Learning from each other: An interdisciplinary approach to Software Engineering

Damien Beard

A subthesis submitted in partial fulfillment of the degree of
Bachelor of Software Engineering (Honours) at
The Department of Computer Science
Australian National University

October 2014

Typeset in Palatino by TEX and LATEX 2ε.

Except where otherwise indicated, this thesis is my own original work.

Damien Beard
24 October 2014

In memory of Tash.

Acknowledgements

I would like to thank my supervisors Dr Shayne Flint and Dr Ramesh Sankaranarayana for encouraging me to reach beyond my limits. I am grateful for your patience and the many, many hours that you spent talking with me, when I would turn up at your offices wanting to have a chat 'for just a few minutes'. Without you, this project would never have been possible.

Along with my supervisors, I am grateful to Lynette and all her patience with me throughout this year. You have given me invaluable advice and guidance, and I cannot thank you enough.

I am grateful to everyone who helped proofread this thesis in the lead up to submission, including Ellie, Ryan, Richard, Leana and Adam. Your time is valