiftCorp. The team spanned two locations: the United States and Canada.

5.5.1 WHO IS THE CUSTOMER?

Project Speedwell had an identified (and formal) customer team. It consisted of:

- **Product Manager** or customer team lead, he coordinated the vision for the product, including working with the two supporting project teams within this programme of work. The product manager was an IT professional with over twenty years of development experience. He led the customer team and was the identified on-site customer from the developers’ perspective; he was based in the United States with the developers.
- **Business Analysts** who assisted the product manager by writing stories, agreeing acceptance tests and working closely with the Canadian based subject matter experts. There were three business analysts assigned to the Speedwell team and they were all based in Canada.
- **Customer Coach** who worked directly with the customer team, assisting them in transitioning to XP, and provided a sounding board concerning any of the problems they encountered with both the larger organisation and developers as they undertook their job on the project.

Additionally there was a larger informal community surrounding the customer team including:

- **Subject matter experts** both technical (architect) and business who advised the customer team of the current existing system, their needs and expectations. The subject matter experts, all six of them, were also expected to liaise with the larger business units they represented to “sell” the project and ensure their advice was on-track.
- **User interaction team** who worked closely with the customer team to help them decide the work flows and usability of the system during story development.
- **A developer** who for an iteration or two moved out of the development team to work closely with the customer team to help them write stories.
- **The project manager** who provided ongoing liaison between the other related projects and business units within the organisation.
- **The executive manager** who held the original vision for the project. This executive left SwiftCorp partway through this programme, before we did the interviews. From other interviewees we can deduce that he was a strong proponent of both agile and the programme. It is unclear from the interviews if another champion from the executive management team
took on this programme. It appears as if once the programme had been sold as a concept it was left almost entirely in the hands of the implementation team, most notably the product manager, project manager and customer coach.

5.5.2 What Happened?

Speedwell grew out of a successful pilot project for both agile and the project concept. The initial team of eight was spread throughout the Speedwell team and the two related project teams, allowing three teams of well-over twenty team members in total to be quickly established across the two sites (United States and Canada).

The customer team worked to create a release plan for Speedwell:

“So the first thing we did was break up all the functionality into the six releases schedule and from simple to complex, trying to bang out some of the more complex things [early] where we could ...”

— Customer, SwiftCorp

The development team worked with the customer team to create tee-shirt size estimates\(^\text{15}\) of this high level functionality, so that a release plan could be established. Once an initial big picture was established they were able to work on an iteration level, doing what they perceived to be as “just enough” analysis prior to the iteration planning game:

“So now that we’ve got this master plan, and as we work out … how we schedule our iteration, we look at the upcoming feature sets, gather all the requirements around the feature sets, hopefully they are all defined and break those features into titles, so all the different stories will comprise those features and then because we have got several different business analysts and myself for writing stories, then we break up who is going to write the stories. I decide if the titles are appropriate. We have a series of review sessions that during the two week iteration process, so we all agree on the titles, break up those titles and then go off and work on the stories, come back as a team, review it with ourselves along with subject matter experts, both technical and functional subject matter experts, make sure that the stories reflect functionalities required and toward the end [of] that we bring the business in, so we … sell the stories and validate again that what we perceive is what is required by the business and then we feed it to the development team”

— Customer, SwiftCorp

So, the customer team attempted to work one to two iterations ahead of the development team. The observed iteration planning game was a standard XP planning game, there was a lot of good discussion occurring between the customer and programmer teams. The whole team discovered that in order to encourage collaboration within the iteration planning games, it was more effective to have two planning games, one focused on the US team and the other focused on the Canadian team. A representative attended via videoconference the other site’s iteration planning game to ensure the big-picture and the larger team retained learnings. Once inside of the iteration, the customer team and development team followed the standard XP process.

---

\(^\text{15}\) The team estimated a piece of functionality as being small, medium or large (i.e. tee-shirt sizes). As is typical with this approach all small pieces of functionality were allocated a time value (e.g. a week), all medium pieces of functionality were allocated another time value (e.g. two weeks) and all large pieces of functionality were allocated a third time value (e.g. a month).
It is also worth noting that during the initial three-months on this project a number of programmer staffing issues occurred:

“... everyone saw right away the poor [performance] scales of the team ... and so the business was able to decide hey look, we can’t carry these people forever, and so they made a major staffing change and they pulled four folks off and they put four different folks on. And then since then there’s been a little bit more turmoil ... one of the people that they brought in as a replacement was also having trouble, so they pulled him and they brought in a different [person] ...”
— Programmer Coach, SwiftCorp

These staffing changes resulted in the programmers moving slower in the first three-months than expected. However, many interviewees noted the intense pressure on the customer team during this period, so it is hard to see how they could have survived with a larger number of effective programmers.

We interviewed the team after two successful releases, approximately three to six months into the project. This project was cancelled after another six months or so:

“I think it was cancelled because all development was outsourced to [Outsourcing Organisation]. The decision had nothing to do with the status of the project.”
— Customer Coach, SwiftCorp

5.5.3 HOW WERE REQUIREMENTS GATHERED, EXPRESSED AND PRIORITISED?

The customer team were building a replacement system, and so there was extensive knowledge of the system domain within the team, as well as an existing understanding of the key drivers for change (i.e. why SwiftCorp was replacing the system). This allowed the team to approach the requirements gathering process with a top-down decomposition approach, where the team created the initial plan and stories, and then involved the subject matter experts and larger business stakeholders to modify that initial draft.

One of the aspects emphasised in the interviews was how the understanding of the story often evolved during the iteration, and how important it is a customer knew how to make decisions concerning the scope of each individual story within the context of the larger release plan:

“... now that you’re going through the acceptance tests you realise that this is a little bit different than what you expected. If you want to add this particular ... sub ... [it] will cost you half a unit, so do you really want to do that or do you want me just to do the simple done thing and just sign off on the story. ... when the customer is presented with those options, it’s worked out really well for us. Sometimes they will say yeah, that’s pretty important, I know ... I’m the customer, I can see twelve weeks ahead, I know what’s happening, so yeah I want you to spend the extra unit right now to drum this out. Sometimes they’ll just say it’s not worth getting involved, just do some simple thing that works here, so just fix that up however you need to, I’ll sign off and we’ll go on.”
— Programmer Coach, SwiftCorp

The team elaborated on how important these conversations were, and also how essential immediate feedback was in order for iterations to complete without “hang-over”:

“[accepting the story:] ... what we intend is for it to be immediate. In practice ... the customer sometimes will be too busy to work through acceptance tests ... so we’ve wound [up]... on a couple of iterations with these dangerous situations
where we’ve got [a] couple of days left in the iteration and there’s six stories up on the progress meter that show that they’re finished, but they need customer sign off ... the usual process is as soon as the customer sits to do sign off on the acceptance test, they’ll notice that something’s missing and they’ll have to say to the developers oh, you forgot this one thing. And so it would often become a bad spot ... where the customer sat down to say okay, I’ve finally got some time, now let’s walk through all the acceptance tests on these six stories, oh you’re missing these two things on this story, these two things on this story, these two things on this story and pretty soon you’ve got a day and a half work to do on a total of six stories that you thought were finished. And so we’re still kind of fighting that a little bit.”

—Programmer Coach, SwiftCorp

Finally, the team noticed the change between the customer and programmer dynamic, requirements were no longer “thrown over the wall” and in many respects that removed a safety-net from the programmers perspective:

“I think a classic thing is ... that developers are used to having the out of saying that all the requirements weren’t clear. And as long as the requirements weren’t clear it doesn’t matter if I fail because I kind of always just blame the fact that the requirements weren’t clear. And so I’ve seen developers struggling a little bit with wanting, ... clearer requirements when really [it] ... doesn’t make so much sense ... because the customer is right in the next room, if you have any confusion about what the requirements are, well you just go ask him, he’s right here. So there’s this kind of ongoing conflict I think ... developers will read this story ... [think] I can’t exactly understand what to do for this story ... if only the customer could make the story ... a little bit more clear, there’s a little bit of grasping for doing things the old way. On the other hand, a complaint that’s a little bit more legitimate I think is that invariably because the customer is some sort of important business type that’s plugged into a lot of other people and a lot of other groups, it is an ongoing struggle I think to see they’re available or not for the developers. That sometimes it will happen that [a] developer has a question, they go over to the customer, oh the customer is not there, they go back in an hour, not there, and oh the customer’s really busy, and so that’s something that I think we’re struggling with ... they know that if they can get hold of the customer, they can get their questions answered. And so that’s worked out, really well.”

—Programmer Coach, SwiftCorp

5.5.4 WHAT WAS THE CUSTOMER’S EXPERIENCE?

The customer lead was very positive about many aspects of XP, particularly with it’s ability to help mobilise the teams rather than be stuck in “analysis paralysis”:

“[the programme] is so immense that teams were pretty paralysed in how we get started, and this [agile] has really enabled us to chunk it up and ... focus, just creativity, rigour ... and a rhythm to help all of us to get our arms around it, so I don’t think we would have succeeded in [the] timeframes without [it]; it wasn’t my idea to use this methodology, when I heard about it, I sensed immediately here’s a methodology that we have to embrace, without it, we are doomed for failure.”

—Customer, SwiftCorp
The initial feedback from the senior executives within SwiftCorp was also encouraging, at the time of the interviews, they were very pleased with the predictability they were seeing from the team:

“The feedback that we’re getting from the customer is the director of this group for every release goes up to [business unit] Canada and does a little PowerPoint show for the executives. And the feedback that we’ve got from the executives in Canada is this is the first time that anybody ever said they were going to deliver some set of functionality by a particular day did. That’s never happened ever in this organisation. And we just had a second release. And she went up and did the same thing again. And I got the same feedback again, oh this is the first time this has ever, you’ve done it twice in a row.”

—Programmer Coach, SwiftCorp

As mentioned in the previous sections, there was also a good relationship and dynamic being built between the business and technical teams, with ongoing conversations occurring instead of “throwing requirements over the wall.” This allowed the customer team to plan and evolve their understanding of the system over time, as well as retain a “big picture” of the project.

However, as with other projects we have studied the sustainability of the customer side of the whole team was questioned, and in this case, not just by the customer team but by the developer team as well:

“... it seems that XP has put a lot of pressure on our customer ... They’ve got to write enough stories to keep developers busy for an iteration, which is a phenomenal amount of work. They’ve got to, in addition to that, continue to interact with the business and all of the usual ways that they’re expected to interact with the business and they need to be communicating with the developers a lot ... to answer questions and also sign off on stories ... I could really easily see those guys burning out and, you know, after release four everything would be fine and successful, but the customer would be in a straight jacket in a padded room beating his head against the wall.”

—Programmer Coach, SwiftCorp

“I think that for the development team [XP is] a better methodology ... I’d love to be a good developer in this methodology because it is broken up in a way that they don’t have to interpret a lot and it’s a pace where they sign-up and you say it could be this number of units and then they have to run to that, so they are productive in the time that they have, everything is very well planned. For the customer team, it’s a huge endeavour ... a more frantic pace that the customer team is running, and yeah, that’s a tough one ... it’s unsustainable right now and it’s going to remain unsustainable”

—Customer, SwiftCorp

The lead customer left SwiftCorp, shortly (two to three months) after the interviews. While there may have been other factors at play, given the concern expressed by a number of team members about the unsustainable load on his shoulders from this project, it is highly likely that the unsustainable nature of this project role contributed significantly to his decision to leave SwiftCorp. The project was cancelled shortly after his departure.

5.5.5 SUMMARY

At the time of our interviews, Speedwell was perceived as a predictable project that was allowing the business to both plan long-term and evolve their understanding of their detailed requirements
in collaboration with the technical development team. The team scaled up their team size by breaking the work across multiple teams. The larger team used a standard XP process, tailored slightly to allow the distributed nature of the team to work.

The customer was not a single individual, but a formal customer team, with an identified customer lead. Additionally a larger informal community supported the customer team. However, there was a significant load on the customer lead and the other customer team members, and there was a growing nervousness within the team as to whether it was sustainable. A few months after the interviews the customer team lead left the organisation. Shortly after that the project was cancelled and all IT development was outsourced.

5.6 PROJECT CALYPSO, OWLCORP

OwlCorp builds systems to support its employees to work in their domain of health. This section tells the story of Project Calypso, a successful project undertaken by OwlCorp. Calypso was attempted three times; we refer to these attempts as Calypso I, Calypso II and Calypso III (the last successful attempt).

5.6.1 WHO WAS THE CUSTOMER?

The identified on-site customer on this project, at the time of the interviews, was a pair of end-user representatives from the affected business unit. One of the end-user representatives had been involved in an earlier version of Project Calypso, although only on the fringes, creating acceptance test scripts based on the original documentation. He was an experienced person in the domain, and had spent over two and a half years in the specific business unit under-going automation where the typical tenure was only two years. The other end-user representative was new to the project, and had no previous exposure to IT projects. She did, however, have significant domain experience.

The customer pair worked very closely together, including sitting side-by-side. They split their responsibilities vertically, as although each had a whole picture of the domain, each also specialised within one of the two-specific areas of the domain under-going automation. This vertical split and their respective domain knowledge meant that they were able to support each other; and at times even substitute for one another, e.g. during annual leave or sickness.

The customer pair was responsible for the user stories created that defined the system to be built, communicating this vision to the programmers and accepting that the functionality met their requirements (including running User Acceptance Testing with a group of users prior to releasing the software). However, they were supported on this project by a larger customer team:

- Two additional part-time end-user representatives, one based in Europe and one based in the United States. These end-user representatives were able to elaborate and refine the system in order to ensure it met the diverse international regulations and user base. The representative in Europe was particularly important as the country she represented had an existing system that met all of their needs, and the new system, while ensuring international consistency would be a “step backwards” for her country.
- An experienced business project manager. The business, based on their experience with the previous incarnations of Calypso, decided it was essential to have their own Project Manager who could help guide them and advise them on scope and push-back with the IT development team. In effect, help them to ensure a realistic vision was agreed, that is, something that could be delivered and still meet business needs.
- An experienced lead acceptance tester, who was officially responsible for the full system and integration testing of the system. Unofficially she provided much of the support and guidance needed by the customer pair on a day-to-day basis that allowed them to undertake
their jobs. She also organised and tracked the stories and their status through the
development cycle. Unfortunately she had recently left the project when the interviews were
undertaken, many interviewees commented on the gap left by her departure. None of the
remaining acceptance testers stepped into her guidance and administrative support role, and
when asked to explain the relationship between the customer-pair and the acceptance testers
there was a clear split between the lead who had been perceived by the customers as within
their team and the remaining acceptance testers who were perceived by the customers as
developer focussed.

- **A big boss.** The senior business manager responsible for the previous projects remained
intimately involved with this project, giving it significant attention and priority so as to
ensure that it did not meet the same end as the previous incarnations of this project had done.
She attended requirements prioritisation sessions as well as remaining on the Steering
Committee. She remained openly available in order to make decisions and provide business
push-back when required.

At the time of the interviews one of the end-user representatives had decided to leave OwlCorp.
The business team was actively recruiting his replacement at the time of the interviews; there was
no intention to reduce the customer-pair to a single end-user representative.

### 5.6.2 What Happened?

Project Calypso is the third project in the same space for OwlCorp over the last five years. The
first incarnation of Project Calypso, Calypso I, used a non-agile process:

> "Three years ago when we first started doing ... [the Calypso] development [it]
was quite a lengthy, document-focussed process. We put three full-time business
people onto it. We spent the best part of a year really writing requirements in the
documents. That is what ... we were asked to do. And then when it came to it,
the developers couldn’t really work with the documents ... Discussions needed to
happen ... about what this meant and what that meant ... and through that
discussion ... new ideas came up. It was perceived ... as scope creep."

— Big Boss, OwlCorp

At this point OwlCorp recognised that there was a problem, and their IT department decided to
look for an alternative and chose to explore Agile (XP), and a second incarnation of Project
Calypso was born, Calypso II. OwlCorp partnered with a consulting firm that specialises in agile
software development; but retained the large scope of the first project. Approximately seven
months later the project was again in the position of insufficient budget for the envisaged scope:

> "I think it was the last throw of the dice really ... but it was too big, and too out
of control, even for Agile to come in and sort out. So we stopped it. We ran out
of money."

— Architect, OwlCorp

> "When it became obvious to us that the date was not going to happen, we just
said forget it. Forget it. It is no use to us if you keep going on and on and on.
We would have had to go back for more funding. We couldn’t justify it. The
money had gone down the drain. Millions of pounds."

— Big Boss, OwlCorp

The consulting company was let go, as was most of the contract development staff brought in to
supplement OwlCorp’s existing development capability. The business representative that had
stayed with the project over the last two incarnations left the project at this point although she
remained in the organisation.
So Project Calypso III started life where tensions between IT and the business were high and a large loss of project memory had occurred. The following sections will explain how the business and technical teams within OwlCorp formed and delivered a small set of functionality from the original scope of Project Calypso.

**Getting Approval To Start.** The goal and scope of Project Calypso III was envisaged by a senior manager within the IT department, based on his knowledge of the business needs developed over the previous 3-years and his knowledge of what might be possible to deliver in a 6-month timeframe. The 6-month timeframe was perceived as essential given the history of the Calypso project to date; the aim was to deliver something small that would help meet a specific business need:

“Everyone was down. Senior management had to go back to the business … cap in hand [and say] it’s failed, we must take a different view, we must revisit the project … I don’t know how he [senior IT manager] … managed to convince them that we needed to turn our attention to [a small scope of the original requirements]. He argued his case and won, and project [Calypso III] was initiated.”
— Architect, OwlCorp

“I think our [IT] management wanted to show that in a short time we could deliver something tangible … What we had to do was take a subset of the existing stories we had from the previous project, and try to get a what we thought was a representative set for this new project, and then … with maybe a few additions and edits here and there, produce this beta. Then off the back of that [beta] demonstration … get the business involved …there was a little bit of suck it and see about how we were going to integrate the business against the sort of predetermined baseline of requirements …”
— Development Team Leader, OwlCorp

Effectively from the IT groups perspective they were trying to demonstrate that they could deliver something workable for the business, despite the previous years of ‘failure’. The business representatives’ memories of the same event provide an added insight into the depths of the rift between IT and the business organisation:

“We saw it [the beta] as a cynical way of getting the Steering Committee to agree to giving money for the project … the beta looked very nice, it was very pretty, but effectively it was, you know, fields on the screen it wasn’t really that full of anything terribly exciting …when they showed it to the Steering Committee they told them that [it] was 80% done, and it was 80% of the fields on the screen, but there was no way it was 80% done, and we felt a bit deceived …

... it was just a different perception, from a business person when you say you’ve got 80% you are expecting ... validation ... and [it to be] solid and [it] doesn’t crash ... so I suppose we started from two different points [of view] ... and that was what ... a lot of the beginning of the project was, hammering out, ... what each of us was ... expecting from it.”
— End-user representative, OwlCorp

**Planning the Release.** Once the project had official approval, the OwlCorp team followed the advice of the Agile Coach and held a kick-off meeting:

“The kick-off meeting was very good, because there were people like me who were new to the project, and there were also people who had worked on the
project last year ... it was quite a mixed feeling. I was really positive, even though I knew things hadn’t gone well, I thought well this is a new start, a fresh start. We can do really well; but there were people [who] were a bit negative [too] ... It was like a sounding ground to address everyone’s fears and get everyone up to speed and get everyone to have the same understanding of how we were going to move forward”
— End-user representative, OwlCorp

The team then implemented the next set of advice from their Agile Coach, which was to have a customer training session, called a bootcamp. The purpose of this session was to help the business people understand their role in the process. Attendees included the business representatives and representatives from the development team. The bootcamp covered what agile was about, and what it would mean to them, as well as how to write stories and prioritise functionality:

“I got signed up for it, ... for some bootcamp, and wind up things with balloons on it and it was great and they took us through all of the Agile methodology and it was quite interesting ...”
— End-user representative, OwlCorp

The team finished the start-up phase of Calypso III with a two-week story writing exercise. The whole team held workshops together (the business and developers), they worked closely together to write the stories that would form the initial release plan for Calypso III. The full development team was not ramped-up for the project yet, so the story writing sessions did not include all programmers. Their approach was to have the development team lead attend all sessions, and the remaining programmers already on the team rotated their attendance, between the story-writing sessions and the iteration zero technical set-up tasks and adjusting the beta. On the last day(s) of this two-week exercise the business, including the executive sponsor, worked with the development team to do a story prioritisation exercise using MoSCoW:

“There was one day when we put all [the stories] out on little tiny little bits of paper ... with their ... Moscow [prioritisation rating] written in there ... it was in the prioritising ... meetings that we started to get a kind of feel for how far we’re going to get to ... So it was a really amazing meeting ... we ended up with, I can't remember how many of stories there were in the end but hundreds of these stories and we just sat them all on the table and ... we made a lead table, and put them in order of their screens and then, you know, we prioritised them per screen if you like so we ended up with, you know, a column with five in it, a column with three in it and a column with 20 in it I think but very much the top ones were the kind of key ones and we did that - a separate list for the must haves, a separate one for the should haves and a separate one for the could haves and one for the what [not] haves and what [not] haves are pretty just kind of ignore really because by that time we were sure we weren't getting them. Yes, I suppose it was basically working out what order of stories to run and we basically said to the techies, do you like with the must haves and should haves because we have to have them... So I think it worked well that way of doing things and, you know, the lead table approach. But again, you know, it was kind of we just never really had a kind of full picture of exactly where the bar was going to be ...”
— End-user representative, OwlCorp

16 “Must have”, “Should have”, “Could have” and “Won’t have” this time - http://www.dsdm.org/atern/techniques/moscow-prioritisation/
This prioritisation process formed the basis of the release plan. As you will see from the next quote, there was quite a difference of understanding of what that meant, as the business expected all of their “must haves”, and most of their “should haves” with a few “could haves”:

“One of the key things which did happen was the meaning of these ratings. That was one of the things which tripped us up later on. There was a miscommunication somehow. I am not quite sure how it happened. When [IT person and myself] put together a presentation on the proposed scope of what we were attempting to do, we were working on a principle of [having] a fixed budget and fixed timelines to deliver your system. So this is a time-box development rather than a feature-box development. We are going to give you as much as we can in that timeline, and we will get the stories that fit in ...

The business took away a different understanding ... they took away that their system should be at least comprised of their must-haves and their should-haves. We didn’t actually meet on that understanding until three or four months later. We were going, what do you mean, you need all of those things? We weren’t going to give you that. Oh but [the business thought that] these ratings mean ...

— Development Team Leader, OwlCorp

The business had ruthlessly prioritised during the initial prioritisation session, as their experiences with Calypso I and II had taught them to. Approximately three months later it became clear that the development team could not deliver all of the “must haves”, and the business was left trying to evolve their understanding of their needs and reprioritise their requirements. This was a very difficult situation for them.

“... right at the start we thought we thought we were still getting our must haves, should haves and could haves ... there was no point rolling out without all the should haves for example ... at about iteration five or six we were quite worried as a business about the velocity ... in the end we said we can't rule out because we've got must haves and should haves irrespective, you know, I know your timelines going to run out but it isn't worth rolling [out] otherwise, you know ... there's no point ....”

— End-user representative, OwlCorp

The business worked closely with the development team to resolve this situation. The end-user representatives worked hard to ensure the good relationships established with the programmers survived the storm. The executive representatives worked hard to ensure the tough decisions were made, and the business did not receive a product that would not be fit for business use.

The remaining aspects of the on-site customer practice were standard XP or covered in previous cases.

5.6.3 WHAT IS THE CUSTOMER’S EXPERIENCE?

Calypso III delivered functionality to a business unit for the first time in three years, the previous three years of IT project work had not released usable software, so on the whole the business representatives were really pleased with their experience as the on-site customer:
“Obviously it is the first time we have done it. I would say it has worked pretty well. The co-location and all the rest of it. That is a definite plus. We didn't have that ... well, we had it a little bit before when we were doing the other process, but it was very much we give you a bit, give us the document and then we will develop it.”
— Big Boss, OwlCorp

“I think this way of working so closely with the technical team is good. I think there are an awful lot of benefits with this process. It is nice that ... it was a completely joint effort in developing functionality. It was ... we would compromise, discuss things, share ideas, and it was really like one big team. I think sometimes it is harder with other people.”
— End-user representative, OwlCorp

They enjoyed the creative and collaborative whole team approach they discovered with Calypso III and XP. The project was not easy and there were a number of difficult and tense periods in it from a customer’s perspective. Most notably, the unexpected scope reduction process that occurred a few months into the project when the team’s velocity was significantly slower than expected. The impact of this scope reduction appeared to have been under-estimated by the development team, as it caused very serious friction and concern throughout the business unit concerned, and reduced their trust in the IT department.

However, unlike every other case we have studied, in this case the on-site customer practice recommended in XP was perceived as sustainable. The end-user representative who was leaving OwlCorp commented that he only stayed as long as he did at OwlCorp because of the project and he was sorry to be leaving the project. However, to further his career in this domain he needed to move to another organisation.

5.6.4 SUMMARY

Calypso III was a typical XP project that worked well and was perceived as a success by the organisation. It is the only project we studied where the on-site customer practice was perceived as sustainable, and it provided much context and insight into the other data collected, looking for the roles and practices that were turned up to an ‘extreme’ level on this project that could help ‘combat’ the sustainability issues uncovered in earlier cases.

5.7 DISCUSSION

In the six case studies presented, the project characteristics vary, but we have found that the customer role is consistently under significant pressure. The commercial settings vary. Endeavour was a new internal management systems, outsourced by a large established company. Discovery was a new scientific system, developed as a product by a start-up company. Atlantis was a new version of an IT development system by an established company in that area. Pinta was a new business transaction system that would also evolve to become a product within FalconCorp’s product suite. Speedwell and Calypso were both internal replacement systems developed in-house.

The alterations made to XP also vary. Endeavour had already done some work on requirements determination before committing to XP. The Discovery customer had helped found the company, worked as a developer early on, and later was also the development manager. Atlantis had already designed a new technical infrastructure before committing to XP, and would involve their QA team only late in project. Pinta reduced the interaction between the customer representatives and the programmers, given the uneasy climate between the organisations.
The kind of person playing the customer role also varied according to the commercial nature of the project. In Endeavour, the customer was a domain expert from the client organisation. In Discovery, the customer was a founder of the start-up and, as typical in start-ups, had to play several roles as the company took shape. In Atlantis, the customer was the product manager, and hence in close touch with existing and prospective purchasers and users of the new product. In Pinta, the customer was a team consisting of both client customer proxies and product managers. In Speedwell the customer was a team of business analysts, with a recognised customer lead. In Calypso the customer was a pair of end-user representatives strongly supported by business unit management. In each of the cases, the customer seemed well-prepared to play the role, and seemed a sensible choice for the particular kind of project involved.

In each of the successful cases, XP appeared to be working well enough, and the projects all seemed to be looking successful, with some credit being given to the XP process. In the challenged case of Pinta, XP appeared to be giving all of the warning signs that the project was in trouble very early in the process, however these signs were apparently ignored, and indeed XP became a scapegoat for what appeared to be a case of the winner’s curse.

Each of the customers did use some strategies to cope with the workload. For example, the Endeavour customer was supported by a more senior manager, an acceptance team, and an analyst involved early on. The Discovery customer was supported by one of the company founders, a research scientist involved in demonstrating the product to potential buyers, a team of scientists verifying the software, and an experienced software tester. The Atlantis customer was supported by a UI designer, a technical architect, a technical writer, and a QA team. The Pinta customer team included analysts, testers, and product managers. In other words, each of the customers was assisted by what initially seemed a sufficient support structure.

But in the experience of the person in the customer role, there was a conspicuous commonality. In ten of the eleven cases, the customer was under stress, and committing long hours, to fulfil the role. In only one case, Calypso, was the customer perceived as a sustainable role.

5.8 Summary

Customers don't write code or refactor it – developers do that; they don't make the key decisions about the project – in any large business, rather more important managers will do that; nor do they tackle the key technical or process problems with the development – the XP Coach does that. Rather the customer determines the requirements of the business and communicates those with a single voice to the development team and then sells the products of development back to the business.

One of the key principles of XP is that "business people make business decisions, technical people make technical decisions". So phrased, this sounds like an equal balance. On the other hand, the customer must be the business's main interface to the development team. This requires the customer to keep in touch with the ongoing evolution of the business – XP allows development to keep track of rapidly changing business requirements, but the conduit for communicating these rapidly changing requirements to development is once again the customer. Simultaneously, the customer must maintain the project's credibility, and funding within the wider business environment. In all but the smallest business, the customer will have to answer to other business stakeholders – managers, employees, boards etc.

XP practices support the development team in making the technical decisions that are their responsibility. XP includes very few practices that actually support the customer in their role – other than proscribing how they interact with the developers. Furthermore, the practices – primarily the Planning Game and interactive nature of XP – effectively tie the Customer to meet the technical rhythms and needs of the development. The customer is supposed to provide stories
or tests or advice or clarifications to the development team instantly, whenever they are required. In practice, it is far too easy for anything that is not to do with programming, or that is not explicitly covered by XP’s practices, to be defined as a "business" requirement, and thus, to become the sole responsibility of the customer. As in Project Discovery, it is easy to see why a Customer may spend forty hours a week working with the development team, and another forty hours dealing with everything else.

We saw that the involvement of multiple organisations required accommodation of the different cultures of the organisations, and presented challenges when implementing the XP Customer role, particularly in a fixed price contractual basis. These findings show us that for XP to embrace change, an organisational and contractual context is needed that allows change to be embraced without penalty.

The existing XP Customer practices appears to be achieving excellent results but they also appear to be unsustainable, and so constitute a great risk to XP projects, especially in long duration or high-pressure projects. Finally, in all of the cases that we studied, the on-site customer practice was fulfilled with an informal team of people forming around an identified customer or customer-pair. In the next chapter we will explore the make-up of that customer team.
6. The (Informal) Customer Team

In the last two chapters of this thesis we have seen the emergence of a team of people working together to fulfil the on-site customer practice that was initially outlined by Beck [2].

Further analysis concerning the composition of this team finds that ten roles emerge from the data. By role we refer to a coherent set of responsibilities backed by a sufficient skill-set that assist in the successful operation of the project. Some of these roles were already well-identified roles in the literature, such as “Acceptance Tester” but many are new roles, arising directly from this research. A role may be played by more than one person, and a person may play more than one role. We identified the following roles:

- **Collaboration Guides**: Geek Interpreter, Political Advisor and Technical Liaison
- **Skill Specialists**: Acceptance Tester, User Interaction Designer and Technical Writer
- **Direction Setting**: Negotiator, Diplomat, Super-Secretary and Customer Coach

Each of these roles directly supports the success of the on-site customer practice in some way, but the specific roles needed will depend on the nature of the project. We depict the structure of these roles in Figure 6-1. The diagram emphasises how the direction setting roles are the primary roles suggested by the on-site customer practice, and how the collaboration guides and skill specialists support the direction roles to undertake that practice.

![Figure 6-1: The Customer Team Roles](image)

In this chapter we will describe each of the ten roles and how they emerged. All of the roles, unless specifically noted, were played by members of the project team. Firstly, we will describe the collaboration guide roles, followed by the skill specialist roles and finally we will describe the direction setting roles. We present the roles in this order as it is the order that they emerged during the analysis. We first understood the detailed supporting roles and later the direction setting roles.

---

17 Please note that the names of the roles (e.g. Geek Interpreter) have been selected to enhance the “grab” and memorability of the role for practitioners. As outlined in our research method chapter, chapter 1, this is an important aspect of grounded theory, and we have tested the names with over 300 practitioners over a 4-year period.
6.1 COLLABORATION GUIDES

Collaboration is an essential value of agile software development [30], and as such significant investment is made in the facilitation of collaboration around agile projects. Highsmith [37] suggests in his agile method, Adaptative Software Development, that the role of a collaborative facilitator should exist within the team to facilitate team communication and collaboration. Likewise, the roles of XP Coach and Scrum Master emphasise helping establish connections between people and reconnecting people within the team [2; 38]. We noticed that specific roles emerged to enhance the business-technical collaboration on a project.

The first of the roles, the Geek Interpreter, focuses on internal project team collaboration. The next two collaboration roles have more of an outward facing collaboration focus. The Political Advisor role emphasises the navigation between the project and the larger business world. The Technical Liaison role emphasises the navigation between related technical areas within the organisation, be they related projects or technical specialist silos within the organisation.

6.1.1 GEEK INTERPRETER

Geek Interpreter: A technical person who supports the business to improve their communication and collaboration with programmers.

The agile principle that business and technical people are required to work together daily [2; 31; 32] changes the dynamic of the business-technical relationship in many organisations. This principle generated a lot of positive discussion in the interviews, for example:

“[Previously] you write a document, you get feedback on the document, it goes into a traditional waterfall ... technical design specification, and it’s not until the product comes out at the end that you realise, well somebody screwed something up .... But you get the pleasure through XP of having that close, close, close relationship in defining the product right at the time ... one of the great powers of XP for me was that ... I could leverage ... the collective intelligence of the whole developer group that I work with ...”
—Customer, EagleCorp

“The close involvement of the business without a doubt that’s a big plus to this process ... because you’ve got them there and then and they’re dedicated to being available to answering your questions, to work with you ... they know what you’re working on and they know how the system is evolving so you can be quite confident that they’re not going to get a shock at the end of the process saying what kind of pile of rubbish have you built for us, this isn’t what we wanted ... well you know they wouldn’t say that because they’ve been involved from the word go and they’ve seen it week by week so they know exactly what they’re getting and they’re there to influence ... shift directions [if they are not happy with what they see].”
—Programmer, OwlCorp

“The business people are with the technical people, and there is constant communication. Every tiny bit that has been developed has been developed with
very close communication between the two groups. I think that is what has made it so good.”
— Customer, OwlCorp

The emphasis on business and technical people collaborating was perceived as fundamental in order for the team to “build the right thing”. We also observed that there was something else being referred to here, as programmers and customers sometimes talk past each other. For example, we observed in one case that a customer did not truly understand the implications of an estimate discussion during a planning game. None of the team members present assisted the customer to understand the types of questions to ask in order to drill down into that discussion in order for them to understand the trade-offs being discussed. However, we noticed that one of the roles emerged to directly address this (and similar) situation(s):

“If I’m kind of thinking, how can you give me a four day [estimate] … for a field on the screen, that’s bonkers, I’ll wander up and say [to a developer he trusts on the team], I’ve got a story, what do you think? … and obviously I couldn’t then quote it back to the first developer but it gives me an idea of what’s going on …”
— Customer, OwlCorp

“If I’d had time to get more involved in the technical stuff, you know, and know what all the jargon means it would have helped the end result because I’d be able to see the implications … [instead] I’d go through detailed things with [the developer she trusts].”
— Customer, KiwiCorp

We have named this role the Geek Interpreter. The Geek Interpreter is a person who helps the customer understand and talk with programmers. The Geek Interpreter generally does not talk to these programmers directly, but instead acts as a sounding board and coaches the customer to speak to developers more effectively. It also emerged that the Geek Interpreter often provided guidance or advice on how to approach tasks such as story writing, user acceptance testing, planning and prioritising:

“With [the Geek Interpreter], … it’s very informal … So I suppose at the moment we just kind of tap on the shoulder [for the] little things that we need help with like for example I’m trying to … organise how to do one of the … [UAT] time trial script[s] … I’ve no idea where to start [that] kind of thing so I’m going to ask her to go through that with me for the first bit and then sort of fix it kind of thing.”
— Customer, OwlCorp

This last aspect of the Geek Interpreter role was only seen when the customer did not have a development background. Customers with a development background typically did not need this type of advice or guidance.

We observed situations where the customer used a Geek Interpreter both when the customer’s background was solely in business as well as when the customer’s background also included development. The only situation we observed where a Geek Interpreter was not in evidence was when the official customer was also playing the role of a programmer on the team. In this instance, they picked up the Geek Interpreter role for themselves and also for the larger stakeholder community. The team all agreed that one person playing both the customer role and programmer role was not something they would recommend, or do again themselves. Not having the separation of these roles lost some of the business-technical dynamic, and the team noticed that it resulted in a technical skew to discussions and decisions that they later regretted. While it
is not the case that the person playing the role of the customer cannot have a technical background, we found people valued a separation of concerns.

We noticed that most programmers value their communication skills with business people, and it might bother them to realise that at times the customer needs to use a Geek Interpreter when communicating with them. As such, it is particularly interesting to note that the customers seem to be aware of this and are very careful when using a Geek Interpreter. For example, one customer mentioned that they never used the advice provided by the Geek Interpreter in potentially antagonistic ways like “but [Geek Interpreter] said …”. The customer’s careful use of the Geek Interpreter has three core considerations. Firstly, the customer did not damage their relationship with the programmers, who might feel insulted that the customer needs a Geek Interpreter when collaborating with them. Secondly, the customer did not damage their relationship with the Geek Interpreter. Misusing the Geek Interpreter’s advice might result in them no longer feeling comfortable providing assistance to the customer. Finally, the Geek Interpreter’s relationship with the programmers remains undamaged, and this could have a significant impact depending on their role within, or relationship with, the programmer team.

Words used to describe the person(s) playing this role included “guidance”, “trust”, and “on my side”. While these terms were also used to describe the team as a whole, it was clear that there was a specific person or persons to whom the customer felt an additional connection. The Geek Interpreters tended to have some attributes in common:

- Comfortable having conversations with the business representatives, and interested in learning about the domain, the “big picture” or goal of the project from a business perspective, as well as the details of a specific story.
- Able to laugh at themselves, develop rapport with people and did not take offence easily.
- Able to listen to the business and show respect.
- Care about their work and take pride and responsibility in it.
- Enthusiastic and have a positive, can-do attitude.

We noticed that in some situations, such as KiwiCorp and EagleCorp, the customer had a strong connection with a single Geek Interpreter. In OwlCorp, four Geek Interpreters existed, each one providing a slightly different nuance and assistance to the business–technical relationship. The number of Geek Interpreters at OwlCorp grew over the course of the project, as the customer was able to build trusted relationships within the team.

Finally, it is important to note that the Geek Interpreter role was never observed to be an “official” team role. We noticed that their official roles were varied and included business analyst, tester, project manager or programmer (on either the same or on a different project).

Summary. A Geek Interpreter is a useful find for a customer, and if one is not readily available then a customer may cultivate and grow his or her own Geek Interpreter within (or external to) the team. The purpose of this role from the customer’s perspective is as a way to enhance their understanding and relationships with their development process and the development team.

6.1.2 Political Advisor

Political Advisor: A person who is aware of the political dimensions within the organisation and is adept at navigating these dimensions to assist the project to succeed.
Every organisation, be it large or small, has a rich organisational life. During our interviews a number of stories that illustrated the richness and diversity of organisational life were shared, and we include some of these here to help illustrate the role of the Political Advisor.\(^{18}\)

One senior business executive related her ongoing frustrations with regards to working with IT. The examples she used to illustrate her frustration gave us insight into the richness of organisational life in a large international organisation. She believed firmly in a strong relationship between business and IT:

“There is often an issue between the business and IT. It may be a perception, but you often get the feeling that the people in IT are saying the business don’t know what they want, and the people in the business are saying they are trying to dictate what we should do, trying to say we think we should do this, and this is the issue and this is how we should solve it ... That whole relationship between business and IT, I think, is key. If you can get that right, then I think the rest follows, to be honest.”

However, despite her trust and personal relationship with the immediate development team, on the larger organisational side the relationship between her business division and IT was very strained. The current IT manager had worked with the business unit on a multi-year project that had been cancelled. She described the point the business unit realised the project was not going to deliver:

“[Regarding a one day steering group meeting that her boss had flown in to attend] ... We had all of the discussions, ‘we’ll still deliver ... we are still working to these dates.’ [Boss left meeting to catch plane back overseas]. I had to ring him and say: ‘you know all of these things we have just agreed to, well [the IT manager has] just told me that all the staff were being let go, the money was running out. So there is no way they are going to be able to deliver’ ...”

Although during the meeting the IT manager did not indicate that they would not be able to deliver, the executive manager and her boss were unable to see how they could deliver with no staff. Her boss sent a memo stating that it appeared delivery was now impossible and that they needed to rethink.

“So then there was this period of mourning. We were thinking, ‘what shall we do?’ We were thinking, ‘shall we just go externally, because we don’t trust what we’re being told?’ ... which is kind of the worst part of it really. Then it gets into this whole political issue of what are the consequences. If you pull the plug and say ‘look how much money was wasted’ and try [to] explain what’s happened, somebody ends up getting fired. That’s the difficulty ... and then you have to be prepared to be responsible for it.”

Following through on that story, the organisation did find a way to move forward and delivered a small part of the original vision for the project – and that was the project under study – but the business executive’s analysis of why they did not pursue the option of removing the IT manager was interesting:

“Before [the IT manager] took on [name of IT Unit] his predecessor was pushed out of the company, and my boss ... was kind of partly the catalyst in that. [Regarding the current IT manager] ... So I think he [her boss] thought it was..."

\(^{18}\) In this section we do not identify stories or quotes from organisations using a pseudonym. The stories are sensitive, and we wish to avoid explicitly situating the stories within the cases that we have described earlier, to protect the privacy of different interviewees from the same case study.
déjà vu, and just thought it is the same thing all over again. You don’t want to be responsible for two people in the same position. So we just lived with it, and I think because he has lived with it, probably because of the history there, [he] hasn’t been willing to ... formally raise some of the issues, and we can’t go above him to the CEO. It has to come from a senior person. He is quite a tough guy, he is not a walk-over, so I try to take my direction from him, and I just constantly tried to work through it all of the time when we hit something.”

How did this project history affect the team? Perhaps in many ways the project history had positive consequences, in that this senior executive worked very closely with the team. While there were some rough patches along the way, she was complimentary about the team and was pleased with the software that was produced. However, she was finding that continuing to “work around” the IT manager was frustrating and time-consuming, and so despite the team’s best efforts, she noted that:

“[Organisation] is not a software development company. My gut feeling is that if I wanted to [do] software development again, I’d go to a software company. Honestly that is what I feel. It is too painful to try and do it in-house.”

Six to twelve months later the team was mostly disbanded and the software development was (partially) outsourced.

In this example we can see that the business executive took her lead on how to play out a situation in a complex organisational unit from her manager. We can also see that this situation was outside of the control of the direct team. Perhaps if they had been more aware of the organisational life beyond their immediate project they may have been able to avoid being outsourced.

In another project, it became painfully aware to the project team that one political player had been overlooked: operations. The IT Development Manager describes the disastrous “go/no-go” meeting that occurred:

“An awful go/no-go meeting we had — you have to hear that story, it’s very good — we were supposed to go live on the 17th of May, that is a Sunday. So on Friday, we had three go/no-go meetings. One at 9 o’clock in the morning, one at 12 o’clock noon and one at 3 o’clock in the afternoon. And at 9 o’clock in the morning it was all lights green, everything’s great. At 12 o’clock, all lights green, everything’s great and at 3 o’clock ... we didn’t even have to have that meeting ... and then operations [raised a number of unexpected objections to the go-live approach and indicated they were unhappy and didn’t want to go-live] ... and then the customer started ... ‘Well ... if that’s your opinion then we are not saying go.’ Like – that was horrible, really, really, really frustrating and if I had known it was a problem, I could have fixed it. I mean, it was fixable and it wouldn’t have to influence [the go/no-go decision].”

On reflection she realised that when assembling the key people to involve in the project steering group, she had opted for a junior member from operations as he was available, but he did not have the authority required to make the decisions necessary in this situation. The people he represented needed to be intimately involved rather than her expectation of simply needing to “rubber stamp” his decisions. She had misread the political dimensions and had no additional advisor on the lookout for signs of unrest within the organisation.

The IT manager was able to fix the problem. She worked with a colleague within the operations division to identify and lobby the key people within the operations division to ensure the system was ready to “go live” two-weeks later. However, it turned out she found this solution too late,
and after incorporating the changes, an error occurred on the operations side and the project finally went live a month late. The project was delayed, and portrayed within the larger IT organisation as a failure due to the delay, despite the fact that the project team delivered working software that added business value. All that was remembered within one (powerful) part of the organisation was that the team was a month late. The customer had missed a key political player within the organisation and the long-term perceived success of the project was compromised.

The IT manager could have prevented this situation by earlier involvement of her operations colleague in this task to help identify the people to lobby. We have named this role the Political Advisor.

Cheng [67] also relates an experience where two different units within an organisation have different goals or measures of success for a project. In the situation he describes, the IT organisation of an investment bank focused on high-uptime, full redundancy, disaster recovery and zero defects; these goals made sense for the majority of the bank’s systems. One business unit, the subject of the experience report, however, had very different priorities, as they needed functionality delivered quickly to meet market pressures. The project team were initially very successful from the business unit’s perspective but not successful from the IT unit’s perspective. Over time the team reduced the functionality produced in order to meet the IT unit’s goals, leaving the business unit unsatisfied. Cheng did not have an answer to this situation, but his experience continues to show the complex political context that agile teams operate within.

While many of the stories from our case studies demonstrate the richness of organisational life or politics with “negative” outcomes, some demonstrate how it can work to achieve positive outcomes for the project. One customer from a large project shared a story of working closely with her executive manager to help her (the customer) know when to break the rules:

“Well we knew that if we actually got people’s formal sign-off — business people sign-off for everything — we’d never actually get anything done. So we ran a bit of a risk. We assumed, based on [senior executive’s] work with people and our knowledge that we [could proceed without formal sign-off].”

This project was seen as a success within the organisation despite breaking the rules. The customer used her executive manager as a Political Advisor.

From the stories related so far, it may appear that the customer must use a senior executive to assist them to navigate the by-ways of organisational life. There are, however, also stories that indicate that Political Advisors can be outside of the formal organisational hierarchy as well. One end-user representative remained in contact with the person who had been the previous end-user representative on the project. He relied on both her advice and organisational connections, and the advice and connections from a senior manager to help navigate the project successfully though to completion. So, in this situation the end-user representative had two Political Advisors, one with end-user and the other with management connections and awareness.

Cockburn [27] outlines the software development project “co-operative game principle”. In this principle he emphasises that the primary goal of a project is to deliver working software, but it is important to remember that the secondary goal is to set the stage for the next “game” or project. Politics and organisation culture remain “in play” much longer than a single project, and customers should use the advice of their Political Advisors concerning both the primary and secondary goals outlined by Cockburn.

A team lead on one team also discussed how the use of a Political Advisor can have unexpected negative consequences for the development team:
“[Customer] was a bit naïve and just passed on everything he heard directly onto [Business Manager]. Because [Business Manager] wasn’t necessarily always on site, and always directly in communication with everyone, she did make a couple of kind of information releases to the Steering Committee which were taken in a very negative way. It was quite early on in the project. She just released information early, just fired it off to the Steering Committee, and it caused a lot of earthquakes and ground-shaking. Luckily most of it didn’t get down to the team. But it was because of that, because of uncensored information being passed through, used out of context, things like that, and it set up a bit of an environment of tension. Because of a lack of communication I think. ... I think it contributed significantly to [Project Manager] going off ill with stress. He took most of it to heart ... I think that was a very negative thing. It makes everything harder when you have to kind of like walk away to have a discussion somewhere else out of earshot.”

So there is perhaps a delicate balance between what to share informally and how that information is then communicated to the organisation formally, particularly in the early stages of both the project and the internal team relationships.

Summary. Organisations of more than one person have politics. Customers, as we discovered in the previous chapter, are by their nature involved intimately and continuously in a development project. This means they cannot by themselves keep up with the organisation’s politics and power structure, and the impact the project will have on the organisation. It emerged that customers recruit one or more Political Advisors whom they can trust. The Political Advisor helped the customer work out who the political players actually were, both official and unofficial, and to identify and understand them. Finally, the customer needed to constantly access the advice and guidance of their Political Advisors and ensure that they were “plugged in” all of the political dimensions necessary for the project to succeed.

6.1.3 TECHNICAL LIAISON

Technical Liaison: A person who undertakes the liaison with related projects and technical silos within the organisation.

Most projects do not exist in isolation; project teams have to interact with existing organisation technical infrastructures and other software development projects. This role emerged very quickly in the studies, initially because it was “missing” from the first case we studied, KiwiCorp, and so caused significant overload and frustration for the customer:

“We probably needed about three of me ... it’s been my life ... look at these grey hairs ... My life would have been easier if I could have been 100% devoted to requirements and testing ... but ... all these technical integration issues were just taking up about half [of my] time ... And they really shouldn’t have been but we didn’t have a very strong technical integration leader, so I was doing ... in hindsight I was doing far more than I ever should have, and probably not very effectively because I mean I’m not technical ...”

—Customer, KiwiCorp

The project manager on the development team also commented on this issue in our interviews, and he noted this situation caused a number of bottlenecks for the development team. One of the
issues that complicated this situation is that the project involves multiple vendors. So, while the project manager would have liked to help the customer, the outsourcing contract his organisation had with KiwiCorp did not allow him to formally undertake the liaison activities with their other vendors. This example illustrates the need for Technical Liaison between software development teams and technical specialist activities such as infrastructure.

KiwiCorp was not alone with the need for a Technical Liaison between the project and specialised technical groups:

“... different teams ... always comprise a project, you’ve got the UI, you’ve got deployment, there’s getting the applications out, initiatives etc., you’ve got security, you’ve got infrastructure, ... the interaction between different systems, so you’ve got all these different groups that you bring together in a project.”
—Customer, SwiftCorp

However, at other large organisations that we studied, OwlCorp, EagleCorp and SwiftCorp, it emerged that “project managers” from the IT side of the organisation tended to undertake the Technical Liaison role. Although the customer remained aware of the impact to the project, it appeared that it fell to another person to actually liaise within the technically silo-ed organisation structures (e.g. architecture, data, operations, facilities, testing).

The liaison undertaken at SwiftCorp seemed particularly effective. At SwiftCorp the project manager and coach invited (and encouraged) the technical specialists to attend the planning games, and daily stand-up meetings. The project manager and coach were both very aware that they needed to make it worthwhile for the technical specialists to attend, so they were careful to initially invite them only to sessions where they would get significant benefit. In their experience of using this approach, they found that over time the specialists became more involved, by choice, as they could see the benefits of being involved based on their experience. However, the technical specialists did not attend all meetings, simply the ones that directly impacted them. For longer meetings, like the planning game, their involvement was normally scheduled to occur at a certain time during the longer meeting. The project manager complemented this with a lot of one-on-one liaison with the technical specialists. Honious [68] from Reed Elsevier describes a similar approach that their project team used to involve their operations group actively in their agile project. On the Reed Elsevier project, the team identified a primary contact person within the operations group and then invited him to participate on their project one day a week. They also noticed that their contact person gradually increased the time he spent with the team to 2-3 days a week.

Additionally, as we studied more cases, it also emerged that the Technical Liaison needed to be broader than simply liaising with technical specialists; related projects were also a key part of this role. Again, we will illustrate the emergence of this aspect of the role with an example of it being missing and the frustration that it caused to the business:

“Well, there is something ongoing at the moment which is to do with some software that was due to be released into the production database through the support team in June. It has been delayed because it ended up being a lot more complex to develop than it was anticipated... I think it has to be rolled out before system testing, or after system testing, I am not exactly sure, but there is a dependency with the tool. If that doesn't happen when it should have happened, then basically it means we are going to have to shift everything back and we can't start our UAT [User Acceptance Testing]. If we can't start our UAT when we are supposed to, our resourcing goes up the swanee because we have got people coming in from specific [business units] around the world on specific weeks to do specific things. So it is a knock-on effect. Now, this particular [change request],
“when it was raised that it is going to be an issue if it is delayed, I said okay, let's get it sorted. We can't afford for this date to slip because the impact means it could be a three week delay, just the way things can work out.”

—Big Boss, OwlCorp

In the quote above the executive manager is frustrated. Not so much that the support team tool development ended up taking longer to develop than expected, but that no-one understood, or more importantly planned to avoid, the downstream ripple effects. Liaising with the different projects earlier might perhaps have provided the opportunity to rebalance the focus of the different teams to assist with that task so as to minimise the overall delay. Her problem and frustration stemmed from the isolated view the team had on their individual projects, with no-one (except her) liaising between the different teams. Hodgkins and Hohmann [69] recommend the introduction of an Agile Program Management Office (PMO) to assist with this situation. While the focus of the Agile PMO is on gaining larger organisational agreement on relative project priorities, as a side effect it also helps manage resource sharing and cross-team dependencies. The frustrations experienced by the OwlCorp executive manager might have been significantly reduced, as the Agile PMO would have provided her with the single point-of-contact she was looking for to assist her with the dependency implications she felt were left to her. So, while in our data the Technical Liaison role was undertaken by someone on the team, it might fall to an external group such as an Agile PMO.

In our case study data, however, it emerged that typically the project manager undertook this liaison as well as the liaison with the technical specialists. So, the second of the key responsibilities of the Technical Liaison role is to liaise with related or dependent projects. Again, working to ensure the cross-project communication occurred so as to reduce delays. For this second aspect of the Technical Liaison role, we noticed that the customer or product manager would take a much more active role than they did for technical specialist liaison. For example, on the EagleCorp product redevelopment version 7.x both the project manager and product manager liaised with their counter-parts from the earlier 6.x version of that product. This more active interest in the related project liaison was driven by the scope of the projects, ensuring scope overlaps or missing scope gaps between the projects did not occur. Lowery and Evans [70] describe a similar need on the BBC iPlayer project. On this Scrum project, they discovered that they needed two regular forums: a “Scrum of Scrums” that provided information dispersion and discussion opportunities concerning project dependencies amongst project managers and a similar forum for the Product Owners concerning the functionality overlaps and gaps amongst the related projects. Finally, it should be noted that the related projects we noticed were a mix of both software projects and business projects.

The Technical Liaison role seems to be particularly complex when multiple vendors existed and the liaison crossed organisational and contractual boundaries (e.g. KiwiCorp and TernCorp). Of particular interest is that in these situations the role often fell onto the shoulders of an already overburdened customer. In cases such as SwiftCorp and OwlCorp, we see a situation where some of the team were consultants or contractors. However, these people were seen as individuals and so appeared to be “adopted” by the organisation and did not encounter the same issues truly outsourced projects encountered.

The time commitments of this role varied substantially. For OwlCorp it was estimated that it took more than one full-time person to undertake this role: a full-time IT project manager, and a part-time business project manager. Similarly at SwiftCorp and KiwiCorp it required more than one full-time equivalent person undertaking the role. For EagleCorp, a much smaller organisation, it featured less as none of the people considered it to be their main activity. The time commitments do vary, and it appears likely that the variation is caused by the size and complexity of the organisation, number of technical specialisation areas and number of related projects.
It is notable that in no case did a programmer pick up the Technical Liaison role. This may be because in XP programmers are actively encouraged to focus on getting their stories completed. Additionally, many conversations we have had, as an agile coach, with technical infrastructure specialists would suggest that although programmers may have the technical expertise, they do not always have the same perspective as technical infrastructure specialists. For example, technical support people can find it difficult to deal with programmers, who sometimes do not understand the outward ripples of simple programming changes.

The Technical Liaison role can sometimes be a formal role on the team, particularly in organisations with large IT departments or in organisations with outsourced operational facilities. In situations where the IT departments are smaller and operational facilities are in-house, this may be a part-time role that is picked up by a member of the project team, and most likely someone with the role of project manager.

A number of authors have also identified a need for cross-team coordination and communication on agile projects [68; 69; 70; 71; 72; 73]. As with our case study data, the research papers and experience reports that concentrate on these issues tend to be larger organisations: Nokia [71], Verisign [69], BBC [70], Reed Elsevier [68], Primavera [73], and Salesforce [72]. Suggested mechanisms include those we have already briefly covered such as the “Scrum of Scrums” and an equivalent forum for Product Owners, and an organisational unit such as an Agile PMO. Other suggested ways to handle this issue are more technique focused, and we cover these here as the Technical Liaison may wish to utilise some of these techniques when undertaking this role. Kahkonen [71] summarises four communities of practice that Nokia has established that bring technical specialists or related project teams together in facilitated workshops. The four communities cover: elaborating and specifying requirements, planning and tracking subsystem development, software integration and, finally, managing the requirements and architecture of subsystems. Tartaglia and Ramnath [73] relate how Primavera have approached a similar situation with a different technique, they use open spaces to help bring interested technical specialists or related team members together to discuss a specific topic.

It may be debateable as to whether the Technical Liaison role belongs in the “customer team” as it certainly crosses the boundaries of business and technical decisions. It is, however, definitely a role that focuses on organisational collaboration and in some cases is at least partially, if not fully, picked up by the “customer team”.

Summary. Projects do not operate in isolation. Projects often depend on technical specialized skills (e.g. infrastructure) and there are often inter-dependencies with other projects, both IT and business, that are being undertaken at the same time. To achieve the business goals, the project team needs to keep abreast of and communicate with these projects and specialist groups. We observed a Technical Liaison role emerging within teams to meet this need.

6.2 SKILL SPECIALISTS

Our interviews made us aware that some skill specialists aligned more closely with the customer role because of their role in establishing and confirming requirements. The customer team is responsible for non-coding activities within the software development project like writing stories, accepting stories and writing user documentation. We have discovered that skills specialists often exist on the team, who assist the customer in undertaking these tasks. The three skill specialists discussed in this section are the Acceptance Tester, User Interaction (UI) Designer and finally the Technical Writer.

These specialist roles do not suggest that only the person playing that role can undertake the specialist activity, but they do encourage a specific focus and accountability [74].
6.2.1 **Acceptance Tester**

Acceptance Tester: A person who helps choose and write acceptance tests to verify the software.

Acceptance Testers\(^{19}\) exist in both the non-agile and agile world. Beck, in both the first and second edition of XP, includes the specific role of “tester”\(^{2;32}\). The aim of the tester is to assist the customer to both choose and write acceptance tests. In the world of XP, however, there is a strong emphasis on the automation of tests and much attention is paid to automated tools to assist with this task\(^{36;75;76}\). In the design of Fit (Framework for Integrated Tests)\(^{77}\) Mugridge and Cunningham specifically considered the role of the customer in developing and applying tests. This represents an important new direction in acceptance testing, but still involves a level of technical expertise to apply successfully.

In many of the organisations we entered, testers had an existing function within the software development space. In some they had an easy transition from a traditional software development approach to XP, and joined a cross-functional XP team from the start:

> “I used to look at what they had developed, and it was completely different from what we wanted. It was really hard. [Developer] put all this work in, and sometimes he hadn't asked for any business input ... It is very hard to face someone, 'that's very good, but it's not right, change it please', without really demoralising him ... But [Acceptance Tester] ... could see that what had been developed wasn't right, it wasn't going to work. It wasn't going to work from a business point of view ... She did have a lot of discussions, but not obviously facilitated, she would just sit round with both of us, and we would go through and try and test the acceptance criteria, to see where it had failed and try and word things differently. She was very good at that, and she had a lot of impact ...”

— Customer, OwlCorp

In others cases, such as EagleCorp, there was a long “stand-off”. For most of the EagleCorp development the official Quality Assurance (QA) team were too busy testing the previous version of the product to examine or assist with the current version. Additionally, QA hesitated in integrating their testing process into a two-week time-box, as XP was initially seen as “engineering only”. The QA lead was invited to, and did her best to, attend all of the “touchpoints” with the team, such as the bi-weekly planning game. While she did assist the process by providing a QA perspective, her team’s existing commitments meant that fuller participation was not possible. This situation caused some tension between the two departments. Over the course of the development effort, however, the engineering manager worked to fulfil that role with the product manager; and the practice of (almost) daily acceptance test sessions was introduced. These acceptance sessions were held near the end of each day, and involved the engineering manager, the product manager and the user interaction designer. Any stories that were marked as “development complete” would be reviewed in that session for acceptance\(^{20}\). The “true” QA did occur at the end of the project; and the QA manager was impressed with the quality achieved by the team. Furthermore, she had many ideas how to involve her team earlier so that the product

---

\(^{19}\) This study is not an in-depth study of Acceptance Testers or Acceptance Testing, we only briefly touch upon this topic as it relates to or is perceived by the customer.

\(^{20}\) Initially, at least, not all stories were accepted.
was even better next time. The engineering manager’s strong commitment and enthusiasm about building a quality product spread across the team, and his support of the QA manager resulted in a more integrated approach to testing in the next version of the product.

In other organisations, such as RavenCorp and RobinCorp, there had been no role of Acceptance Tester, and for a period of time the programmers worked with the customer to undertake that role. Later on, both of these organisations adapted their team so that an Acceptance Tester was recruited and adopted this activity as a full-time role:

“[after working for a while without a tester, they thought] ... this is a lot of work, we need to get somebody in here to do this full-time, and so we brought in [Acceptance Tester.]”
—Customer, RavenCorp

The Acceptance Tester was one of the most obvious skill specialist in our data. In the case where the role did not exist previously, it was the first specialised skill role to be added. Programmers appreciated the addition of someone on the team with this focus, as did the customers. It was rare, even for detail focused customers, to have the time to see through all of the detailed testing required to ensure the product had been verified:

“I have been pretty busy with a lot of different things going on ... I have a lot of confidence that [Acceptance Tester will] be very diligent about making sure all the little things are captured and stuff”
—Customer, RavenCorp

“Sometimes the testers would fail it when we’d passed it because the testers have an amazing sense of how to break things, you know, they’d put in random characters and things, you know, just kind of testing with what we wanted ... so that was quite good, they complemented each other quite well.”
—Customer, OwlCorp

Hussman [78] and Andrea [79] also report similar findings of trust from business representatives in the Acceptance Tester(s) on the project. Andrea [79] elaborates further that the Acceptance Tester also has the responsibility for creating a testing strategy working with the customer and the larger team to understand the trade-offs concerning different tests, and prioritising which tests the team should undertake, and how those tests should be implemented (e.g. manual, automated, or not done). Hodgetts [80] describes the adoption of acceptance testing in two organisations, and how testing gradually moved to the front of the process, and their teams became test-driven with the Acceptance Testers working closely with the customers during requirements elicitation and definition, so truly being part of the customer team. Other papers [81; 82] provide further evidence that this approach of moving to test-driven executable specifications was helpful, from the perspectives of the customer, the developers and the testers. Finally, Geras [83] reminds us that Acceptance Testers cover exploratory manual testing too.

Hussman [78] describes his experiences of customer teams, and he also finds that the role of Acceptance Tester is an important role in the customer team. It also emerged from the case study data, however, that some Acceptance Testers were much more closely linked to the customer team and some to the development team from both the programmers’ and customers’ perspectives. So, having the title “Acceptance Tester” does not automatically mean you are part

---

21 During the period of this project, the company had gone through a down-sizing round and the QA department had been affected significantly. The QA manager was somewhat concerned with the lack of value placed on the QA function by the organisation. She appreciated the support that the engineering manager provided over the course of this process.
of the customer team. For example, when interviewing the OwlCorp customers towards the end of our interviews we collaboratively drew a picture of the whole team. We included the different people on the team and looked at who was unofficially on the “customer side”. One of the Acceptance Testers was definitely drawn close as belonging to the “customer side” but the remaining Acceptance Testers were seen solely as belonging to the development or technical side of the team. The differentiating factors seemed to be that an Acceptance Tester had a strong interest in the domain, and worked closely with the customer as their support, helping them explore their ideas or needs at a detailed level.

**Summary.** It emerged that the project team needs to ensure that someone has testing experience and is prepared to take on the role of Acceptance Tester on the project. Real testers understand testing, are good at it, and take the customer’s side. The role of Acceptance Tester may or may not be a full-time role on the project, but we have found that it tends to be not only a full-time role, but also a formally recognised role on the project team. It is essential the Acceptance Tester role be perceived both by its undertaker and by the rest of the team as belonging to the customer team, otherwise the project ends up with too much technology-focused testing and insufficient business-focused testing.

### 6.2.2 USER INTERACTION DESIGNER

User Interaction Designer: A person with user-centred design skills who helps design what to build.

One of the strengths that User Interaction (UI) Designers bring to the project team is User-Centered Design (UCD). There are some inconsistencies in the terminology used in this area, but we consider UCD as being comprised of three broad but distinct areas, which are briefly described below:

- **User Research** [84], which focuses on understanding who the users are and how they work.
- **User-Interaction Design** [49; 85], which focuses on designing a system that will meet the user’s needs or goals effectively.
- **Usability Evaluation**, which focuses on evaluating how well the system meets the goals of the user.

It is these UCD skills and techniques that the customer team can leverage in order to deliver a system or product that works for the users [84]. As Beck [32] agrees with this view, in the second edition of XP the team includes the interaction designer role. The purpose of this role was to “work closely with the customers, helping them to write and clarify stories”. This supports the emergent role we observed occurring on teams in practice.

From all of our cases, only two, EagleCorp and SwiftCorp, featured a specialist UI Designer on the team. EagleCorp had decided early on that one of the key differentiators for their new product 7.x was usability, and as such, had specifically hired a user interaction designer to join the team. The designer introduced the team to the concept of personas [65], and personas soon pervaded all of their stories and conversations concerning functionality. SwiftCorp also considered the user interface to be a core element of their system, and they worked closely with a user interaction design team throughout the project:

“...we work more closely with the UI team than the development team, because as we work through how we are going to do this set of functionality ... there is an
The EagleCorp designer initially struggled to adapt to the iterative time-boxed approach used by the team. He would have liked the opportunity to have more up-front design time, but he did work closely with the customer and the team throughout the process. After a few iterations, he moved out of his office, along with the product manager, and joined the development team in their team area. He worked in conjunction with the product manager on deciding how to design the user interaction or experience for the stories, as well as for the “big picture” approach. The developers quickly came to see him as part of the customer team, and it was important to get his approval, as stories would not be accepted unless he was assured that the interaction was perfect. The coupling between the product manager and the user interaction designer was quite tight, and in fact, approximately 18 months after the interviews, the user interaction designer went on to become the product manager for the product.

Often the UI Design appears to be Big Design Up-Front (BDUF), because the UI Designers do their own iterative design and evaluation for usability. Ferreira, Noble and Biddle [86] also discovered that significant up-front UI design work was undertaken prior to the programmers joining the project. One of the most important aspects of this up-front design was the user research aspect and user-interaction design of UCD. However, they also discovered that the overlap (where designers and programmers worked together and iterated through the UI) was critical to the success of the project. Miller [73] writes about the Alias experience of combining agile with UCD. Like Ferreira et al. she also writes of an initial up-front design aspect for UCD, she calls this “cycle 0”. The teams then divide into parallel tracks, with the developers working on “cycle n” while the UI specialists focus on the user research for the cycle n+2, the design for cycle n+1 and the evaluation or usability testing of cycle n-1. Like Ferreira et al. and Miller, we found that UI Designers, in conjunction with the programmers, are learning together about how to make this work in the incremental fashion of agile iterations. While in this section of the thesis we are not attempting to argue for up-front design, we are, however, suggesting that in order for the customer to “build the right thing” UCD skills and techniques are essential.

Patton [87] noticed that commercial software where users can “opt-in” is where usability or interaction design thrives, and this fits with the commercial software development of a product such as the one developed by EagleCorp. SwiftCorp also paid significant attention to the user interaction as they have large numbers of temporary and low-paid users, who needed good interfaces to ensure quick sales response times and minimal errors. Most of the other cases we explored do not fit in this space and have not had specialist user interaction designers. This does not, however, necessarily mean that no thought has been paid to the user’s experience. For example, in SparrowCorp the Business Analysts were very aware of the users interactions with the system and how to create an experience that would work best for them. They modelled their interactions on Excel and would at times create quick Excel prototypes for the users to try before building the system.

Fox, Sillioto and Maurer [88] support our finding that UI design can be approached either by a dedicated specialist or generalist(s). In their research they noticed that teams approached UI design in three ways. The first approach is when a specialist designer worked on the team with UCD skills and experience, this approach is called the specialist approach. EagleCorp and

---

22 There was, in fact, some “grumbling” about the perfectionism, particularly in the early stages, until the developers understood what was required in order for the story to be done.
SwiftCorp appear to have followed this approach. The second approach is when one (or more) of the generalist members of the team with an interest in UI design would introduce and adopt UCD techniques on the project, this approach is called the generalist approach. SparrowCorp provides an example of the generalist approach to UCD on an agile project, although in this case, the generalist was a business analyst instead of the programmer Fox et al. may have expected. In the final approach, the first two approaches are blended. A UCD specialist works on the team, but this specialist also has development skills, and so also operates as a programmer on the team as well. We did not observe this approach in the cases, although perhaps EagleCorp moved in that direction as the UI Designer moved into the product manager role. Chamberlain, Sharp and Maiden [89] have studied the overall experience of blending Agile and UCD. Of their five principles for integrating UCD and Agile Development, one is particularly pertinent to our research: the importance of real collaboration between the members of the team, and the need for the Customer to be an active member of the whole team and not just a passive bystander.

**Summary.** It emerged that UI Designers feature on the customer team, providing UI advice and guidance to the customer team. Their UCD skills are of particular value in understanding the user, and helping to design an effective experience for them. In the case where UI is seen as critical, this role is likely to be a full-time recognised role on the project team. In all other situations, this role is likely to be performed in conjunction with another role on the project team. One especially important reason that UI work be aligned with the customer is that usability may lead to new stories or acceptance tests being introduced to the project.

6.2.3 **TECHNICAL WRITER**

Technical Writer: A person with specialist technical writing skills who produces the user documentation to accompany the released software.

XP downplays technical documentation but *user* documentation is still important. Programmers do not tend to have the technical writing skill-set required to write effective user guides. We noticed that Technical Writers often end up on customer teams, particularly when the application is a software product.

In EagleCorp, we discovered our first and only “official” Technical Writer on an agile team. At EagleCorp the Technical Writer was assigned to the customer team on a part-time basis, and was often assigned to multiple projects. The Technical Writer said the following about agile development:

> “I love [XP], because it really lends itself well to doing documentation. I mean usually tech writers have to wait until there is some sort of [final] build before they can really start. You know they do a lot of the initial work off specs ... It’s really difficult, ... [as] you really don’t know what you are documenting until very close to when your shipping it. [But with XP] ... I see the build every single day, [and] I am continually going over my docs”
> — Technical Writer, EagleCorp

In traditional software development, the Technical Writers often start writing the user guides from the requirements specification. When the software product is delivered a few days before shipping, they then find that the requirements specification does not reflect the software product’s functionality. Long and intense hours are then spent re-writing the user guides to match the
software delivered. In XP, the Technical Writer is able to distribute the load, as the software changes, and continually make small tweaks to the user documentation:

“[after describing in her own words the situation above... so]... if they change a little button, now it’s a month away from ship – it’s not a big deal. The day before, it wouldn’t be a huge deal.”
— Technical Writer, EagleCorp

So, in agile software development, Technical Writers have a less intense workload over the last few weeks before shipping, they are required to evolve their technical documentation regularly. We must be aware that this situation could quickly become frustrating for some Technical Writers and may result in them pushing for a more defined specification up-front, in order to better balance the amount of work and required re-writes over time.

Beck also noticed the need for the role of Technical Writer on the whole team, and introduces it in the second edition of XP [32]. Beck expands on the role, stating that it is more than just writing documentation, it is about being involved in the process early on, asking “How am I going to explain that?”, and helping the team to create learning and feedback loops from the perspective of users. This aspect of the Technical Writer role did not emerge from the case study data but it resonates with our own experience coaching teams that include Technical Writer skill specialists.

Heimgartner and Locke [90] write about the experiences of involving Technical Writers on two different agile projects. In the first of the projects, the Technical Writers did not embrace agile development. The Technical Writers felt like ‘second class’ citizens given the focus on ‘working software’ as the only deliverable that measures team progress, and did not adapt well to a co-located team. They ended up adapting the agile process so that the Technical Writers created their own iteration plan for technical writing work, and operated one iteration behind the development team. The changes, however, did result in the developers and Technical Writers communicating with each other much more than previously; this result was seen as very beneficial. The second of the projects is from an experience at BMC Software. The Technical Writers from BMC software operated much like Beck [32] expected. At BMC Software, however, a Technical Writer would need to participate on multiple agile projects. This situation would arise quite often, and the BMC Technical Writers discovered that attending multiple daily-standups and planning sessions was both time-consuming and disruptive, giving them less time to focus on delivering user documentation. So, when considering this role, it will be important to consider how many agile projects they can effectively participate on.

Non-product development projects seem to focus less on user documentation, perhaps for similar reasons as non-product development projects focus less on user experience [87]. This does not mean, however, that it is gone completely. For example, in SparrowCorp the Business Analyst undertook writing user documentation, including training materials. Similarly, with Raven Corp, while the company did not hire a Technical Writer, a scientist worked on creating user documentation and examples of product use, to help users learn how to effectively use the product.

Summary. In software product development it is common to see professional Technical Writers join the team. In situations where professional Technical Writers do not join the team but end-user documentation is still to be produced, it is essential that someone on the customer team have (or learn) technical writing skills. The Technical Writer not only produces the user documentation, but also helps the team reflect and question the impact of the software to end-users.
6.3 **DIRECTION SETTING**

The direction setting roles comprise the core of the on-site customer role and practice outlined by Beck in the first edition of Extreme Programming Explained [2]. It is these roles that set the direction of the project, resulting in the single voice describing “what to build”. The collaboration guide and skill specialist roles support the direction setting roles.

We note that while Glaser recommends against using analogies to develop or explain a grounded theory [91], upon explaining these roles over the course of four years to over 300 people that the analogy is a useful one for practitioners. It appears that the analogy of treaty negotiation provides a good starting point for practitioners, helping them to understand how these roles worked together to create the “single voice” of the customer to the development team.

6.3.1 **NEGOTIATOR**

Negotiator: A person who works with the end-users and other stakeholders to negotiate a single-voice of what to build, and communicates that to the programmers.

DeMarco [40, p. 5] suggested that negotiating “with a whole community of heterogeneous and conflicting users is a gargantuan task”; and went on to liken the Diplomatic skills required to “the skills of a Kissinger negotiating for peace in the Middle East.” We use this analogy when introducing the lead role of the Customer team.

On every project we studied, everyone could clearly identify the on-site customer(s), even though there was an entire customer team. One person, or in some instances a pair of people, were the identified contact point. It emerged that, like in DeMarco’s analogy, the Negotiator picked up the task of gaining agreement within the larger stakeholder community on the vision for the software and communicating that as a “single voice” to the programmers.

For example, the EagleCorp product manager, who played the role of the Negotiator on the Atlantis 7.x development project needed to bring together both internal and external stakeholders. His internal stakeholders consisted of senior executives, sales, marketing, and operational support representatives, as well as the architect from the engineering group. His external stakeholders included the Customer Advisory Group representatives from existing customers as well as representatives from potential new customers. The EagleCorp Negotiator used a number of different facilitated workshop techniques to facilitate an agreement amongst this diverse stakeholder base as to the scope of Atlantis 7.x. Finally, it should be noted that the negotiation aspect was an ongoing activity. As new information came to hand, re-planning resulted; both small changes and dramatic changes to the initially envisioned scope needed to be negotiated and agreed on with the stakeholders. A Negotiator emerged in all of the other cases. We give them the name Negotiator to more clearly define the role, but also to more clearly allow the “onsite customer” term to refer to the customer team.

A number of texts [37; 42; 46; 50] exist that would support this concept of a Negotiator or facilitator working with large or diverse groups of stakeholders to achieve a vision of “what to build”. These texts recommend the use of facilitated workshops, to achieve a shared understanding amongst the stakeholders.
It also emerged that the Negotiator utilised their negotiating skills with the development team as well as the stakeholders. In the following quote, one of the OwlCorp Negotiator’s describes how his pair interacts with the development team:

“She's good at negotiating, she'll quite happily be able to kind of you know assert a point across and that kind of thing and go up to the developers and they saying we haven't got time and she says yes you do, it needs to be done anyway, whatever. She's good at kind of negotiating with them and I think a lot of the kind of second half of the development, a lot of that was negotiations and stuff and she used to do that.”

— Customer, OwlCorp

So the Negotiator not only works to obtain a “single voice” of what to build from the multiple organisational stakeholders, but also continues the negotiation on a detailed level with the development team.

We saw great variety in the background of the Negotiator, the stakeholders they would represent, and the techniques they used to obtain the agreement amongst the stakeholders. The Negotiators’ backgrounds ranged from those with an IT background (e.g. business analysts, developers or project managers) to those that had no IT background at all (e.g. the KiwiCorp librarian). We have found, however, that there were also a number of similarities amongst those undertaking the role of the Negotiator. Firstly, all Negotiators knew the domain well:

“They need to have domain knowledge, huge domain knowledge to be a customer, a strong customer. If they don’t have that it’s hard for them to gain the confidence of the developers to say I’m going to tell you what we’re going to build and you’re going to build what I tell you and that’s the way its going to be. That’s a great customer. Because I’m right, you know. Now for someone to come in here and say well, I don’t know what we’re doing and we’re going to just meander around for a while, that sounds like a lot of rework, scope creep and all those bad things in there that don’t make delivery fun or easy. So I think it’s kind of domain knowledge and a real determination.”

— Programmer, SwiftCorp

Notice that this domain knowledge does not necessarily mean they are end-users or business stakeholders. For example, the programmer quoted above is referring to a business analyst, not an end-user. On another team, however, although the team had confidence in their Negotiator it was evident that they felt a little isolated from the direct users of their product:

“I’ve always felt bad that we never talk to the customer. Like engineers ... don’t actually go out and have meetings with the customer. [EagleCorp] has their Customer Advisory Council ... but we just get features ... so it would definitely be nicer to have more of an interface to customers.”

— Programmer, EagleCorp

Other attributes the Negotiators had in common were that they:

- Understood the business drivers on the project, which may or may not have always aligned directly with their needs as an end-user (if they were an end-user).
- Knew who to approach for information or decisions, thus were well connected within the organisation and able to use their connections effectively.
- Were aware that multiple perspectives existed, and helped people with different perspectives understand one another.
- Were comfortable working at both the “big picture” level and the detailed levels.
• Felt or took responsibility for the success of the software; that is being able to use the system effectively.
• Were able to think or understand the current as well as future possibilities.
• Were confident, decisive, and stable under intense pressure.
• Enjoyed project work and liked working with technical people.

Perhaps unsurprisingly this list of attributes aligns reasonably well with Beck’s initial list of attributes of a good customer [3, p.18]. The attributes that Beck brings up in addition to our list include:

*Is determined to deliver value regularly and is not afraid to deliver too little rather than nothing and can make decisions about what’s needed now and what’s needed later [3, p.18]*

Interviewees did discuss the importance of regular delivery, which occurred through the XP process, but they did not add the aspect of not being afraid to deliver too little instead of nothing. We did, however, notice that the Negotiator was acutely aware of almost the exact opposite, which is that in some situations releasing nothing (rather than too little) may indeed be the right business decision. So it is perhaps reasonable to assume that they were very aware of delivering regularly but their emphasis was more business orientated.

In Scrum the Product Owner sets the direction of the project, and is expected to be a single person, who might or might not be advised by a committee. As we noted in section 2.6 there is a great deal of similarity between the descriptions of the XP Customer and the Scrum Product Owner. Scrum takes its concept of Product Owner from the Product Management community. Like the Product Owner, the Negotiator does have a leadership role. A leadership role is noticed in a number of experience reports [70; 92] where a team forms to fulfil this role. Scrum, however, takes the specific stance that while a committee may advise him or her, ultimately it is the Product Owner that is responsible for making the decisions [38]. On the other hand we noticed that Negotiators do not necessarily actually carry the full responsibility, that is, they will not necessarily lose their jobs if the system or project ends up being perceived as a failure. It is rare for that responsibility to be fully on their shoulders, it seems more likely that it will be on the sponsor’s or Big Boss’s shoulders. What was interesting, however, is the Negotiators we observed did choose to own the responsibility to obtain an agreement amongst stakeholders. These individuals chose to pick up that responsibility. This fits with Beck and Fowler’s [3] initial words, which were “willing to accept ultimate responsibility for the success or failure of the project”.

Finally, despite the fact that there was a customer team, on most projects we studied, it was the Negotiator who was most often clearly overloaded. This has often led to burn-out, or the person performing the role leaving the organisation prior to project completion. The organisation loses this person’s valuable knowledge and the application may suffer once the lead vision-holder is changed.

**Summary.** Business people need someone to help them clarify their vision and ensure that an application gets built that will meet their competing needs, that will be accepted by both stakeholders and end-users as achieving the business goals. Programmers need to know who to talk to concerning their story, they need to be confident that it is the “right” person, and that it is someone who can talk to the programmer in a way the programmer can understand. The Negotiator meets these needs, but they will need to work closely with the Diplomats as we discuss below.
6.3.2 DIPLOMAT

Diplomat: An end-user or stakeholder who brings the perspective of their group to the project.

We continue to use the treaty negotiation analogy for this next role, Diplomat. For it is the Diplomats who are present at the negotiation table. Diplomats are business representatives or technical specialist representatives (e.g. architect). Their job is to represent the interests of their area, whether that area is a country, a business unit, or a subject area. Notice that Diplomats are members of that area, not proxies. It is important that, as Diplomats, they articulate the view of their area well and work with others around the table to hammer out the “treaty” or single voice of the customer.

To represent the views and needs of a group of people the Diplomat must be in touch or connected with this group of people; keeping them in the loop on trade-offs that will have to be made during the process and obtaining their buy-in to the agreement as it takes shape. At times the Diplomat may even choose to bring other representatives to the table as feasible:

“... we were kind of told that yes we were representing and you can't please everyone so, you know, take a best stab but obviously things we don't know the answer to, ... you don't know why a field was originally there ... so we do research to get to the answer and we go and find out who knows because generally in [the business unit] there's someone who's sitting on all the information and that kind of thing ... you have to kind of I suppose put your own kind of preferences aside as well and see what the best is...”
— End-user Representative, OwlCorp

Despite them being members of the area themselves, the Diplomat still needs to connect with the people they are representing. Typically the projects second someone from the area to work with them on the project, so they have a deep insight into what will make a difference, please the people they represent, and get the system accepted:

“... but [Region A] very easy to please if it's new... if it works really well and stuff then It's great and it will look new ... and [that] just that makes a massive difference ... it looks nice and the error messages say something intelligible and it's good, you know, it's got user written all over it rather than kind of I was developed outside of things, you know ... [later concerning the release process with these users ...] they get really kind of bored of the fact they can ring up the support team and, you know, nothing will happen. It never changes. So nobody bothers ringing up anymore so the support team think the whole things working fine and dandy and it actually could have been broken for about three days over there and no one will every know. So I went over the first day and they seemed to be working fine, just to check that everything was kind of...”
— End-user Representative, OwlCorp

They also know the tough sells coming up. For example, in OwlCorp one of the Diplomats was very aware that one region within the company would be very unhappy with the new standardised system:
“[Region B]... It's not the system they're going to rebel against it's the fact they're going to be forced into one box.”
—End-user Representative, OwlCorp

This region’s version of the system was already better than the new standardised system, so for a time (yet to be defined) it was taking a step backwards. So the Diplomat has a very outward facing role into the organisation. Their involvement with the project team can vary significantly. Some Diplomats worked full-time on the project team, and developed strong relationships with the programmers (e.g. OwlCorp). Other Diplomats were part-time and while they developed a strong relationship with the Negotiator, they had no relationship with the larger project team (e.g. most of the programmers on Project Atlantis had not met any of the EagleCorp Customer Advisory Council members).

From the examples we have presented above, it is easy to focus on the Diplomat being the end-user. While the end-user and business representatives are crucial to agile projects [30] and real customer involvement is strongly encouraged [2; 32], other perspectives must also be present to fulfil the customer role. Technical specialists have one of the other perspectives that feed into the single voice the customer must provide. The involvement of technical specialists in the customer team on technically-focused projects might typically be expected. For example, there was a technical advisory group on HawkCorp’s Advance project and on the ChannelAdvisor project outlined by Isham [93]. Technical Specialists may, however, be crucial representatives on business-focused projects too. For example, at EagleCorp the architect was a key influence on the technology decisions made, and non-functional requirements (e.g. performance and scalability). It is important that these types of requirements are not forgotten [78]. At SwiftCorp, technical specialists such as data specialists informed the decisions concerning data standards that must be followed for the software to be accepted. Another perspective that we saw represented was that of package vendors. For example, on TernCorp and FalconCorp some directions were not considered feasible from the package vendor’s perspective, and the direction was altered to reflect that.

Diplomats face a number of different pressures in their role, perhaps many of the same pressures we associate with Diplomats during a treaty negotiation:

- They can be drawn into a team mindset, and lose sight of the day-to-day realities of the people they are representing. This situation appears even more likely to occur when the Diplomat is working full-time on the project. But there may also be a danger of this occurring even when continuing to work part-time on their “day-job”. As the Diplomat is learning to think and make decisions from a project perspective, they are exposed to a different way of looking at the world, and so can begin to see themselves as separate to the people or unit they represent:

  “... there's only two of us and, you know, we'll be working on this totally all the time and ... you just get indoctrinated and so it's been really good to have a fresh pair of eyes on things ...”
  —End-user Representative, OwlCorp

- Project work is addictive, and many of the business Diplomats we met were reluctant to return to their day-job. At least one of these Diplomats later transitioned into the IT departments after her full-time project secondment had ended:

  “... I don't think she necessarily wants to go back to her [business unit] job, I think she quite fancies working with these guys and yes, I think you get quite into it, it gets quite addictive ... [this Diplomat later transitioned into the IT group] ...”
  —End-user Representative, OwlCorp
They can have moments of deep despair when looking at final scope outcomes. As with a treaty negotiation, no representatives at the table will get all of their desired outcomes, compromises must be made:

“[Other end user representative is] good for detail, she's quite enthusiastic as well although she has her moments of deep despair when we're running short of time for stories and stuff but she, you know, she's enthusiastic and I think that's kind of an asset because otherwise when things are going wrong the whole development team kind of goes ah and everyone hates it but [she still appears] quite enthusiastic ...”
— End-user Representative, OwlCorp

Additionally, organisations must give Diplomats time to do project work. A wise customer (or organisation) will not expect the Diplomat to be able to successfully undertake this role while undertaking all of their existing commitments:

“[new skill specialist coming onto project]... he's never done UAT before in his life but he's coming as well. Part of what we did is because we were a bit short on staff and [business unit names] are just all chronically understaffed pretty much, although I didn't say that. They can't spare people so ... [we pulled in this skill specialist to represent them instead] ...”
— End-user Representative, OwlCorp

We have also noticed that initially Diplomats (and the development team) will probably underestimate the amount of involvement required from the Diplomat. For example, in TernCorp one of the Diplomats originally expected to only be available to the project for 50% of her time and the remaining 50% could be spent on her day-job commitments. Her department re-allocated the other 50% of her work. It soon became apparent, however, that the project required her involvement full-time on the project, but there was no opportunity to re-negotiate. Her solution, partly because of her personal decision to take on the responsibility of the success or failure of the project, was simply to work longer hours. The impact to her personal life was significant, as she was working very long hours for an extended period of time, and even with all of that effort, she did not manage to maintain all of her organisation commitments.

Summary. Customer teams require organisational representative(s), who may include: subject matter experts, end users, senior stakeholders, decision makers, and architects, to name but a few. They are responsible for representing their organisational area or perspective on the project. A project will typically involve many departments within an organisation, and will require the perspectives of multiple parties, who will not always see “eye-to-eye”, as each party will have competing goals, priorities, and requirements. To work successfully, the Diplomats must have the time to participate in the project. Their role is significant and includes not only representing their perspective on the project team and (sometimes) working directly with programmers, but also uncovering and representing the diverse views in their area. Finally, the people in this role must be open to negotiation and be able to understand the perspectives and needs of other departments/areas.
6.3.3 **SUPER-SECRETARY**

Super-Secretary: A person who undertakes the administrative support workload of the customer team. This person will also be the person the programmers approach for guidance when the Negotiator is unavailable.

There are many administration and organisational tasks that need to occur in order for the customer to be effective in their interactions with both the business and the programmers. Overloaded customer team members find it easy to either let these tasks “slip” or become a burden that results in them either not being as effective (e.g. losing stories) or working even more hours in a day.

We have found that typically one person on the team will surface to pick up the “administrivia” load from the rest of the team; we have called that role the Super-Secretary\(^{23}\). We have found that the Super-Secretary always has another formal role on the customer team, so this role is always “part-time”, despite the occasionally very large amounts of work involved. The Super-Secretary will typically always record and organise the stories, as well as track them through their lifecycle\(^{24}\). The Super-Secretary also has a detailed understanding of all of the stories:

> “In fact she could have probably done my job. You know, she had a very good knowledge of the business ... so that helped I think ... She also [has] this amazing encyclopaedic knowledge of everything, so you'd say what story was this and she'd trip out the story number and the letter and everything to do with it, just you know like that.”
> —Customer, OwlCorp

The Super-Secretary also undertakes other tasks such as:

- Following up the story status with the programmers
- Printing cards or tracking cards on the wiki, as required by the programmers or business
- Organising meeting rooms for iteration kick-offs or planning meetings

Beck [2] outlined the role of Tracker as a member of the XP team, and this role seems to align with many aspects of this administrative role. There is, however, a focus on the requirements or customer responsibilities of the team. From our perspective, this role is richer than that described by Beck. Beavers [94] described the role “Requirements Architect” that BMC needed to introduce to improve their management, elaboration and prioritisation of the projects requirements. This role appears to undertake a similar function to the Super-Secretary.

One of the remaining tasks that often falls to the Super-Secretary is that of Negotiator or Diplomat stand-in. When the identified “customer” is not available, perhaps because of the close relationship that the Super-Secretary holds with the customer, and his/her detailed knowledge of the stories, the development team begin to use him or her as a “stand-in”, obtaining his or her impressions of a story:

---

\(^{23}\) The formal role that the person holds on the team can range from Business Analyst, Tester, Project Manager and in one case Developer.

\(^{24}\) For example, a few Super-Secretaries used a sticker system with different colours representing each stage to track the stories. While the team as a whole might help put the stickers on the cards, it was clear that the Super-Secretary quietly ensured that this system was in use and updated.
“[Super-secretary], who is our tester/admin/general kicker-up-the-arse person. She is brilliant ... she works very closely with [the Customer] in the testing side of things. When we think we have finished a card, if [the Customer is not] around ... We will talk to [Super-Secretary] about, do you think this is done, and she kind of gives a non-developmental look over what we have done. She is like an informal regression test for us, because she will try and use the application all the time and go through it ... She is kind of like support for the [Customer] in that way.”

—Programmer, SparrowCorp

“[Super-secretary] was a tester, but she did more than testing ... yes, she kind of coordinated a lot of things and she did a lot of documenting stuff. She had [an] overall view of everything ... She used to think in all directions so sometimes she used to catch points that we didn't catch or in fact even business didn't catch. She used to think ‘what about this, what about that?’”

—Programmer, OwlCorp

The Super-Secretary can often become one of the most helpful (although often under-appreciated) people on the team, from both the business and technical perspective. Perhaps it is this aspect that helps draw out why this role, despite its apparent administrative nature, is only undertaken by quite experienced or senior team members.

Given that this role is always a secondary one, and often unrecognised, the person performing it can become very overloaded. Although we have named the role Super-Secretary, there is often a limit to this person’s “super” powers. In one case, the Super-Secretary had become too overloaded and had recently left the project. The team was feeling the ramifications of her departure and perhaps becoming aware of the true load she had been shouldering for the best part of the 12+ month project. It is important to keep track of this person’s load and consider ways to mitigate the overload they will experience.

Over the last few years we have presented this research to over 300 people, and we have discovered that, as hoped, the name Super-Secretary does help people envision what might be associated with this role. It does also carry an additional negative perception, however, as people do not want to be secretaries, even super ones. This became particularly evident in a session in which we used a presentation explaining these roles during a customer team-training day. The customer team was able to use the roles to help them consider how they covered all of the bases. During the session we sensed one person’s discomfort. Outside of the meeting, the individual revealed that she found the role name offensive: she was “not a secretary”, and she “would not be getting people cups of coffee!” Since then we have been actively working on identifying a better name for this role. Unfortunately names such as “personal” or “executive” assistant also retain many of the perception issues associated with the term “secretary”. One name that we considered was “Steward”, based on the analogy of a King’s Steward, which perhaps better illustrates the importance and significance of this role. Additionally it helped illustrate the role this person plays when the customer lead is unavailable. We have yet to completely settle, however, on the name of this role, as “Steward” or similar titles like “Minister” have over the years evolved in our society to reflect more power or ownership than we have seen in this role. For the purposes of this thesis we retain the name Super-Secretary.

Summary. The Super-Secretary ensures the rest of the customer team members are not distracted from their core roles by administration and organisational tasks. The Super-Secretary ensures

---

25 We initially added “Super” to the role name to help combat this perception problem, but it turns out it does not do so sufficiently.
that the organisational and story tracking tasks occur, otherwise stories might get lost, cards run out, and so on. Finally, and perhaps importantly, the Super-Secretary will typically be the stand-in for the Negotiator or Diplomats, should they not be available.

6.3.4 CUSTOMER COACH

Customer Coach: A person who supports the other customer team members to undertake their roles.

XP introduces the concept of a coach: someone who helps the team transition to XP, both from the team and people dynamic and the process perspective [2]. At SwiftCorp the team had two coaches, one that specialised in the development practices and the other who specialised in working with the customer community and the associated customer practices.

The SwiftCorp Customer Coach had a development background, but worked with the customer team, supporting them to drive out and communicate the direction of the project. He was almost 100% customer focused, leaving another coach to focus on the development practices. The Customer Coach, however, was only part-time on the project, approximately three days a week. During that time he was the customer’s personal “cheer” team when it all seemed too much, and would help them determine how to break the task down into achievable steps. He had a lot of experience at writing stories and acceptance tests and perhaps even more importantly the soft collaborative and community building skills needed in the customer team.

“A coach who comes in with a really strong personality is sort of a ... is a born leader and people just want to follow him and people will listen to what you say, then I think the process can work really well, introducing it into an environment. If you don’t have that, if you don’t have the voice that everyone just naturally listens to, it might be problematic, I don’t know because it feels to me that it has ... as the process becomes less about formalised process and more about people, then it becomes more important I think to have stronger personalities than those people, where at least for the people who are playing leadership ... roles. Because in sort of a traditional waterfall process, well you listen to what the project manager says, why do you listen to what the project manager says, well because he’s the project manager and we’ve been through this drill a million times. And in an agile process, okay this is kind of new to me, I would, you know, maybe I’m confused about what I should or shouldn’t be doing, well I’m following that guy, because I’m following that guy, the strong personality, the domineering guy who’s coming in and telling us what to do”

— Programmer Coach, SwiftCorp

Only one other team, RavenCorp, had a person on the team with the title “coach”. He was aware of the customer issues and worked with the organisation to help iron out some of the issues in this space. For example, having no single voice from the customer team was resolved by reducing the three customer representatives to one. The RavenCorp Coach was more focused on the development team and their associated practices, so provided some support but not the dedicated focus that the SwiftCorp Customer Coach did. For example, the RavenCorp coach was unaware that the RavenCorp customer was working 80 hour weeks regularly in order to fulfil his responsibilities. The SwiftCorp Customer Coach was very aware of the hours the customer team was working, but also more importantly was able to discuss and relate the issues highlighted by
the customers, whereas the RavenCorp Coach was much more in tune with the development team than the customer team.

Other teams did not have a dedicated Customer Coach but did instead have a Project Manager playing that role who was working with the team, coaching them in story gathering, writing, involving the right people during the project, the organisational politics, and so on.

Finally, it also became clear that when we were conducting research with some teams, we had an impact on the project, particularly with those teams who did not have a Customer Coach. Someone was there listening to them, asking questions. In one situation, the Customer was distressed that they had let an important aspect slip through on a recent iteration. As part of the follow-up to this statement, we were able to remind them that they had worked over 60 hours. The interviewee was then able to decide that they could not have worked more hours and none of the work they did during that iteration could have been dropped, so they were able to “forgive” themselves for this “mistake”. Based on this type of situation we initially named this role Customer Counsellor, however, the term Coach is a more accepted name in the agile community for this type role.

The Customer Coach may also play the role of a Geek Interpreter (refer to section 6.1.1), as they do have that interest in technical-business collaboration and typically have a technical background. However, they have a wider focus. They give the Customer someone to talk to, to help them resolve their issues, ensure they realise they are not alone and to mitigate the risk of customer burn-out. To be effective this person should:

- Have enough IT and business experience to provide effective and pragmatic support.
- Have an awareness of XP practices and how the on-site customer practice works effectively
- Not try to solve the customer’s problems but work with them to help them solve their own problems.
- Be someone the customer can trust.

In this practice we aim to provide professional support to customers. This practice combines the patterns Mentor and Shoulder to Cry On outlined by Manns and Rising [95]. We have found that the Customer Coach practice makes a difference to the well-being and effectiveness of the customer.

While Beck [2] introduces the role of Coach, we have discovered it to be beneficial to have a customer-focused Coach. Hussman [78] writes of his experiences as a Customer Coach, and his experience also seems to support the recommendation of a dedicated Customer Coach.

**Summary.** The Customer Coach is the person on the team who has a grasp of XP practices and how the onsite customer practice works effectively. They are the person who mentors and advises the Customer; they are often the Customer’s first port of call for advice or “a shoulder to cry on”. The Customer Coach will typically operate much like a counsellor in that they will help the Customer find their own path, rather than doing it for them.

**6.4 DISCUSSION**

“A real customer must sit with the team, available to answer questions, resolve disputes, and set small scale priorities. By “real customer” I mean someone who will really use the system when it is in production. If you are building a customer service system, the customer will be a customer service representative. If you are building a bond trading system, the customer will be a bond trader.”

—Kent Beck [2, p. 60-61]
“An XP project is steered by a dedicated individual who is empowered to determine requirements, set priorities, and answer questions as the programmers have them. The effect of being there is that communication improves, with less hard-copy documentation - often one of the most expensive parts of a software project.”
—Ron Jeffries [96]

The initial books about XP describe the customer role as a single person. In the cases we studied, we discovered that a team of people form to perform the on-site customer practice initially outlined by Beck [2]. While today there is a greater acknowledgement within the community that the customer is more than one person, this chapter provides a deeper insight into the roles required on that Customer team.

The ten roles that we outline emphasise three core elements of the on-site customer. The first group of roles are Collaboration Guides and these include the Geek Interpreter, Political Advisor and Technical Liaison. Their focus is advising or guiding the customer team as they build trusted relationships both within the team and in the wider organisation. These roles tend to be undertaken by people who are not formally recognised as being part of the customer team, but they are a necessary component. The second group of roles are Skill Specialists, and these include the Acceptance Tester, User Interaction Designer and Technical Writer. The focus of these roles is to assist the customer to undertake their specific customer activities such as writing stories, verifying stories and writing user documentation. These are often recognised roles on the customer team, and typically filled by specialists. The last group of roles are Direction Setting roles and these include the Negotiator, Diplomat, Super-Secretary and Customer Coach. These roles form the heart of the on-site customer practice, with the Negotiator typically being the leader of the customer team, and also the “official” XP customer. It emerged that the Direction Setting process is typically negotiated amongst a large group of stakeholders rather than simply “being known”, as Beck seems to suggest. The aspect of real customer involvement still remains essential and the Diplomat fulfils this aspect of the on-site customer practice. The Super-Secretary and Customer Coach are the last two elements at the heart of the Customer team. The Super-Secretary removes much of the administrative burden from the Negotiator and more importantly, often fills in as the stand-in for the Negotiator when the Negotiator is not available. Finally, the Customer Coach, when available on the team, provides essential guidance and advice to the entire customer team, particularly the Negotiator and Diplomat, helping them to take the steps that determine the direction of the project.

Despite there being ten roles, we never observed a customer team where the customer team was comprised of ten people, each playing one and only one of the roles outlined in this chapter. Instead we observed that one person can play multiple roles, and multiple people might combine to play one role (e.g. Diplomat, almost by necessity will need to have multiple people playing this role). The individuals on the team may or may not be formally part of a Customer Team, but their role is still essential to the effective functioning of the Customer Team.

How these roles are established is also contextually dependent. Someone in the customer role may informally create the roles to provide the support they need, or the roles may be created as part of a more formal management process. In the cases we studied, the former occurred, with the team forming based on need, as very little guidance was available. As our research has progressed, we have had the opportunity to use the findings presented in this chapter to form customer teams, both as someone in the Negotiator role, and as the Customer Coach.

There is a concern about the use of roles in the agile community, as evidenced by the recent email thread starting on the 14 August 2008, entitled “roles, skills and hats” on the agile-usability group [97]. This thread exemplifies many such discussions over the last few years on these lists, with
some in the community suggesting that role specialisation tends to make the team less effective (and perhaps less collaborative); and others suggesting that roles can also assist us by allowing us to focus on a specific goal associated with that role (e.g. usability) and also provide a certain recognition or respect for this goal. This chapter uses roles to help illustrate the key concepts that need to be covered by a fully functioning customer team. They are illustrated in such a way as to reduce the role specialisation concerns evidenced within the community. For example, most are not names people would place on a business card, but do hopefully conjure up a memorable activity-based or analogy-based description of the need. We specifically draw attention to this issue for skill specialist roles, as these are the roles most likely to cause concern.

6.5 SUMMARY

In this chapter we have explored the notion that the XP customer role is not one person but a group or team of people within a larger whole. We have attempted to describe the essential elements within the team as roles:

- **Collaboration Guides**: Geek Interpreter, Political Advisor and Technical Liaison. The core function of these roles is to assist with collaboration and relationship building, both internally within the team and externally between the project team and the larger organisation. All of these roles are informal roles and sometimes (particularly in the case of the first two) will only be directly visible to the Core Customer Team (Diplomat/Negotiator).

- **Skill Specialists**: Acceptance Tester, User Interaction Designer and Technical Writer. The core function of these roles is to assist the customer to undertake activities such as writing stories, accepting stories, and writing end-user documentation. Often these roles are formal or recognised roles on the team, but not always. Sometimes it is sufficient to have someone on the team with this skill-set.

- **Direction Setting**: Diplomat, Negotiator, Super-Secretary and Customer Coach. The core function of these roles is to set the vision of the software and establish what the team will build to meet the business need. These roles combine to make up the heart of the customer team.

We have described ten different roles that comprise the customer team. We observed that there are multiple people forming a customer team, and that most people pick up one or more roles.
7. THE CUSTOMER PRACTICES

In the last chapter we outlined the roles that emerged to form the customer team; now we will turn our attention to what the customer team did on the project. Practices are a form familiar to practitioners in the agile community, for describing activities on projects. So, in this chapter, we will outline the eight customer practices that emerged from the case study data. We present the practices in the order that they emerged during our data analysis:

- Customer’s Apprentice
- Programmer Onsite
- Programmer Holiday
- Roadshow
- Pair Customering
- Customer Boot Camp
- Big Picture Up-Front
- Re-calibration

These practices contribute to three of the second edition practices outlined by Beck [32], as depicted in Figure 7-1 below. While in the background chapter we focused on the practices in the first edition we must now recognise that in the second edition Beck introduced three overarching practices: Real Customer Involvement, Whole Team, and Energized Work. Real Customer Involvement emphasises the direct involvement of end-users and other business stakeholders on the project. Whole Team refers to the practice of including all of the skills and perspectives on the team necessary for it to succeed. Importantly, Whole Team emphasises the importance of the sense of team, all team members sharing a sense of purpose and supporting each other. Energized Work emphasises working “only as many hours as you can be productive and only as many as you can sustain”. We can position each of the new practices we identify as contributing to the three more abstract higher-level practices suggested by Beck.

Figure 7-1: Customer Practices

The list above provides the core practices that have emerged from the case studies as contributing to the effectiveness of the on-site customer. In this chapter we will discuss each of the above practices individually. In our studies we also observed wide-spread use of the already well-known customer practices including Planning Game, Short Releases, Stories and Tests [2; 32]. We do not discuss these practices in this thesis beyond what was discussed in chapter five. Typically these practices were working well and in the way described in the literature. We also do not address the many practices that every customer will employ that apply to any software product development project. For example, market research, business process modelling, budgeting and so forth.
7.1 **CUSTOMER’S APPRENTICE**

Customer’s Apprentice: Programmer works on the customer team for an iteration so that programmers can understand the complexity of the customer teams’ role.

Merholz et al. [98, p.36] define empathy, in the context of user research, as

“being aware of another, or sensitive to, and vicariously experiencing the feelings, thoughts, or experiences of another without having those feelings, thoughts or experiences explained to you.”

We observed that people on effective whole teams exhibited empathy and respect for other team members and the role that each person plays on the team. It emerged that one way for developers to understand the Customer was to “walk a mile in their shoes”.

We first encountered the dramatic nature of this experience in a story from SwiftCorp. In the story, which multiple members from the same team all related to us, the programmers were becoming increasingly frustrated with the customer as the customer was providing them with insufficiently detailed stories (and sometimes simply not enough stories for a complete iteration). In exasperation one of the programmers suggested he come and work with the customer for an iteration or two to help them out. This event became the turning point for the team:

“[Programmer] said well why don’t I spend a day and go and write stories with our customer and that’ll help him out, that’ll help him out in terms of (a) they’ll just be another ... warm body writing stories (b) [they can answer] any technical questions he has and (c) it gives me an insight into how difficult or not difficult it is. And if it turns out to be really easy then we’ll continue to hit our customer and we’ll track spikes through his chest. But that’s not what happened. He came back and said he kind felt that oh wow, there’s really kind of a process to writing stories and to mine requirements and go hunt down the people in the business that want something and to get them to explain what it is that they want and so on and so forth. And it made them ... it humbled them a little bit I think and that made the relationship much more productive ... And it completely turned things around for that group. In that sense the developer had a little bit of insight into the pain in the customer’s world.”

— Programmers Coach, SwiftCorp

“[Programmer] came out of the development group for a while and helped us write a few stories and I think that helped him understand the somewhat ambiguous nature of what we were trying to do ...”

— Customer Coach, SwiftCorp

“Being a customer is really difficult. I was acting for one for a while.”

— Programmer, SwiftCorp

In doing this, the programmer did indeed help the customer move forward but, perhaps more importantly, he gained a deeper understanding and appreciation of the extent of the customer’s task, and was able to take that understanding and empathy back into the team. This event was critical in helping the team work together more effectively.
The SparrowCorp team relate a similar experience on their project. In this situation the programmer-customer bond was enhanced when the programmers attended the “Introduction to the System” sessions that the customer held with external stakeholders:

“Six different developers, six different places with [the customer], and what they saw was us being very positive about the application, and positive about the people that had built it, and talking about what would come next ... they saw us demonstrating an external face, a trust that they could deliver.”

—Customer, SparrowCorp

The programmers undertook some technical installation tasks at each visit, but more importantly they got to see the other ‘face’ of the customer. The programmers saw her internal face every day, but here they were able to see her external face: the positive way she projected the project and the team, and to see the pressure she faced from external stakeholders. Again, peeking into the world of the customer was beneficial for creating a stronger whole team with greater business-technical collaboration opportunities.

A number of experience reports [80; 99; 100] also describe situations where situations arose that led to opportunities for the Customer’s Apprentice practice to be used. Hodgetts [80] describes a Government Workflow Project where the business experts were not skilled in analysing the processes and creating a specification for the programmers, and so an “analysis backlog” developed. The team resolved this issue by two of the programmers moving over to assist (and coach) the business experts for a couple of iterations. The benefits were two-fold, the business experts developed new skills, and were able to better meet the needs of the programmers, and the programmers were able to understand and help address the issues that had begun to reduce programmer morale on the team. It can be inferred from the write-up that relationships were also strengthened. Williams and Stout [100] write of a similar experience at Sabre Airline Solutions. At Sabre Airline Solutions, again there were issues with the customer not being able to meet the needs of the programmers. In this situation, however, it appears the issue was that the Customer was not co-located and there were only three customers for a programmer team of 30. The Sabre Airline Solutions team introduced senior developers to assist with the customer’s workload; they operated as “feature owners”, and they helped the customer to write stories and acceptance criteria. This strategy was implemented as a permanent strategy and there was recognition that it would be preferable for this strategy to be a short-term solution until additional people could be added to the customer team. Takats and Brewer [99] write of their experience at Sapient, while working with the U. S. Office of Naval Research (ONR). Their experience report differs from Hodgetts and Williams et al’s, as they did not react to a problem or issue with the customer. A few of the senior programmers expected to be on the development phase were selected to assist as note-takers (or in some cases facilitators) during initial JAD (Joint Application Development) sessions that occurred on the project. This team proactively included programmers as what we call “Customer Apprentices”, so that the programmers would gain a direct insight into the domain, and create relationships with the customer team prior to the development phase of the project starting.

These examples help to illustrate the various situations where programmers see the customer as the bottleneck and have no understanding why the customer does or does not do “x”. The Customer’s Apprentice practice allows the programmers to gain an insight into the customer roles, exposing the programmers to the complexity and ambiguous nature of setting the direction for the project team.

Sharp and Robinson [101] observed four mature XP teams over the course of a week or so. One observation made during their study may be particularly pertinent to the Customer’s Apprentice practice. Sharp and Robinson observed that the analysts (playing the customer role) were less
likely to involve programmers in solving the problems they themselves faced, but would happily assist the programmers to solve the problems the programmers faced. So it may be that programmers will be more accepting of the Customer’s Apprentice practice than the customers.

It also emerged that programmers who have played the role of the Customer’s Apprentice are more likely to see the team as a whole team. They will step in to help the customer out when the customer becomes overloaded, and will also “defend” the customer within the programmer team, helping other programmers become aware of the true demands on the customer. This practice therefore helps to move us closer to the vision Kent Beck had of the whole team practice [32], creating a stronger and more effective business-technical collaboration. It appears that this practice works best when the programmers suggest it themselves, but we also found it was possible for a Customer to headhunt a good programmer candidate for this role.

This practice has three important implications for the whole team. Firstly, the programmers develop an insight into the difficulties and complexity the Customer faces. This understanding or empathy assists in creating more of a whole team. Secondly, the programmers also (often but not always) have the opportunity to understand the domain directly from end-users and other stakeholders and build relationships with these people as well as the Customer, resulting in improved Real Customer Involvement from the programmer’s perspective. Thirdly, in each example the programmer assisted in reducing the burden of overload from the Customer, improving their ability to experience Energized Work.

**Summary.** The Customer’s Apprentice practice involves a programmer (or programmers) moving temporarily onto the Customer team to assist them to perform their role. The programmer will typically assist the Customer by helping to write stories or acceptance criteria. The programmer may also attend meetings with the larger stakeholder community. Our analysis suggests that this practice contributes to the Whole-Team, Real Customer Involvement and Energized Work.

### 7.2 PROGRAMMER ON-SITE

Programmer On-Site: Schedule site visits for programmers so that programmers can understand more about the end-users they are building the software for.

Beck aspires to obtain Real Customer Involvement [32] on XP projects. His initial practice aimed at that goal was called On-site Customer [2]. He later replaced that practice with the Real Customer Involvement and the Sit Together practices [32], his idea was

“To reduce wasted effort by putting the people with the needs in direct contact with the people who can fill those needs.”

Given the emergent findings from the case study data and the other papers discussed in this section, we suggest an alternative practice of Programmer On-site. This practice resembles Constantine’s [102] advice for *office visits*.

Firstly, let us consider the Sit Together practice. Coffin [103] writes of a Traffic Violations System where the team moved into a meeting room in the same location as the end-users of the system. Coffin writes that this move resulted in the end-users feeling actively included in the requirements and design. Additionally, it also allowed the unit’s supervisor to attend the daily stand-up meeting, and if an issue arose, a programmer could just drop by a user and explore the issue in more detail.
In our studies it emerged that if the teams collocated, it was typically the Customer who moved to the location of the programmers. At OwlCorp the End-User Representatives relocated to another building some distance from the location of their business unit. It was easy for the Customer to spend 40 hours a week with the programmer team, as there was so much to do. The collocation amounted to an almost eight month secondment, and it seems likely that the customers became more in-tune with the project team than the end-users they represented.

At KiwiCorp the programmers were located in a building that is approximately a 20-minute walk away from where the organisational representatives were located. The KiwiCorp Customer walked between the two buildings on a daily basis. The dual location resulted in the KiwiCorp Customer, an already overloaded Customer, spending over two hours a week simply walking between buildings. So, if the Customer needs to move physical location to become the “onsite” Customer, then there is a risk that they will become isolated from the business organisation, and can also lose their business or ‘true’ end-user perspective.

Now, let us turn our attention to the Real Customer Involvement practice. In our case studies we noticed that programmers were keen to better understand or connect with the direct end-users of the system they are building. Programmers who had a Negotiator who was not also an end-user were more likely to notice and comment on the lack of (customer) end-user contact, particularly when they do not see the Negotiator-Diplomat interaction. For example, in EagleCorp we see comments like this:

“I’ve always felt bad that we never talk to the customer. Like engineers … don’t actually go out and have meetings with the customer. [EagleCorp] has their Customer Advisory Council … but we just get features [and] we don’t see how people would use it … it would definitely be nicer to have more of an interface to customers.”

— Programmer, EagleCorp

Similar comments were discovered in other cases where we had a product manager or business analyst playing the Negotiator role and the end-user was not directly available to the programmers. In situations like OwlCorp, where we had two end-user Diplomats available to the team full-time, we did not see comments like these from the programmers.

It is not as simple, however, as simply putting end-users and programmers together. At RavenCorp, for example, despite a full-time end-user representative being on-site, there appeared still to be a deep lack of understanding of what the end-users were trying to accomplish:

“I guess the other thing that I would change … [is] the … attitudes … it seems like … the software developers … don’t have as much of an interest in what we’re doing on the science side as far as what … the product we’re producing is going to mean to [the domain] in general and if they could be more interested in that and if they could … grasp how useful … just how cool what we’re making is … then they’d] enjoy working … [and] … be more committed to this company … not just be someone who, [comes] in to do their 8 hours of work…”

— Scientist, RavenCorp

A number of recent papers [84; 104; 105; 106] have begun to consider Real Customer Involvement, most turning in some fashion to UCD (User-Centred Design) for inspiration. Beyer, Holtzblatt and Baker [84] provide some insight into involving real end-users in the project. Their recommendation is to use Contextual Inquiry from the UCD toolkit. Observing the end-users in their day-to-day activities, and then providing summarised models from those observations to inform the larger project team. The additional, and in the context of this research important, recommendation is to include programmers as part of the cross-functional contextual
inquiry team. Beyer et al. note that the inclusion of programmers on this team does not always occur in practice, but then recommend involving them as early as possible afterward and making the programmers aware of the contextual inquiry findings. Judy and Krumins-Beens [105] detail how an all-male team used personas to help find inspiration and create a product aimed at women. The all-male team created three personas by considering their own friends and family. Broschinsky and Baker [104] combined the use of contextual inquiry and Personas, with one of the models resulting from their use of contextual inquiry being a set of Personas. Broschinsky et al.’s experience helps to further highlight an additional insight concerning programmers and direct user connections. Earlier we quoted one of the EagleCorp programmers concerning the lack of direct user involvement. This team used personas, but the EagleCorp programmers still felt disconnected from the users. Broschinsky et al. noticed that the data only resonated with the programmers on their team once they brought actual or real end-users in to meet the programmers. It was at that point that their findings became real and believable for programmers.

Meszaros and Aston [106] turned to Paper Prototyping to help facilitate direct user involvement on the project. The programmers assisted in this process and so gained insight and connected to the stories they were implementing. Overall although the use of UCD helped programmers understand end-users it seems clear that there is no substitute for the programmers actually interacting with the end-users.

Moreover, while we have tended to focus on the impact of these techniques to the programmers in the above paragraph, it is essential to remember that the benefits are always two-fold. Beyer [84] highlights that a number of misconceptions arise with Beck’s concept of Real Customer Involvement including that “people cannot articulate their own work practice” and end-users and other stakeholders “are not designers”. The use of UCD techniques like Contextual Inquiry, Personas or Paper Prototyping, allow us (the project team) to assist the users articulate their work practices and design an effective system.

Programmer On-site is concerned with the whole team understanding the end user and context of use, with the programmers gaining enough information to make helpful suggestions. The extreme version of this practice would be to have programmers on-site 100% of the time, but we saw great value when the programmers simply visited the end-user sites. Programmer On-Site also allows the Customer to remain grounded in their organisation and connected to all of the end-users and other stakeholders.

Note that this practice emphasises the programmer and end-user connection, whereas the previous practice Customer’s Apprentice emphasises the connection between the programmer and the customer (typically Negotiator). Both practices use empathy and ‘walking a mile in the other person’s shoes’ to develop that relationship, respect and empathy. Note also that this practice is distinct from the UI Designer role in the project. That role has specialist expertise, which can include UCD techniques. This practice uses UCD techniques to increase programmers’ empathy with users, and to make them more likely to work well with the customer team (including UI Designers).

**Summary.** In XP projects we wish to create meaningful connections between end-users and programmers. This practice recommends allowing programmers to peek inside the world of the end-user. As with the original XP practices, this practice has different levels of extremeness, with the most extreme being programmers moving their work area to the end-user site. The more likely version of this practice is to utilise UCD techniques, such as contextual inquiry, personas or paper prototyping, to gain insight into the world of the end-user and their work practices. Ideally the programmers would be part of a cross-functional team undertaking this research. This practice contributes directly to obtaining Real Customer Involvement that is helpful and productive for both the team and the end-users. Additionally, this practice contributes to the Whole Team practice, as it allows the whole team, including the programmers, to connect with...
the end-user’s perspective and needs and be in a position to contribute to creating an effective experience for the end-user.

### 7.3 PROGRAMMER HOLIDAY

Programmer Holiday: Schedule an iteration of technically focused tasks so that the customer can have some time to think-ahead.

We noticed that teams increase the number of programmers assigned to a project slowly at the start of a project. The programmers who join early will often be working on technical tasks, not driven by stories, such as setting up the technical environment. This phase of the project is often termed *iteration zero* or sometimes iteration -1 [107]. During iteration zero the customers are typically working on the initial negotiations concerning the direction of the project from a functional perspective. This process gives the Customer the “thinking time” needed to establish the direction of the project, and describe some initial stories for the first few iterations. The projects we studied often lasted over six months, and were typically over twelve months in duration. Was there ever a need to replicate that situation once the programmer team has ramped-up to full capacity?

As we have seen the XP iteration driven approach is intense both for the business and technical sides of the team. We observed that programmers dealt with the intensity by taking time out of the story driven development cycle to do a technical task for an iteration:

> “… one of the ways the [developers] deal with [the intenseness of the process] is to give the opportunity for the developers to … choose during the iteration cycle to step out of the development process and to work on something that's more of a sideline, so they'll work on something that's supporting the iteration ... [called a joker card] ... I'm going to step out of the development, user story development and just work on a supportive project. It may be possible - I haven't really looked into this too much - it may be possible for a [customer] to do something similar, I'm not quite sure how that would work at this point in time …”

—Customer, EagleCorp

These technical tasks included repaying technical debt, upgrading software or hardware, developing a tool to support development (e.g. a code generation tool) or conducting research into new technologies. Mackinnon [108] describes in his experience report concerning Connextra, one of the first XP teams in the United Kingdom, a similar concept, involving what the Connextra team named “gold cards”. They discovered that gold cards helped improve the team morale and reduced the monotony of the iterations for programmers.

The SwiftCorp Coach discussed one team who took this joker or gold card practice to an extreme and had a whole iteration for refactoring:

> “[the company] use to have re-factoring iterations, so they would let the team re-factor for a whole iteration ... their rule was that at the end of the iteration all the stories ... and all the unit tests ... run and ... they could do whatever they wanted under the covers. It's a little bit like giving all the developers, I mean pardon me for saying this about the developers 'cos I was one, so I can say it's like here's

---

*Sometimes the task may take only part of an iteration.*
some lollipops and popcorn, you know, have a good time and it's just like, you know, at the end of the two weeks, they were like gorged and they were ready to have salad for a while again, you know, that's maybe not a good analogy.”

—Coach, SwiftCorp

So a mechanism to provide a “time-out” for programmers emerged, but how could a time-out of the iteration process be provided for a Customer? The Customer drives the iteration process, so a time-out for the Customer automatically appears to have the result of programmers not having enough stories for an iteration.

At OwlCorp, the team did at one point briefly redeployed some of the programmers, so as to provide some time for the Customer to “think-ahead”:

“Yes, I mean there was a stage while this was being developed ... that I was effectively called off to do another project.”

—Programmer, OwlCorp

But what if you cannot redeploy the programmers onto another project? We faced this situation on an XP project where we played the Negotiator role. The project was two and a half months in, one release had been made, and the second was well on the way to completion, but the customer team was not sure about the end goal of release three. During the first two releases the Diplomats had discovered much about the team and the process. They were comfortable moving forward with both the team and the process; trust had been built. The third release was critical, but they were not sure what stories would be needed to meet its goal, they were not sure how to “break the back” of this problem. The Customer team needed time to think, yet the Customer team’s typical day was almost entirely spent answering questions and clarifying stories from the programmers for the second release. The Negotiator had noticed a growing unease within the programmers because they were carrying a lot of technical debt and had not had a break from the story-driven iteration cycle to fix the technical debt and research a new build and testing tool. Inspiration arose from the findings that had emerged on the research to date, and an agreement was struck. The programmers would spend an entire iteration on technical tasks of their choosing, and the customer team would step away from the iteration process and “break the back” of the third release. This is the practice that we now call Programmer Holiday. An alternative implementation of Programmer Holiday is to redeploy some or all of the programmers onto another project.

The Programmer Holiday practice involves a break from story-driven development iterations. The customer perspective is the ability to focus on what to build next. The programmer perspective is a break from the intensity of XP. They are able to decide as a team the content of either an entire (or significant portion) of an iteration: what technical debt, technical investigations or non-story development tasks would be most beneficial to the project. This practice allows a repeat of iteration zero to occur at different points within the project timeline, and not just at the start of a project. The customer may also request programmers to act as the Customer’s Apprentice or organise for them to do site visits and observations of end-users (Programmer On-site) during a Programmer Holiday. Programmer Holiday allows the Customer the time they need to focus on communication with stakeholders, and determine the next steps to take on the project.

Programmer Holiday directly contributes to Energized Work for the whole team, but most importantly, from the perspective of this research, for the customer. The customer gets a break from supporting the current iteration and is able to focus almost solely on setting the direction for the next stage of the project. Without the Programmer Holiday practice, the alternative that we saw in many of the case studies was the customer attempting to do both, and often ending up working very long-hours. Programmer Holiday also contributes to Whole Team. As the
SwiftCorp Coach stated in the earlier quote, giving the programmers a two-week iteration of refactoring is like giving them candy. It also can provide the perception that the customer is not unreasonable and only story-driven, and that the customer can also understand the importance of the technical tasks.

**Summary.** Programmer Holiday provides the customers with time to “think” about the direction of the project, and to work with the larger stakeholder community to reassess that direction based on the findings of earlier iterations or releases. This break or time to think is achieved by sending the “programmers on holiday”, most commonly by allowing the programmers to have an iteration to focus on any items of a technical nature such as technical debt, or research into new tools or technologies. A Programmer Holiday can be combined with the Customer’s Apprentice or Programmer On-site practices as well. Programmer Holiday is a practice that contributes to Energized Work and Whole-Team.

### 7.4 Roadshow

Roadshow: Demonstrate the software to end-users and other interested stakeholders so that we can obtain feedback on the direction of the project.

Beck [2; 32] would like Real Customer Involvement on XP projects, and more specifically he would like opportunities for end-users and other stakeholders to provide feedback on the software as it evolves. Software systems often have a large number of end-users and stakeholders. It emerged that the customer team typically included Diplomats, who work with the team to represent the perspective of different sections of this community. But how did the customer obtain the feedback on the software from the larger communities they represent?

The SwiftCorp customer provides some initial insight with these words:

“At the end of each release we’ve sat down with the business and showed them how the functionality worked, talked about changes that are coming up ... so I think that it really brings everyone together on a frequent basis and focuses their attention on some of the features.”

—Customer, SwiftCorp

So we can see that the customer creates an opportunity to demonstrate the functionality developed, as well as opportunities to discuss upcoming functionality with the larger stakeholder community regularly. We have called this practice a Roadshow.

Most of the other case studies also used this practice. EagleCorp, a software product development company, described their Roadshows, and the different audiences of their Roadshows, in some detail. We noticed that the intent of the Roadshows varied slightly based on the interests and needs of that audience, as well as what the customer team needed from that audience.

EagleCorp used a Roadshow to reach an internal audience. The internal audience consisted of the sales, marketing, operational support departments and the executive management team. The team used a different form of Roadshow for the weekly and monthly interactions with their internal audience:

“... so I publish a report every week to all of the [internal] teams, this is what we’ve been working on, this is what we’re building, you know, here’s the user
stories, here’s the application if you want to go play with it and give me your feedback, ... and then every month I’d do a technical field address ... a webinar [to] ... get feedback ... I’d run through all the application functional areas that we’d created and I also did that from the personas perspective so I got the organisation more used to using the personas and understanding the personas. So there wasn’t a huge cost on my time and, you know, the challenges that all technical field organisations getting [anyone’s] time is a challenge, but you know we get reasonable turn out every month …”

—Customer, EagleCorp

The team used the Roadshows not only to report progress, gain feedback on the functionality in-development, but also to introduce a new concept to the organisation. EagleCorp was very passionate about the usability of this version of the product, and so had introduced both personas and user stories. The Roadshow provided an opportunity to educate a larger audience within the organisation than the development team about these techniques and make them accessible to the larger group. The EagleCorp team were happy at how, over the course of the 12-month XP project, both personas and user stories had become an integral part of the language used within the organisation. Roadshows and a successful XP project appeared to be, at least in part, responsible for that organisational change. An additional side-effect of these internal Roadshows was that the executive management was assured that the team was making demonstrable progress towards a shippable product.

EagleCorp also used a Roadshow to reach their many external audiences. The product manager described their interaction with the Customer Advisory Group (CAG). The CAG was a selection of representatives from their existing product base, it consisted of a mix of end-users, champions, and other stakeholders from the organisations that had purchased their software product. This group met on average once a quarter, with two of those meetings typically being multi-day face-to-face meetings, and the other two meetings typically being a much shorter 1-2 hour webinar. The customer describes the importance of two-way communication during these sessions. During the longer multi-day face to face meeting the Roadshow typically includes a presentation from one (or more) of the companies using the product currently:

“It’s very much a two way communication so typically ... we’ll ask ... the customers to ... present to us how they’re using the application at their organisation. So they go through and tell us, or they show us what business problems it solves for them, how they use it, how their groups use it, and then they’ll tell us some of the challenges that they have – whether it’s business level challenges with adoption inside their organisation, or whether it’s, you know, problems with, problems ranging from they don’t like the way a certain feature works all the way through to, you know, how can we justify the cost of bringing the application into the ... bringing more users to the application ... it’s very valuable because it helps me understand what some key customers are doing, some of the key issues that they have. It also gives me an opportunity to say, hey, which parts of the application don’t you use, which parts of the application need major changes before you could find them valuable and, you know, it certainly helps shape things for me specifically. It helps me ... [as] ... they are not entirely representing all of our customers so I have to make sure that I apply a logic when I’m working with them but it helps me get a very good understanding of whether certain applications that we’ve given to them or they’re using, where some of the real major holes are and how I should, you know, spend my engineering function dollars actually building the next generation tools.”

—Customer, EagleCorp

Angela M. Martin

Page 108 of 150
Chapter 7: The Customer Practices

The key focus of the Roadshow is demonstrating the product and getting feedback on what has been developed since the last meeting, as well as what is on the radar for the next development period. One of the advantages that XP gave them as a software product development company was the ability to demonstrate working software rather than discussing ideas or using ‘smoke and mirror’ prototypes:

“I think it makes a lot of difference that, again, until it’s something that they can really look and feel, that they can actually touch ... this industry is still, you know, selling on a promise a lot of the time. You know, [clients] over a course of time have just become very disillusioned with selling them promises. So, just being able to show them the application and show them the fact that it is functionally working, it makes a lot of difference ...”

Customer, EagleCorp

The case studies also revealed the usefulness of a much less structured ad hoc version of the Roadshow. At EagleCorp and RavenCorp a sales representative would demonstrate a version of the working software to a potential client. The sales representative would then collect feedback from the potential client and feed that back into the customer. At times, the sales representative would be able to demonstrate the incorporation of that feedback on a subsequent visit. Other case studies used this type of unstructured and ad hoc Roadshow for internal audiences as well. For example, at OwlCorp the customer would often demonstrate the software system (as it currently stood) to a business unit, as part of the business unit’s regular meeting. These Roadshows were typically initiated as considered appropriate by the business unity manager or the OwlCorp customer.

Scrum has a process that initially appears to be the same as the Roadshow practice we describe; the Scrum practice is called a Sprint Review [38]. Both practices involve a regular meeting with interested project stakeholders to review the functionality developed. One of the biggest differences between these two practices is that Roadshows are tied into the pulse of the organisation rather than solely to the pulse of the development team. For example, at EagleCorp, the Roadshows involved external clients at the quarterly Customer Advisory Council meetings, and whenever the Engineering Manager or Sales Representative was calling on an important client who might be interested in the product. Another difference we noted was that Roadshows were typically organised and facilitated by the customer team, sometimes with other members of the team present, whereas the whole team typically undertakes Sprint Reviews. These two differences may be inter-related, as tying the Roadshows into the pulse of the organisation may make it harder for the whole team to attend.

Hansson, Dittrich and Randall [109] provide an ethnographic description of how end-user participation was obtained by a small organisation using an agile-like method. One of the methods used were “user meetings”. At these regular meetings users were encouraged to meet other users, and share examples of how they had solved problems, or learnt new functions within the software. Users are also invited to submit proposals to change the software, which are all considered and ranked. Two interesting differences appear to exist. The first difference is that no demonstration of the software appears to take place, this maybe because the system is a mature system, unlike the new software development projects we studied, or it may simply have been taken for granted that the system would be demonstrated at these meetings. The second difference is that all programmers involved had direct contact with users at some point during this process. Long and Starr [110] provide an experience report from Healthwise concerning their implementation of Scrum Sprint Reviews. Over time they adapted Sprint Reviews to become more like a science fair model. The different teams build displays, and the (in this case) internal stakeholders walk through the displays and stop for a short-presentation if interested. This technique may provide a way to scale-up the Roadshow practice.
Summary. A Roadshow provides a way for the customer team to obtain frequent feedback on the direction of the software and streamline their communication with a large and diverse set of interested end-users and stakeholders. The Roadshow typically involves a demonstration of the working software, and so the customer team are able to leverage this opportunity to also provide confidence to this community that this project will deliver value, that it is not simply vapourware. One important element concerning Roadshows is that they are always tied into the pulse of the organisation and not the project team. Roadshow is a practice that contributes to Real Customer Involvement and Energized Work.

7.5 Customer Pairing

Customer Pairing: Two members of the customer team working collaboratively to provide a single-voice to the development team.

We noticed the practice of customer pairing in varying degrees in many of the cases; it particularly stood out in OwlCorp, where two end-users worked together as a customer pair on the team:

“The work was quite evenly distributed. We would share a lot of ideas in conversations that we had, and we would discuss practically everything. We wouldn’t make decisions on our own very often. We would always ask the other person and discuss it to make sure everything was covered. I think that helped as well, to make sure what we were asking for was right ... to not to have had a second opinion would have been very difficult. To make every single decision about a new system on your own would have been hard. And also from the support side as well. If he was angry or annoyed, I’d support him, and vice versa ... I think it would be just too hard to be the only business-person surrounded by 10 or 15 technical people. It is nothing against them, because I really like them all, but you know sometimes, you just need someone [customer pair] to see it from your point of view.”
— Customer, OwlCorp

“[Customer pair from above quote] was on holiday, it was a nightmare. I was just running around madly trying to get everything done and it's also a lot scarier I suppose because me and [the other customer], we're kind of happy to make decisions but we'd much rather make decisions knowing the other person is agreeing with it, ... it's a lot more freaky to do it yourself with the weight of [the business unit] and their perfection on top of you that kind of thing, it's a bit harsh.”
— Customer, OwlCorp

Over the course of our interviews at OwlCorp we discovered that some of the most important aspects of customer pairing are that:

- The pair can support and complement each other when making tough decisions and prioritisation calls. Making decisions was perceived as less “scary” when operating as a pair than if they were operating as an individual.
The pair can bounce concepts, ideas and decisions off each other. The perception of the pair is that this results in an improvement to the quality of requirements and the associated decisions.

Working together reduces the nightmare of running around madly trying to meet everyone’s needs, and thus reduces the overload. This pair noticed the stark contrast concerning the overload introduced into the other person’s workload when one of them was unavailable to the team (for example, when one of them was on holiday).

The pair was in the same meetings and hearing the same things, and as such they could each discuss their interpretations of what occurred with each other directly. Their perception was that this was positive, as otherwise they would need to relay the meeting to an external party prior to discussing their interpretation, which would lose some of the context and setting of the meeting, and introduced an additional bias into the discussion.

This pair, like most of the other same customer team role pairs we observed, divided the work between them along functional area boundaries. So while they worked very closely together, one of them was more responsible for Functional Area A and the other was more responsible for Functional Area B. Both of them had experience and knowledge of both of the functional areas, and they worked together almost constantly.

At OwlCorp the pair spent almost 100% of their time as a collaborative pair. In most of the other cases, the pair utilised a divide-and-conquer strategy that allowed them to both work independently as well as collaborate as a pair. An illustrative example of this strategy comes from SparrowCorp. At SparrowCorp one of the business analysts was responsible for the requirements and needs of six of the regions affected by the system and the other business analyst was responsible for a similar number. This division of labour resulted in the business analysts being able to work independently, and develop the strength of relationships required with each region. However, they brought that information back into the pair in order to forge a single-voice for the development team:

“We [the two business analysts] sat in a room together and talked about them [the requirements] a lot … [we] spent a lot of time in small rooms like we are in now, and talked about what we needed to do, reflected, … The countries that we were responsible for work in quite different ways, have different stresses, and so a lot of our collaboration was around working out precisely what the business needed, and actually writing the story was the sausage factory at the end of the discussion …”

— Customer, SparrowCorp

The most common work divisions, amongst same customer team role pairs, that we observed were functional or geographic boundary divisions as evidenced in the examples from OwlCorp and SparrowCorp. These divisions made it easier to manage the level of domain knowledge required as well as reduce the number of people working with that customer. The data suggests other divide-and-conquer strategies are also possible. For example, at RavenCorp while there was a functional divide, the customer-pair also appeared to be aware of their individual skill-sets and backgrounds when dividing the work. One of the customers had a stronger scientific background and the other had a stronger development background.

The examples described to date demonstrate customers pairing with someone on the customer team performing the same role (e.g. two Negotiators or two Diplomats). This next example shows that pairing was also used across roles within the customer team. At EagleCorp, the user-interaction designer and the product manager worked together very closely, mostly as a pair. In this situation they did not divide by function or geography but more by type of work, with each of them picking up the types of work more suited to their specialisation, but then working closely
together to create the single-voice for the development team. We observed the cross-role pairing occurring within the Direction Setting roles, or between a Direction Setting Role and a Skill Specialist role.

Finally, it should be noted that in some cases, for example, KiwiCorp, there was very little data to support Pair Customering as a practice. The KiwiCorp customer did, however, work very closely with her executive manager (Big Boss), to make decisions and priority calls when it got “too hard”.

It is interesting to consider if some of the reasons attributed to the effectiveness of pair programming might also be at work here with Pair Customering. Cockburn and Williams [111] outline eight benefits of pair programming, three of these benefits appear also to apply with Pair Customering: Satisfaction, Problem Solving, and Learning. Cockburn and Williams report that one pair of student programmers felt that the pairing assignment was more work than the individual assignment, and asked for the amount of work to be reduced on the pairing assignment. The lecturer did not adjust the amount of work on the pairing assignment, but the students continued to pair, despite the perception that more work was required. Notice that in the cases where Pair Customering was used, it was always used voluntarily, and the customers initiated it as an informal practice. The quotes used earlier in this section also suggest the customers found it satisfying. Additionally, at OwlCorp when one of the pair was leaving the project, the business manager and the remaining person from the pair were actively recruiting a replacement in order to continue the practice of Pair Customering. As with pair programming, the interviewees using Pair Customering identify an increased ability to solve problems, make better decisions and to learn from each other. Two of the remaining benefits identified by Cockburn and Williams concern quality; it might be that future studies will provide additional data concerning the quality improvements from Pair Customering. The final two benefits that Cockburn and Williams outline concern team communication and team management. The customer teams we have seen in operation already work closely together, so we were unable to observe any cause and effect relationship. So, an initial analysis suggests that future work may find a similar pattern concerning the costs and benefits of pair programming and Pair Customering, but there is still much work to be done.

In our studies it also emerged that programmers can sometimes find it difficult to work with a customer pair. One of the issues encountered by programmers is at times the different pair members will provide different directions. Additionally, in some situations, like at SparrowCorp and SwiftCorp, the programmers have told us how they have used one of the BA’s instead of the other BA in order to obtain the decision they preferred or thought was the “right decision”:

“[regarding BA pair] ... it’s worked out fairly well because they’re both pretty knowledgeable about the systems. There’s no question that has been a little bit of, not confusion but tension I think between the developers and the customers in that respect because there’s a little bit of the idea of they know that one of the customers is a little bit harder going through the acceptance test than the other customer. So sometimes it’s like little kids of parents where they say go ask mum if we can go outside and play, mum said no, go ask dad. And that can be an issue if the customer is so busy to the point that they aren’t communicating amongst themselves long enough. And that can be even more of a problem. For the most part though, we’ve had pretty good success with multiple customers speaking with one voice to the developers.”

—Programmer Coach, SwiftCorp
The BA pairs were aware of this activity and for the most part found that they synchronised together often enough that the project was not in-danger of being misdirected by this type of activity.

Pair Customering allows the customer team to use a divide-and-conquer strategy that supports Energized Work. Notice the stark difference for the OwlCorp customer when their partner is on annual leave; for them it was a “nightmare” and they had to “run around madly”. The additional sounding board effect (or two heads are better than one) that the OwlCorp customer refers to also helps to create less stress for the customer that again helps to facilitate Energized Work. A less strong claim is that Pair Customering may also contribute to Real Customer Involvement. The divide and conquer strategies utilised by the customer team, typically also allow each of the pairs to establish stronger working relationships with stakeholders particularly when commuting to a different location for geographically dispersed teams. These relationships assist in bringing this real customer’s perspective into the team in perhaps a stronger way than might occur without Pair Customering.

Summary. Informal customer pairs have emerged in the case studies. Pairs result from members of the customer team establishing close working relationships and ensuring regular communication touch-points with each other. If some aspect of their partner’s work is unfamiliar to them we noticed that they often spend time to gain an understanding or at least passing familiarity with that space, so as to increase the effectiveness of their partnership. In order to provide a single-voice to the development team the customer pair must collaborate regularly, and it appears that at a minimum most pairs write the stories together, but may collect ideas or discover needs separately as a divide-and-conquer strategy that works in an XP environment. One pair took that practice further, and worked together almost 100% of their time. This pair was particularly aware of the dramatic contribution Pair Customering makes to Energized Work.

7.6 Customer Boot Camp

Customer Boot Camp: A customer-focused training event for an XP project.

How do the customer team, be they from a traditional business analyst or from a business background (e.g. an end-user representative), learn how to interact with an XP development team effectively? The business analyst team leader at SwiftCorp noted it as a potential issue:

“... [XP] is new to everybody ... and because it is not a methodology that the team, the BA team has adopted, they’ve not been trained on it, so it’s an adjustment, so that [is] like one of the biggest challenges. Not knocking the process, again I think the process is excellent, it’s just that the company hasn’t adopted it in density ... and until they do, you are going to have teams that have never done it, and so they are a little disorientated and apprehensive.”

— Customer, SwiftCorp

The OwlCorp coach, aware of this type of issue, suggested the team have a special customer boot camp that trained the customer in the agile process and their role. The training would involve a number of representatives from the team but would be focused on the customer’s perspective. This activity was remembered by the customer:

“I really enjoyed that. Before that, people said we were going to boot camp, and I thought, what is that? Even when they described it, you still don't really understand exactly what it is. But it was very good, though. I enjoyed doing that.
Chapter 7: The Customer Practices

... It was a nice way to learn, the way of doing things. Can you write this story, and what do you think of this. It was just a laugh. Some of it did go on a bit. It was hard taking in all the information, being so new to the project. But I did enjoy it.”
—Customer, OwlCorp

The boot camp involved a mixture of talking and doing or practicing techniques just discussed. The practice sessions were a mixture of both fun game simulations and practical examples from the domain. The key aims were to help people buy in to the process, and to gain a practical understanding of their role and what they need to do on the project. It emerged that the boot camp did not answer all of their questions, and neither did they retain everything, as they often needed to try to do some things in real-life before all of the concepts embedded:

“I think the actual process, the whole Agile process wasn't that clear after I came out of the boot camp. It wasn't until I started to do it that I started to realise what everything was. Even that iteration, just basic things like that. I didn't feel comfortable at the finish of boot camp that I understood all that perfectly. I understood bits and pieces.”
—Customer, OwlCorp

However, the customer boot camp provided the customer with a “kick-start”, an initial understanding of their role, the process, and some initial ideas of techniques like story writing that they would be expected to put into practice during the project. Additionally, it provided an initial place to meet and connect with project team members. The customer could then explore these concepts in more detail during the iterations with these same team members. Finally, the customer had access to a Customer Coach for the initial iteration or two, for further explanations.

A number of recent experience reports [112; 113; 114; 115; 116] have also reported the importance of including customer-focused training sessions as part of an agile adoption effort. Ganis et al. [112] write of their use of the Extreme Construction game when introducing XP into their environment at IBM. This non-software simulation involves specifying and building a physical model of a product using arts and crafts materials. The team invited their customers to attend their agile training, and as such the customers were quickly exposed to the ideas, principles and practices of XP. The non-software simulation allowed both technical and customer team members to gain an appreciation for all of the XP practices. Ganis et al. noticed that although their agile training was not customer-focused, involving their customers in their training resulted in the customers gaining an understanding of the importance of rapid feedback from them during the iterations. The customer also became convinced of the impact some of the technical practices had on improving the quality of the system. At one point during the project, the customer noticed the team not performing some of those practices, and so he raised his concerns with the team which encouraged the team to pick-up those practices again.

Rasmusson [113] writes of his experiences on a number of Thoughtworks consulting engagements, where ThoughtWorks was helping organisations to adopt agile practices. He writes of two practices that he uses, one being a four-day boot camp that occurs near the start of an engagement. The four days are broken down into two days that focus on aspects relevant to the whole team, including an introduction to agile, roles and responsibilities, release planning and team practices. The customers and other client stakeholders are invited and encouraged to attend these days on the boot camp. The last two days of the boot camp focus on technical practices for the development team. So, with this boot camp structure we see the whole team involved in the first two days, with the last two days focusing on developer specific techniques and practices. One of the advantages Rasmusson discusses, that comes with structuring the boot camp in this manner, is that everyone starts on the same page, and begins to form together as a whole team.
Tengshe and Noble [114] describe their experiences of introducing Scrum into Capital One Auto Finance. They use a mixture of coaching and training, including the use of the certification training courses offered through the Scrum Alliance. The Scrum Alliance has recently added a new course to their offerings, a certified product owner course. The product owner certification course [117] concentrates on techniques for writing stories, prioritising the product backlog and predicting the end date of the project. Capital One Auto Finance plan to incorporate the new product owner certified training into their training offerings. Moon [115] discusses her experiences of introducing Scrum into KeyCorp, a financial institution with $100 billion in assets. One aspect of her story is particularly relevant: the introduction of the certified product owner training from the Scrum Alliance. Moon describes the feedback from the first (and only) stand-alone product owner certification training as troubling. She found that the separation of training courses into role-specific training (Scrum Master, Product Owner and so forth) resulted in a fractured and disjointed view of Scrum. She resolved these issues by creating a combined course for all of the Scrum roles, which included aspects for each role, so that the whole team had an agreed understanding of the process as a whole.

Vriens and Barto [116] suggest an alternative viewpoint to the other papers. Vriens and Barto suggest that it is preferable to not involve customers in the initial training, but instead to bring them in after a few successful iterations. Their perspective is that involving the customer too early will result in the customers being concerned and “rejecting” the technical practices such as pair programming. Their recommendation is to hold off briefing customers until such time that the development team is comfortable with the technical practices, and the associated benefits can be better quantified. This conflicts with the advice from the previous authors, who used the initial training to help form a whole team, and build trust and understanding. However, it perhaps is important to consider the customer’s reaction to the training and the long-term objectives of the agile transition prior to involving them in an early training session.

The Customer Boot Camp practice supports the customer to become an effective member of the whole team, as it helps them understand more about their role and responsibilities. Therefore, we believe that this practice helps us to obtain Real Customer Involvement as well as help us move towards a true Whole Team that includes the customer.

**Summary.** Customers, whether their background is traditional analysis or business, benefit from some initial training and guidance on what their role and responsibilities are on an XP project. The training sessions tend to be multi-day and typically also involve the whole team. The most effective sessions tend to utilise a mixture of lectures as well as practical sessions that allow the attendees to practice these new ideas, practices or techniques.

### 7.7 Big Picture Up-Front

**Big Picture Up-Front:** A short period of envisioning amongst the business stakeholders to set the direction of the project and create an initial release plan.

In our studies it emerged that typically the customer engaged in some activities prior to the first iteration with the development team. The intent of these activities was to help answer the question “what to build?”, and to set the direction of the project:
“So, you know, it’s the classic … it’s pretty much always the classic problem – you have, you know, huge amounts of requirements, finite time, you know, what do you do build?”

— Customer, EagleCorp

“A year prior to December [project start date], we realised that we needed to [do goal of project]. So … we did a lot of research on how to [do that] … and how to sell our services along with our hardware system, things like that. In Summer and Fall we set an agreement out of how it should be, so from August to October we had a round where we talked about developing internally … or to go out and find the delivery partner … In December we met with the [selected project team] and we had I think 14 days, something like that …. We were quite amazed that in 14 days, all of that final review of a very complicated system and how they got along, which role they play and everything. That was just because of the use cases. It was good 14 days.”

— End-user Representative, TernCorp

Note that these activities concern the domain and the project whereas the Customer Boot Camp practice addresses the process. At TernCorp the initial project concept or goal was first seriously considered by the business organisation over a year before the full project team started work. The business representatives were actively researching concepts and vendors over the course of a year before commencing the software portion of the project. The software project began with a 14-day period where the end-user representatives worked with some of the members of the project team to create an initial big picture for the project. For this project the big picture consisted of a set of use cases and a release plan. For the duration of the project the release plan was on the wall of the project room and was a series of post-it notes on brown paper.

In our SparrowCorp case study it also emerged that significant investment occurred prior to the software portion of the project:

“So it’s a pretty big system. $12 million budget, 10 man years development. [And] that’s just developers, so if you add on analysts it’s probably about 15 … I came on to rescue the project because it had been going for six months, and they were thinking of going into … development, but they didn’t actually have any idea what they were meant to be developing, and they didn’t have any agreement with any of the countries that they would implement what was developed. So I said look, you’ve got a good start, let’s do this properly and get agreement from every country … because there is no point building a solution if nobody is going to use it … We did the simple standard change management approach. Find out what everybody is doing, then from that you design a standard process, and then … what each process will be, and agree with them that they are accepting the changes that they have to make … and say will you sign up to that … [we used] … workshops. Not all in one session. It took six months. We visited each country at least three times to do that. The first time to find out what they were doing, … the second time to discuss the new process and third time to do the impact assessment of the changes. … They hated it … They didn’t want to change. Nobody wants to change … [however] they conservatively estimate about $4 million per annum returns on this. When you talk to them when they don’t have to put [the figure] in their performance contract they will tell you that it is $20 million per annum …We went through a gate, a steering group gate, so basically

---

27 This project used use cases instead of user stories.
“the people, the owners of the project, business people, they said okay, we’re comfortable. We will take this to our Board and we will trust the family for the whole amount.”

—Project Manager, SparrowCorp

The output from this six-month big picture phase included an understanding of the as-is process, both by the regional business units and the analysts, an agreement for a new process, a release plan with four release milestones and an associated set of stories. Both of these examples demonstrate the sort of effort required from the customer to establish the big picture of the software project, so that they can provide a single-voice to the project team. A naïve assumption sometimes made by developers concerning this process is that the user will know what to build when asked. The SparrowCorp business analyst drew our attention to this:

“They [the developers] very quickly learned not to say ... the user asked for this. There isn’t any user, there are 10 countries worth of users, so this is a synthesis of the needs of the community, not any one particular user. I can give you an example of someone who will find this useful, but they are all going to need it.”

—Business Analyst, SparrowCorp

To understand the “10 countries worth of users” the analysts had undertaken a significant process re-engineering activity prior to the story writing and release planning. They had undertaken this activity so that they could understand what stories needed to be written, and in what order, so as to be in a position to create an initial release plan.

KiwiCorp provides an additional illustrative example of how the customer decided what to build. The KiwiCorp customer team had worked with an external facilitator to help them identify what was needed in the software, using a series of collaborative workshops over a month or two. The attendees of the workshops included a number of different stakeholders from different business units affected by the proposed software. The output of these workshops was a functional specification that was agreed upon amongst those identified stakeholders. It was at this point that a decision was made to use XP for the software project. The functional specification was used as a basis for the stories, and it provided the initial big picture for the project team. The project team did not adhere religiously to the functional specification, as their understanding evolved over the term of the project concerning what needed to be built. The functional specification had provided an excellent initial big picture of the project, showing what needed to be built and forming the basis of evolutionary change.

Most of the case studies provide a similar story: the customer engaged in the vision of what to build ahead of the engagement of the software project team. This process can be divided into two phases. The first phase covers the initial inception and research and typically results in an initial approval to proceed, often with a selected software development partner. This should be common practice for any development project, agile or otherwise. The second phase, Big Picture Up-Front, covers the period in which the customer works with some representatives from the development team to create the big picture of what will be built. The output of this phase typically consists of artefacts that describe the functionality (e.g. stories, or use cases, or process models and so forth) as well as an associated initial release plan. It is important to remember that although this activity sets out the initial idea of what will be built, all of the project case studies modified that design as their understanding evolved over the course of the iterations, showing a commitment to an agile process.

A number of published papers [99; 118; 119; 120; 121] highlight Big Picture Up-Front activities on agile projects. Fuqua and Hammer [120] describe their project to build a Common Policy Editor at Internet Security Systems. Fuqua and Hammer explain that one of the key lessons from their project was: “don’t try to find all of the stories up-front, and expect to throw many away”.

Angela M. Martin
They started their project with a four-month conception phase, and by the end of the project they had uncovered more than twice the original number of stories. Over the course of the project their understanding of the needs of the project evolved. Fuqua and Hammer don’t suggest the removal of that initial conception phase, but do suggest shortening its duration. They claim customers do not need to try and resolve all of the detail, as our understanding will change through the iterations as the working software emerges. Bang [119] describes a project to develop a student welfare intranet. On this project they also discovered it was important to spend a short-period of time, in this case six weeks, developing an initial big picture of the system. The outputs from their 6-week phase included wire-frames, information architecture, 50 user stories and an initial release plan. They did not spend longer on this initial big picture phase of the project as they knew that their understanding of the system would evolve and grow over the course of the iterations. It appears that their intent was to do “just enough” design to set the direction of the project, as doing more would be wasted effort.

Keenan and Bustard [121] outline in their paper their hypothesis that developing a Big Picture Up Front28 can enhance an XP project. Keenan and Bustard suggest that some aspects of the Soft Systems Methodology, in particular the rich pictures and conceptual models may help us to model the domain under study in a useful way for project team members and business stakeholders. These models can then help the project team discover and describe the user stories for a project. Takats and Brewer [99] describe their experiences of developing a big picture for a naval logistics command and control system that Sapient developed for the U.S. Office of Naval Research. Takats and Brewer discovered, like Keenan and Bustard hypothesised, that visual models can be beneficial when establishing the big picture of a system. Takats and Brewer also explain two other very powerful findings from their experiences. The first finding was that the team used a series of workshops to bring all of the stakeholders together to “own” the big picture. They contrasted this approach with an oft-tried traditional approach of one-on-one interviews. In the workshop approach the group as a whole had the responsibility for weaving their perspectives together into a single-voice, and as such the group owned the resulting output. Takats and Brewer claim this approach further improves the quality of big picture. The second finding was that it was important to include members of the development team in these workshops. Although Takats and Brewer note that the project took several years, and at the time of writing the team had completed two successful releases within a year, no mention of the total time spent on this initial big picture exercise is given. We do know, however, that it took at least a month or two, as it consisted of a series of 3-day workshops with stakeholders from all around the world. Again, as with the other experiences described, the workshops facilitate the creation of the big picture, but the customers’ and developers’ understanding continued to evolve throughout the course of the project. The big picture initially helped the stakeholders to make informed decisions about what to build and how to prioritise stories and create a release plan.

Armitage, Wisniewski, and de-Ste-Croix [118] outline a new approach, termed an Agile Round Table (ART) that British Telecom (BT) is utilising to make project feasibility decisions. Like other papers discussed in this section, the ART approach uses a facilitated workshop to gather executive decision makers in a room to decide the feasibility of a project. The duration of the ART workshop is typically half a day, although the preparation by the two facilitators will often be up to a week. Armitage et al. have run over 50 successful ARTs and have discovered that it not only reduces the decision making time (by months) but improves the quality of the decision made, and allows BT to position effort in the right places. One interesting aspect this paper highlights is that over the last few years agile practitioners have been finding ways to condense the up-front work required to design the big picture for a project. Traditional specification approaches may have taken many months or in some cases years; initial agile approaches often

28 This paper provided us with the name for this practice.
Chapter 7: The Customer Practices

took multiple months but are now more likely to take weeks. Collaborative workshops involving the stakeholders, end-users and development team are one of the techniques that appear to help condense this timeframe. The other is a growing awareness of what constitutes “just enough” design will evolve over the course of the project — typically stories and release plans — and agile offers many opportunities to change direction based on that new understanding.

Big Picture Up-Front directly supports Real Customer Involvement. It actively encourages the end-users and stakeholders perspectives to be included in the software vision, often through a series of workshops whose output is documented in easily understood artefacts such as stories and release plans. Big Picture Up-Front also supports Whole Team, as developers are often included in the workshops to help build their domain understanding and improve the estimating process. Finally, Big Picture Up-Front also supports Energized Work for the customer team. The one case study in which Big Picture Up-Front did not occur was at RavenCorp. The RavenCorp customer worked excessively long hours (70-80 hours per week) as he attempted to develop the big picture while supporting the current iteration (40 hours per week). While the 30-40 hour difference between those two figures is not solely attributable to developing the big picture in parallel, a significant portion of the time appeared to be attributable to that.

**Summary.** Big Picture Up-Front involves the business spending time to set the direction of the project prior to starting the development iterations. The outputs from Big Picture Up-Front vary as to the nature of the software being built, but typically visual models are common, as well as stories and an initial release plan. The time frame for this phase also varies. The time required to create and agree on a big picture appears to depend on multiple variables including the complexity of the business domain, the number of stakeholders and the length of the project. It also emerged that, over the last few years, agile practitioners have been working to discover ways to focus on “just enough” design and condensing the timeframe required from months to weeks. This is based on the understanding that what to build will evolve over the course of the project. Big Picture Up-Front ensures we set out in the right direction, but does not plot the course of the project in detail. Big Picture Up-Front supports the Real Customer Involvement, Whole Team and Energized Work second edition XP practices.

### 7.8 Re-Calibration

Re-Calibration: Plan to adjust commitments and resources regularly based on what both customers and developers learn during the iterations.

After a few iterations, many teams realise that they are not going to deliver everything that they initially hoped they could during release planning:

“We thought we’d bitten off what we could chew but we worked out that our eyes are bigger than our stomachs – it works for IT too ... so I guess one way of doing it would have been to ... do the prioritising right at the time – what can we actually leave till last? And actually leave that till last.”

— Customer, KiwiCorp

“... and we found surprise, surprise the scope was way bigger than what we thought [and] the time that we had available ... so we went through another round with the executive team of scoping down and actually changing the date one could deliver the application. ... they were pretty reasonable I think ... they
understood that up until [that] point ... we didn’t have [a] real strong idea of how long it would take to build what we were trying to build. I think they realised that, you know, hey this is the first time we’ve been able to get our arms fully around what it is and to go from what it is to at least a timeline for building it. You know, the reality is ever since then, you know, we’ve had to go through some scope adjustments, ... [and] some reprioritisations to bring the product into you know releasing it effectively next month ...”

— Customer, EagleCorp

Most teams appear to go through similar stages during this realisation process. The velocity data from the first iteration will typically indicate that the release plan is optimistic, but both customers and developers typically attribute this to the effects of a new process or technology. Their expectation, both on the business and technical sides of the team, is that they will improve and catch up. After a few iterations, however, the team begins to gradually realise that the plan was overly optimistic: more stories exist than initially planned and velocity and estimates become more predictable.

In our interviews we noticed that the reaction at this point begins to differ slightly between the business and technical sides of the team. The technical side of the team typically believed that it was good that they had uncovered this situation as early as they had. Their perspective was that it allowed the business team to make the scope reductions required in order to meet the deadline. The business side of the team, however, perceived the situation as more problematic:

“On the ground it wasn't too bad, I mean the atmosphere between developers and the business people I think has always been pretty amazing, that's partly why I enjoy working with them because, you know, they're a good bunch and from business to developer on the ground during the week it's great. I suppose a little bit higher probably the tension kind of was a little bit like, you know, I suppose [manager]... kept having to do the dirty work for us because we kept saying we can't do that with this, it's a could have and we were assuming we were going to get it because we were told we could have could haves and if we can't do that then this is really no point ... [and] being pressured into the kind of story points versus actually what we need ...”

— Customer, OwlCorp

“I think we agreed what would be a realistic range and what would be a stretch range. Not really kind of knowing what that would mean in terms of an outcome. ... I think what we then said was we want to track the progress against that realistic stretch and pessimistic goal ... I would say ... one of the things that we got done on a little bit. I am sure in discussions, we had been told ... that we could have all the musts, all the shoulds, and some of the coulds ... And if we had known that we couldn’t have the coulds, we would have put some of them in shoulds, because they probably should have been shoulds. I think that was quite a difficult lesson for us, and we got stung on that one. And then it got to a point where we found out that we couldn’t have any of the coulds, and there was some coulds that really should have been shoulds. And then we were told that was kind of scope creep and that wasn’t possible, and that was going to be a delay. That whole thing wasn’t good. .... If we were really truthful, it was a bit of a quick and dirty prioritisation and then we were kind of held to it. I guess we were a bit naïve.”

— Big Boss, OwlCorp
The primary consideration for the business team during this period is whether any reduced set of functionality will be sufficient to deliver the necessary business value:

“... so one of the things that we did at that point was I wrote several presentations that presented the different functional scopes and the different dates in which we could deliver them ... the original sets of priority 1’s would mean we’d end up delivering the product I think like in April/May next year and ... we scoped that down a little bit and added in a couple of things, meaning we could deliver a product in like January and then scooping it down a little bit again to come up with a functional set we could build ... in December. It meant at least that we kind of all had something that we knew we were shooting for – a specific date and time ...”
—Customer, EagleCorp

The programmers’ perceptions of this process were typically self-focussed. They tend to mention “endless meetings” where they estimate remaining stories, and customers not being available to answer questions during the iterations. It appears they may not understand the difficult decisions and senior executive meetings and briefings that are typically occurring. Recalling our analogy of treaty negotiation, this is often the point at which the stakeholders realise that they need to renegotiate the details of the treaty and make some hard decisions. Some stakeholders remember this process with a great deal of negative emotion. Interestingly enough, in some cases the customer’s sense of “betrayal” appeared to be greater than if the situation occurred on traditional projects. One explanation perhaps is that the customer believed XP was a silver bullet. The difficulty and negative emotions experienced by the customer at this point in the project were often not appreciated by the development team. Developers need to be aware that pat answers such as “it is better to know early” or “we just need to cut scope” might not be well received.

Beck [2] discusses the planning strategy of XP, and specifically outlines a steering phase. The intent of the steering phase is to update the plan based on what the team learns, including new stories, a better understanding of velocity, and estimates. Weyrauch [122] describes the agile adoption at Medtronic. One of the barriers the team faced in their agile adoption was the perception that agile projects do not need to plan. Much like Beck [2], the team worked to correct that mistaken impression and the result was the new perception that agile projects are all about planning and re-planning constantly, “the exact opposite of the original worry” [122].

Honious [68] describes the path of a product Reed Elsevier was developing. The team was working on a release plan that would result in the product being demonstrated at a tradeshow. Reality, however, intruded on those initial plans, and it soon became clear that the team would not be able to make the deadline with the current scope and constraints. The team worked with the senior stakeholders to re-plan. Honious emphasises that it was unacceptable to just “drop functionality” as there was a minimal feature set required for the tradeshow, so the team developed solutions with this constraint in mind. The feature set was already the minimal acceptable for the business case, so the team’s re-planned solution deferred features that were not visible to users, and they also added another pair to the development team. The team was able to obtain management’s buy-in to this solution and the project was a success.

Button [123] also describes his experience with a client and the need to plan to re-plan. Button explains the client’s previous attempts at developing software with an agile development approach. He describes the client’s experience as “unkept promises”: delivery dates were not met and in the end the project was not perceived as a success. He started the engagement knowing he needed to regain trust with that client and create a transparent planning process. He learnt that he made initial assumptions about the client’s knowledge of software development. Button assumed the client was aware that teams often go slower initially, and so Button had not understood the
The client was seriously concerned about the team’s initial progress. Over time, the team was able to plan (and most importantly re-plan) with the client and use that to form a trusted relationship.

This suggests that customer expectations need to be managed concerning this event. Experience shows that initial plans are often optimistic and customers will need to regularly re-plan.

Re-Calibration allows stakeholders to make changes to the plan regularly as they learn more about the project, thus supporting Real Customer Involvement. Re-Calibration contributes to creating a Whole Team as they move away from a “blame” culture towards a proactive and regular re-planning event. As Beck [32] states, a “mutually agreed-upon plan, adjusted when necessary to reflect changing reality, suggests a respectful, mutually valuable relationship”. Finally, Re-Calibration also contributes to Energized Work as it ensures they re-plan the work for the whole team including the customer’s workload.

**Summary.** Re-calibration is an important aspect of the XP planning process, and as it is not sufficient to plan once, the team must regularly re-plan as more data becomes known. It emerged that it is important to set business stakeholders expectations that this event will occur, and help manage their expectations, as the initial plan will often be optimistic. Most teams became aware of the need to re-plan after a few iterations, but often waited for another iteration or two prior to re-planning, causing some stress for the customer. Re-calibration contributes to Real Customer Involvement, Whole Team and Energized Work.

### 7.9 Discussion

*One of the things that is unsaid in the XP literature is how to be a customer*  
*— Martin Fowler [29]*

This chapter outlined eight practices that have emerged from our studies that begin to address the gap identified by Fowler on how to be a customer. The chapter also identifies the inherent interwoven relationships between the practices, and how they strongly contribute to Real Customer Involvement, Whole Team and Energized Work.

The emergent practices primarily support Real Customer Involvement by preparing the business representatives for their role (Customer Boot Camp), and providing opportunities for the business representatives to contribute towards the creation and refinement of what to build (Big Picture Up-Front, Roadshow and Re-calibration). The emergent practices primarily support Whole Team by providing opportunities for the programmers to develop empathy for the customer team (Customer’s Apprentice) and the end-user (Programmer On-Site). Finally, the emergent practices primarily contribute to Energized Work by reducing the intensity of the process (Pair Customering and Programmer Holiday).

The eight practices do not provide guidance on the mechanics or skills expected of the customer team. For example, no guidance is provided on how to design a process model or facilitate a workshop. This parallels the original XP practices, which also do not provide detailed guidance on skills expected of programmers. For example, the original XP practices do not address how to write an object-orientated program or use design patterns. Such skills are considered fundamental but are not elaborated upon in the process. As with all of the XP practices, the emergent customer practices focus on describing how to support teams to work together more effectively and how to ensure they “build the right thing”.

Our studies also indicated that an additional two customer-focussed practices are well-utilised by teams: Planning Game and Stories. In our studies, teams used the planning game with little variation from Beck’s original description [2]. In our interviews it was apparent that stories, and perhaps more importantly the conversations surrounding stories, were one of the key ways that
customers and programmers communicate what they are going to build. Most teams followed Beck’s [2] advice of writing a story on an index card, and using the story to prompt conversation between a customer and programmer. Most customers and programmers explained in their interviews that they had needed to work together to understand and develop:

- A way to break problems down to ensure that the context of a story was understood, so that it could be effectively estimated, prioritised and planned.
- A consistent way of writing stories.

One of the most interesting aspects of stories that emerged is that they seemed to bring the team together. The team learnt how to break down a problem into stories that were both meaningful to the business and at the right level of detail for the programmers, and to determine how big a story should really be.

The original customer-focused practices of Planning Game and Stories complement the eight practices described in detail in this section.

7.10 SUMMARY

In this chapter we have outlined the eight new customer practices that emerged from our studies:

- **Customer’s Apprentice:** A programmer joins the customer team for an iteration or two so that programmers can understand the complexity of the customer team’s role.
- **Programmer On-Site:** The customer schedules site visits for the programmers so that they can understand more about the end-users of the software.
- **Programmer Holiday:** The customer schedules an iteration of technically-focused tasks so that the customer can have some time to re-set the direction of the project or “time to think”.
- **Roadshow:** The customer demonstrates the currently working software and upcoming release plan to end-users and other interested stakeholders so that they can obtain feedback on the direction of the software.
- **Customer Pairing:** Two customers work collaboratively to provide a single-voice to the development team. The customers can be performing the same roles (e.g. two diplomats or two negotiators) or different roles (e.g. a negotiator and a user-interaction designer) on the customer team.
- **Customer Boot Camp:** A customer-focused training event for an XP project. It is beneficial for both customers and developers to attend this event.
- **Big Picture Up-Front:** A short period before the project iterations commence during which the business stakeholders and the customer team work together to set the direction of the project and create an initial release plan.
- **Re-Calibration:** Planning to adjust commitments and resources regularly based on what both customers and developers learn during the iterations.

The eight practices outlined help to support the second edition XP practices: Real Customer Involvement, Whole Team and Energized Work.
8. CONCLUSION

In the final chapter of this thesis we will summarise the thesis, discuss the resulting model and how that answers the research questions we started with, outline the contributions this thesis makes, and survey future work opportunities.

8.1 SUMMARY

We started this thesis in chapter one with a high-level introduction to the subject, and why we judged that the role of customers on XP projects was a topic worthy of further exploration. We then introduced our three research questions:

- What is the experience of being an XP customer?
- Who is the XP Customer? and
- What are the practices that support the XP customer?

In chapter two we provided a review of the literature that we undertook to help us set the context of our research. The particular research method we adopted requires a careful approach to literature, allowing it to inform, but not direct, the research study. The majority of literature would be used inductively as a way to help develop the theory by comparing and contrasting the literature with the research data during the analysis phase of the research.

In chapter three we introduced our research method: grounded theory. We described why grounded theory was an appropriate method to approach our research questions, as well as describing our approach to data collection, analysis and verification. This chapter also briefly touched upon the biases that we brought to the research and how grounded theory combined with the interpretative research perspective allowed us to mitigate those biases.

In chapter four we provided an overview of the eleven case studies that were generated from 66 interviews and associated observations. In chapter five we elaborated on six of those case studies in detail, in order to portray the experience of being an on-site customer effectively. Our key finding in this chapter was that although the on-site customer practice was perceived as rewarding and that practitioners would be reluctant to do a software project “any other way”; the serious risk concerning the unsustainability of the role was also highlighted.

In chapter six we described the customer teams, and the roles that emerged within those teams in our case studies. We were able to classify the ten roles into Collaboration Guides, Skill Specialists and Direction Setting roles. In chapter seven we described the eight practices that emerged from the case studies. These eight new practices supported Real Customer Involvement, Whole Team and Energized Work.

In this chapter we will pull all of our results together into a single model in the discussion section that follows. We will then outline our contributions and suggest future research.

8.2 DISCUSSION

In this thesis we set out to answer three research questions concerning the role of the XP customer. The first question was simply what is the experience of being an XP customer? To answer that question we interviewed over 60 people on eleven different XP projects and observed the interactions between the XP customer and other project roles. Chapter four, entitled “Overview of the Cases” and chapter five entitled “Being an On-Site Customer”, describe that experience, predominantly from the perspective of the customer.

Our findings strongly indicated that, in the majority of cases, the customer was passionate about their role on the project and using XP. For instance, they indicated that they “wouldn’t do it any
other way”. In the interviews both the customers and the programmers conveyed how much they had enjoyed the direct conversations and connections they had with each other, and felt that this approach contributed to creating a better quality system. Conversely, our findings also indicated that the customers were working more intensely and for longer hours than on traditional projects; the customer was not working at a sustainable pace. For instance, one customer commented upon the project “needing three of me” as she had been over-stretched concerning her working hours, and also commented on the difficulties and stress caused by the role (“look at all these grey hairs”).

Our next research question focused on who was the XP customer? This question sought to clarify the types of people who undertook the role, and to also understand if it was a single person or a team undertaking the role. Our findings concluded that it was always a team of people, what is more, the team of people tended to consist of up to ten distinct roles. The roles emerged as follows:

- **Collaboration Guides:** Geek Interpreter, Political Advisor and Technical Liaison. The Geek Interpreter contributes to enhancing the collaboration within the team, particularly between the customer and programmers. The Political Advisor contributes to enhancing the collaboration between the customer and the larger organisational life of the business, ensuring the customer is better able to navigate the political dimensions necessary for the project to succeed. The Technical Liaison contributes to enhancing the collaboration between this project and other related projects, as well as any technical specialist silos that might exist within the organisational structure. The collaboration guide roles that emerged from this research expand the collaboration facilitator role initially outlined by Highsmith [37].

- **Skills Specialists:** Acceptance Tester, UI Designer and Technical Writer. All of the specialist roles support the customer to undertake their on-site customer responsibilities. The Acceptance Tester supports the customer by assisting with the acceptance testing of the software. The UI Designer supports the customer by lending their UCD (User-Centred Design) skills to assist the customer in deciding and designing what to build. The Technical Writer supports the customer by lending their specialist writing skills to create user documentation to accompany the software when released. Each specialist skill-set helps to reduce the workload of the customer and improves the resulting software released. The skills specialist roles that emerged from this research support Beck’s identification of these potential roles that he outlined in XP second edition[32].

- **Direction Setting:** Negotiator, Diplomat, Super-Secretary and Coach. Each of these roles contributes significantly to deciding what to build. The Negotiator is the lead role; responsible for working with a large group of end-users and other stakeholders to create a single-voice describing what to build to the development team. The Diplomats are the end-users and other stakeholders that are around that negotiating table. They provide their experiences and perceptions of what is needed, and work with the Negotiator and other Diplomats to create a single-voice. The Super-Secretary supports the Negotiator by providing administrative support and also by filling in for them with the programmers when the Negotiator and Diplomats are not available. All of these roles work towards ensuring there is a strong and coordinated focus on understanding and communicating what to build, which is what Beck initially hoped to achieve with the onsite-customer practice [2]. We used the analogy from DeMarco [40] where he likened the role of an analyst to Kissinger negotiating for peace in the Middle East to help form the emerging roles that allow Beck’s initial vision to be fulfilled.

Our finding that a team of people implements the on-site customer practice is useful, but we then wanted to know if this finding had any bearing on the passion and overwork issues found in our
earlier chapter. Figure 8-1 depicts our interpretation of the relationship. The three corners of the figure show the sets of roles and our interpretation of their impact on both passion and overwork which are depicted at the centre. Not all teams studied implemented all of these roles, but the teams we saw with the majority of the roles implemented (for example, OwlCorp) tended to report significantly less overwork, and hence the impact of their passion was more pronounced. Therefore, our conclusion is that the team roles help to increase the passion, and decrease the overwork experienced by the customer.

![Figure 8-1: Impact of Customer Team Roles on the Experience of Being a Customer](image)

Notice that each of the role groups helps in a particular way to support this change. The Collaboration Guides tend to further enhance the collaboration and sense of team, which so many customers were passionate about retaining, and had enjoyed experiencing to a much greater degree on an XP project. The Skill Specialists tend to increase the customer’s certainty (or abilities) concerning a task such as writing a story, or testing the software, therefore improving their perception and passion for the task at hand. The Direction Setting roles primarily help to provide a clear focus for the core function of the customer: deciding what to build. The roles also assist to reduce the workload experienced by the customer, particularly the Skill Specialists, Technical Liaison and Super-Secretary. These roles should address the problem of overwork and reduce comments from the customer such as “we needed three of me”.

Our final question for this research was: what are the practices that will support the XP customer? This question sought to uncover the practices the customers and developers were using
to enhance the effectiveness of the on-site customer role on an XP software development project.
Eight new practices emerged:

- **Customer’s Apprentice**: A programmer joins the customer team for a few iterations so that programmers can understand the complexity of the customer teams’ role as well as assisting to reduce the overload experienced by the customer.
- **Programmer On-Site**: The customer schedules site visits for the programmers so that the programmers have the opportunity to understand more about the end-users.
- **Programmer Holiday**: The customer schedules an iteration of technically focused tasks so that the customer can have a break from the iterations and time to re-set the project direction.
- **Roadshow**: The customer demonstrates the software and planned stories to end-users and other stakeholders so that the customer can obtain feedback on the direction of the software.
- **Customer Pairing**: Two customers work collaboratively to provide a single-voice to the development team. The customers could be performing the same roles (e.g. two diplomats) or different roles (e.g. a negotiator and a user-interaction designer) on the customer team.
- **Customer Boot Camp**: A customer-focused training event for both customers and developers. The training should cover the role of the customer as well as the techniques (e.g. story writing) and practices (e.g. Programmer On-site) with opportunities for the whole team to put what they have learnt into practice in a safe and fun environment (e.g. a simulation).
- **Big Picture Up-Front**: A relatively short period of design amongst the business stakeholders and the customer team to set the direction of the project and create an initial release plan. The most common technique used to develop the big picture are facilitated workshops involving the stakeholders, customer team and developer representatives.
- **Re-Calibration**: The customer schedules regular sessions (every few iterations) with the end-users and stakeholders to adjust commitments and resources. The adjustments will be made to take into account what both customers and developers learn during the iterations.

Our studies were all practicing XP as described by Beck [2] initially, commonly referred to as XP first edition. We noticed that the eight new practices helped to enhance or support the three second edition [32] practices: Real Customer Involvement, Whole Team and Energized Work. As with the roles uncovered by our second research question, we also noticed that the practices changed the experience of the customer. Our interpretation of the impact of the practices to the variables of passion and overwork is depicted in Figure 8-2. The practices associated with Whole Team and Real Customer Involvement are shown as promoting the increase in passion, with the practices associated with Energized Work are shown as decreasing overwork. The practices are depicted in bold with two asterisks where those practices contribute primarily to one of the second edition XP practices and in faded text where they are a secondary contributor.
Our conclusion is that the practices help to increase the Real Customer Involvement and the feeling of an effective Whole Team. It is these points that customers and developers identify as those that increase their passion or preference to not “do it any other way” than XP. Additionally, and just as importantly, there are a significant number of practices that directly address overwork, providing a break from the intensity of XP, and reducing the amount of overwork experienced by the customer.

Figure 8-3 depicts the significant change to the passion to overwork ratio when the team roles and customer practices are combined. At the centre of the figure we depict the expected increase of passion and decrease of overwork that accompanies the experience of being an on-site customer when all of the team roles and practices are utilised. The figure categorises the roles and practices by Real Customer Involvement, Whole Team and Energized Work. For readability purposes only the primary contributors to each second edition practice are depicted, but it is important to remember that the team roles and practices create a much more interwoven set than that depicted in Figure 8-3.

Finally, the emphasis on workload in our interviews and observations suggests that even the roles and practices that we identify do not completely eliminate overwork, but they do reduce its impact. The roles and practices reduce overwork to a more manageable level, but perhaps customers will also need to take a break between projects. As one of the customers with a development background noted in her interview concerning overwork:

“This is what an analyst does. This is it ... You always have busy times and not so busy times, and you just manage that. If you are not suited to that way of working you don’t stay as an analyst.”

—Customer, SparrowCorp
8.3 **Contributions**

In this thesis we studied the role of customers on extreme programming projects. We explored the experience of being an on-site customer, and investigated who undertook that role, that is, whether it was a single person or a team of people. Finally, we uncovered what practices the customers used to enhance their effectiveness. We believe our study makes the following contributions to the understanding of agile software development:

- We discovered that being an on-site customer is perceived as rewarding, and something that many customers would want to do again. However, the sustainability of the role was brought into question. There was a strong preference to use the XP approach to deciding and describing what needs to be built, as the direct connections established with the programmers, and the subsequent collaboration, were seen as contributing to a better quality of system than traditional approaches. Conversely, the experience is also described as intense and many customers were experiencing fatigue from overwork, and as such the on-site customer practice was not perceived as sustainable in its current form. The thesis brings to the attention of the agile community the inherent unsustainability of the on-site customer practice.

- We discovered that a team of people undertake the on-site customer practice, and in this thesis we provide a model of the ten roles that were discovered to comprise an effective customer team. Initial descriptions of XP suggested the role could be undertaken by a single person, but no cases were found where this was the situation. A team, often an informal team, always filled the on-site customer practice. This thesis provides a model of the ten roles that emerged from the case study analysis. The roles include collaboration guides, skill specialists and direction setting roles. This model shows how effective teams are distributing...
the workload across a number of people to reduce the unsustainability of the on-site customer practice. Therefore this model offers useful insight and guidance when the customer team is being assembled.

- We discovered eight practices that support the customer and programmers to enhance the effectiveness of the on-site customer practice. The eight practices support the team to obtain Real Customer Involvement on the project, and enhance the collaboration and effectiveness of the Whole Team. Additionally, six of the practices also assist in reducing the overwork experienced by the customers, so that they are more likely to experience what Beck calls Energized Work. The thesis describes these practices such that they can be utilised by those wishing to improve the effectiveness of the on-site customer practice on their XP project.

It may not be possible to entirely remove overwork from the on-site customer practice. It appears that the combination of the team roles and practices help to minimise the overwork, but do not completely eliminate it.

### 8.4 Future Work

This thesis suggests a number of interesting research opportunities. Our current study has identified the importance of these questions but a more focussed study is necessary to determine the answers. Questions of immediate interest are:

- Increasing the scope of the substantive theory beyond XP projects to include other agile methods such as Scrum. This opportunity is particularly relevant given the rising dominance of Scrum in the agile community.
- Obtaining more depth on any of the roles or practices outlined in this thesis. Big Picture Up-Front and the roles that the Negotiator and the UI Designer play during this period are critical to the success of every project, and as such we would recommend starting with this practice and those roles.
- Continuing to explore the issue of overwork as part of the inherent nature of the customer role. It appears worthwhile to turn our attention to entrepreneurial domains in our search for answers to this longer-term question concerning the career sustainability of an XP customer.
- Using a quantitative research method to explore how common these results are across many teams.
- Developing a model, most likely based on both qualitative and quantitative research, that highlights the number of people required in the customer team. Is this number relative to the programmers, business complexity or number of stakeholders or a combination of all of the above?
- Obtaining more depth on the impact of outsourcing on agile projects. This research provides some initial insights, but practitioners need further advice and guidance concerning the use and impacts of different contracting structures.

All of these opportunities represent useful additions to the knowledge of the agile community.
REFERENCES

References


References


References


[63] ResearchWare. (2006), HyperResearch Qualitative Analysis Tool.


[65] A. Cooper. (1999), The Inmates are Running the Asylum: Why High-Tech Products Drive us Crazy and How to Restore the Sanity., Sams.


References


References


[117] M. Cohn. Certified Scrum Product Owner Training, 


[121] F. Keenan, and D. Bustard. (2005), BPUF: Big Picture Up Front. in "Sixth International Conference of Extreme Programming and Agile Processes in Software Engineering" (H. Baumister, M. Marchesi, and M. Holcombe, Eds.), Springer-Verlag, Sheffield, United Kingdom.


APPENDIX A: APPROVED HEC APPLICATION

HUMAN ETHICS COMMITTEE

Application for Approval of Research Projects

Please write legibly or type if possible

Note: The Human Ethics Committee attempts to have all applications approved within three weeks but a longer period may be necessary if applications require substantial revision.

1 NATURE OF PROPOSED RESEARCH:
   (a) Staff Research / Student Research (delete one)
   (b) If Student Research - Degree: PhD (Information Science)
       Course Code: …INFO 690…
   (c) Project Title: Exploring the role of customers on eXtreme programming projects…

2 INVESTIGATORS:
   (a) Principal Investigator
       Name …………… Angela Michelle Martin
       e-mail address …………angela@mcs.vuw.ac.nz………………………………
       School/Dept/Group …. Information Management
   (b) Other Researchers Name Position
       ……………..none……………………………………………………
       ………………………
   (c) Supervisor (in the case of student research projects)
       ………Dr Robert Biddle & Dr James Noble……………

3 DURATION OF RESEARCH
   (a) Proposed starting date for data collection 15 September 2003………..
       (Note: that NO part of the research requiring ethical approval may commence prior to approval being given)
   (b) Proposed date of completion of project as a whole 31 March 2008

4 PROPOSED SOURCE/S OF FUNDING AND OTHER ETHICAL CONSIDERATIONS
   (a) Sources of funding for the project
       Please indicate any ethical issues or conflicts of interest that may arise because of sources of funding
       e.g. restrictions on publication of results
       ………No funding has been sought to aid this research……………………
   (b) Is any professional code of ethics to be followed Yes
       If yes, name
       ……………………………………………………………………………….
Appendix A: Approved HEC Application

(c) Is ethical approval required from any other body

No

If yes, name and indicate when/if approval will be given

...........................................................

5 DETAILS OF PROJECT

Briefly Outline:

(a) The objectives of the project

○ To explore and describe how the XP concept of an on-site customer has been implemented in both New Zealand based and international software development projects.

○ To propose an extension(s) to the XP “on-site customer” practice based on the case study and related literature findings.

(b) Method of data collection

○ A combination of qualitative data collection techniques will be used including semi-structured interviews, observations of the project team at work by the researcher and review of artefacts/documents produced as part of the project work.

(c) The benefits and scientific value of the project

○ To explore how software development methodologies are used by practitioners on real projects so that we may better understand this process and formulate and present methodologies that will allow practitioners to complete software development projects with less cost and schedule overruns and met the needs of the users of the systems.

(d) Characteristics of the participants

○ Software developers, software development project leaders and domain/business experts that are working (directly or indirectly) on the XP project.

(e) Method of recruitment

○ Organisations will be initially approached via XP community interaction mechanisms including email, lists, user groups and conference participants with a brief outline of the intended research.

○ The researcher will then work with each interested organisation to develop a mutually agreeable research proposal or terms of reference which clearly outlines the purpose of the research, the scope of the research, the time commitments expected from the organisation’s staff. This proposal will supplement the information sheet and consent form and the initial draft of this proposal is attached to this application.

○ Participants on the project nominated by the organisation for study will then be approached, participation will be voluntary.

(f) Payments that are to be made/expenses to be reimbursed to participants

○ None

(g) Other assistance (e.g. meals, transport) that is to be given to participants
o  **None, the researcher will go to the participant**

(h) Any special hazards and/or inconvenience (including deception) that participants will encounter

 o  **No deception will be involved. The organisation will be affected by ‘lost’ project time but this time could be considered part of the general post-implementation reviews that are best-practice to implement.**

(i) State whether consent is for (delete where not applicable):

(i) the collection of data

(ii) attribution of opinions or information

(iii) release of data to others

(iv) use for a conference report or a publication

(v) use for some particular purpose (specify)

……………………………………………………………………………………………

Attach a copy of any questionnaire or interview schedule to the application

**Attached**

(j) How is informed consent to be obtained (see paragraphs 4.31(g), 5.2, 5.5 and 5.61 of the Guidelines)

(i) the research is strictly anonymous, an information sheet is supplied and informed consent is implied by voluntary participation in filling out a questionnaire for example (include a copy of the information sheet)

**No**

(ii) the research is not anonymous but is confidential and informed consent will be obtained through a signed consent form (include a copy of the consent form and information sheet)

**Yes**

(iii) the research is neither anonymous or confidential and informed consent will be obtained through a signed consent form (include a copy of the consent form and information sheet)

**No**

(iv) informed consent will be obtained by some other method (please specify and provide details)

**No**

……...

With the exception of anonymous research as in (i), if it is proposed that written consent will not be obtained, please explain why

**Not applicable**

(k) If the research will not be conducted on a strictly anonymous basis state how issues of confidentiality of participants are to be ensured if this is intended. (See paragraph 4.3.1(e) of the Guidelines). (e.g. who will listen to tapes, see questionnaires or have access to data). Please ensure that you distinguish clearly between anonymity and confidentiality. Indicate which of these are applicable.
(i) access to the research data will be restricted to the investigator ...No

(ii) access to the research data will be restricted to the investigator and their supervisor (student research) Yes

(iii) all opinions and data will be reported in aggregated form in such a way that individual persons or organisations are not identifiable Yes

(iv) Other (please specify)
Not applicable ........................................

(l) Procedure for the storage of, access to and disposal of data, both during and at the conclusion of the research. (see section 7 of the guidelines). Indicate which are applicable:

(i) all written material (questionnaires, interview notes, etc) will be kept in a locked file and access is restricted to the investigator and supervisor

Yes

(ii) all electronic information will be kept in a password-protected file and access will be restricted to the investigator and supervisor Yes

(iii) all questionnaires, interview notes and similar materials will be destroyed:
(a) at the conclusion of the research No

or
(b) ___3___ years after the conclusion of the research Yes

(iv) any audio or video recordings will be returned to participants and/or electronically wiped Yes

(v) other procedures (please specify):

...None……

If data and material are not to be destroyed please indicate why and the procedures envisaged for ongoing storage and security

...Not applicable.................................................................

(m) Feedback procedures (See section 8 of the Guidelines). You should indicate whether feedback will be provided to participants and in what form. If feedback will not be given, indicate the reasons why.

○ Participants will be involved in continual feedback to confirm the researchers interpretation of their comments and to validate findings. Participants will also be provided with a copy of the resulting publications.

(n) Reporting and publication of results. Please indicate which of the following are appropriate. The proposed form of publications should be indicated on the information sheet and/or consent form.

(i) publication in academic or professional journals Yes

(ii) dissemination at academic or professional conferences Yes
(iii) deposit of the research paper or thesis in the University Library (student research)  

Yes

(iv) other (please specify)

None

Signature of investigators as listed on page 1 (including supervisors).

NB: All investigators must sign before an application is submitted for approval

[paper copies are signed by Angela Martin and her thesis advisors]
APPLICATIONS FOR HUMAN ETHICS APPROVAL

CHECKLIST

• Have you read the Human Ethics Committee Guidelines?
• Is ethical approval required for your project?
• Have you established whether informed consent needs to be obtained for your project?
• In the case of student projects, have you consulted your supervisor about any human ethics implications of your research?
• Has your supervisor read and signed the application?
• Have you included an information sheet for participants which explains the nature and purpose of your research, the proposed use of the material collected, who will have access to it, whether the data will be kept confidential to you, how anonymity or confidentiality is to be guaranteed?
• Have you included a written consent form?
• If not, have you explained on the application form why you do not need to get written consent?
• Are you asking participants to give consent to:
  - collect data from them
  - attribute information to them
  - release that information to others
  - use the data for particular purposes
• Have you indicated clearly to participants on the information sheet or consent form how they will be able to get feedback on the research from you (e.g. they may tick a box on the consent form indicating that they would like to be sent a summary), and how the data will be stored or disposed of at the conclusion of the research?
• Have you included a copy of any questionnaire or interview checklist you propose using?
• Has your application been seen by the head of your school or department (or the person given responsibility to consider applications on behalf of the head (see Guidelines 5.3.2).

PLEASE FORWARD YOUR COMPLETED APPLICATION FORM TO THE SECRETARY, HUMAN ETHICS COMMITTEE, HUNTER BUILDING OR, IN THE CASE OF APPLICATIONS FROM SCHOOLS OR DEPARTMENTS WITH AN APPROVED ETHICS SUB-COMMITTEE, TO THE CONVENER OF THAT SUB-COMMITTEE.
Attachment A: Participant Consent Cover Letter

[Participant’s name]
[Company name]
[Employment address]

Dear [Participant’s name],

**PhD Research into the XP customer role**

Thank you for indicating, in our recent [email] conversation, that you would be interested in being interviewed for this research.

Prior to conducting the interview, Victoria University of Wellington requires that I obtain your written informed consent. This consent is a normal part of any research project and forms one criterion of the Faculty of Commerce and Administration’s Human Ethics Committee Guidelines that I must meet.

Attached for your information, therefore, is:

- A copy of the proposal submitted to [Organisation manager who approves research in organisation]. This proposal outlines the purpose, scope & approach to the project.
- An interview agenda, so that you will have a good sense of the nature of the questions I would like to ask during an interview with you.
- An information sheet that covers your rights throughout this process.
- A consent form that you can sign and return to me [in the enclosed envelope] if you decide to participate in this research.

I would like to emphasise that you do not have to participate in this research and that you are free to withdraw from this research without explanation up to the 30th of November 2003.

If you require any further information or if you would prefer not to be interviewed, please feel free to email me at angela@mcs.vuw.ac.nz or call me on +64 21 668 511. Alternatively you may contact either of my supervisors Robert Biddle (Robert@mcs.vuw.ac.nz or +64 4 463 5833) or James Noble (kjx@mcs.vuw.ac.nz or +64 4 463 6736).

Yours sincerely

Angela Martin
(PhD Student)
Phone: +64 21 668 511
E-mail: angela@mcs.vuw.ac.nz
Attachment B: Information Sheet

ETHICAL APPROVAL FOR INTERVIEW: INFORMATION SHEET

Nature and purpose of research

This research project contributes towards the requirements for a PhD degree. The purpose of the research is to investigate the application of a software development process, and to critically assess the issues associated with the use of this process.

The research will be carried out by interviewing up to [insert number] staff members of [Company name]. An interview agenda has been attached to this information sheet outlining what information will be sought during the interviews. No personal details or information will be collected.

Purpose of the material collected

Material collected from the interviews will be used within my research project as evidence of the application of a software development process and the organisational issues associated with its use. The final research report will be published as a PhD thesis held at the Victoria University Library. Conference and journal publications will also be written based on this research. Written interview notes will be stored at my home and/or at my office at the University, and destroyed at the completion of the project.

Before the interview proceeds, I need to be sure that I have permission of your employer for you to be interviewed as the consent form has a statement to this effect. To aid you through this process I will provide a signed copy of the attached proposal to you.

Confidentiality

All raw data will be kept confidential to myself, my supervisors Robert Biddle and James Noble, and a professional transcriber Joanne Stevens. The research report (or any conference papers or journal articles that may result from the study) will not identify you or your employer. There will be an opportunity to review any written notes that result from the interview to ensure factual material is recorded accurately.

Timing of the Project

Initial interviews are scheduled for the [first week of November 2003], however follow-up interviews can be organised for 2004 should you wish to continue to participate in this research. The interview will take about one hour. The possibility of a further interview will be discussed with you at the time of the first interview.
Attachment C: Consent Form

Victoria University of Wellington
Consent to Participation in Research

Project Title: Exploring the role of customers on eXtreme programming projects

I have been given and have understood an explanation of this research project and the confidentiality conditions. I have had an opportunity to ask questions and have them answered to my satisfaction.

I agree to be interviewed by Angela Martin for the purpose of this research for her PhD degree and resultant publications and conference publications. I consent to the collection and use of my perceptions, experiences, opinions and information in this research.

I understand that I may withdraw from this research up to the 30th of November 2003 without penalty or explanation.

I confirm that I

DO DO NOT

have approval by my employer to participate in this research project.

Do you agree to have interviews tape-recorded?

YES NO

Would you like to receive a copy of any 2004 publications that are based on these interviews?

YES NO

If yes, please provide an email address, or another contact mechanism (phone number / street address), that we can use to contact you to organise to send a copy of these publications to you.

__________________________________________________________________________

__________________________________________________________________________

Name: ___________________________
Signed: _________________________
Date: ___________________________
ATTACHMENT D: RESEARCH PROPOSAL SUBMITTED TO ORGANISATION

ON-SITE CUSTOMER RESEARCH PROPOSAL

Introduction
The purpose of this document is to establish a research project with [Company]. In order to achieve this purpose this document covers the following:

- Objectives of the project
- Scope of the project and
- Approach to the project.

This proposal reflects an initial project concept that focuses on the on-site customer related practices of your implementation of eXtreme Programming. We look forward to working with you to mould it into a project that will meet our mutual objectives.

Objective

- To recommend the key on-site customer practice improvements to be implemented on your next eXtreme Programming project
- To increase academic and industry knowledge of the on-site customer practices that work and do not work when implementing agile development practices.

Scope

<table>
<thead>
<tr>
<th>Is</th>
<th>Is Not</th>
</tr>
</thead>
<tbody>
<tr>
<td>To explore the on-site customer related practices utilised on the project including:</td>
<td>To explore the development specific practices including:</td>
</tr>
<tr>
<td>- Release &amp; iteration planning</td>
<td>- Pair programming</td>
</tr>
<tr>
<td>- Writing user stories</td>
<td>- Refactoring</td>
</tr>
<tr>
<td>- Estimation process</td>
<td>- Metaphor</td>
</tr>
<tr>
<td>- Priority planning</td>
<td>- Testing</td>
</tr>
<tr>
<td>- Planning meetings</td>
<td>- Collective ownership</td>
</tr>
<tr>
<td>- Contractual issues</td>
<td>- Continuous integration</td>
</tr>
<tr>
<td>- On-site customer involvement</td>
<td>- 40-hour week</td>
</tr>
<tr>
<td>- Coding standards</td>
<td></td>
</tr>
</tbody>
</table>

- To understand and document the implementation of the ‘text book’ processes of the on-site customer including:
  - Activities that worked well for the team when using these processes (what we must keep on the next project)
  - Issues and difficulties the team encountered with these processes (what we need to change on the next project)
  - Suggested improvements to these processes (how we might change these processes on the next project)

- To implement change or prepare a change management plan.

- To provide recommendations for improvement based on literature and research findings.
- To provide recommendations for improvement based on the consultant’s experience.
Approach

We will utilise a mixture of semi-structured one-on-one interviews with the project team and project artefact review to understand the project team’s experience with this method.

The *interviews* will focus on the key project roles associated with the on-site customer practice including:

- Customer
- Programmer(s)
- Coach
- Big Boss
- Business stakeholders

The interviews will occur in a meeting room at your premises. The on-site location will minimise the disturbance to the interviewee’s day-to-day activities. All interviews will be taped and later transcribed. The interviewees will be asked to validate the transcription of the interview and the interpreted findings.

The *artefact review* will focus on the key on-site customer related artefacts including user stories, plans and project walls. This review will allow us to understand the adherence and useful variations to the eXtreme Programming standards.

The *costs* to your company are expected to be minimal and time based only. The total time required of each identified team member (maximum of 7) is expected to be between 3 to 4 hours over a period of 2-3 months. The maximum period in 1 day for any team member will be 1 hour.

All interactions would be arranged around the team members existing commitments, so disruption should remain minimal.

All raw data obtained during this project will remain *confidential* to the researcher, Angela Martin, her direct supervisors, Robert Biddle and James Noble, and a professional transcriber, Joanne Stevens. All findings will be reported at an aggregate level with the company and individuals retaining their anonymity. However [Company] is welcome to be identified in subsequent publications should they wish to be acknowledged as a practice leader.

Approval to Proceed

[Company] authorises this project to proceed under the conditions stated in this proposal.

Name:                                             Signature:

Date:

---

29 Please note that we requested the removal of the “approval to proceed” section based on feedback from participants during our first interviews in the last quarter of 2003. We received HEC approval to remove this section in the first quarter of 2004.
Attachment E: Proposed Interview Agenda

INTERVIEW DETAILS:

Date: To be decided

Topic(s): The implementation of the planning & estimation practices on the xxxCorp xxx Project

OUTCOME:

- To recommend the key planning & estimation practice improvements to be implemented on your next project
- To increase academic and industry knowledge of the planning & estimation practices that work and do not work when implementing agile development practices.

AGENDA:

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Agree outcome, agenda &amp; rules for the interview</td>
<td>05 mins</td>
</tr>
<tr>
<td>2.</td>
<td>Previous project and software development process experience of interviewee</td>
<td>10 mins</td>
</tr>
<tr>
<td>3.</td>
<td>Planning &amp; estimation practices &amp; processes (what we did on the project)</td>
<td>15 mins</td>
</tr>
<tr>
<td>4.</td>
<td>Activities that worked well for the team when using these processes (what we must keep on the next project)</td>
<td>10 mins</td>
</tr>
<tr>
<td>5.</td>
<td>Issues and difficulties the team encountered with these processes (what we need to change on the next project)</td>
<td>10 mins</td>
</tr>
<tr>
<td>6.</td>
<td>Suggested improvements to these processes (how we might change these processes on the next project)</td>
<td>05 mins</td>
</tr>
<tr>
<td>7.</td>
<td>Wrap-up</td>
<td>05 mins</td>
</tr>
<tr>
<td></td>
<td>Review progress</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Next steps</td>
<td></td>
</tr>
</tbody>
</table>

RULES:

- The interview will be tape recorded to reduce the risk of the interviewer not obtaining all of the information provided by the interviewee during the interview.
- The transcription of the interview and subsequent findings will be provided to the interviewee to ensure all information has been recorded and interpreted accurately.
- To ensure all items are covered in sufficient detail any discussion concerning development practices will be noted in a register for further discussion.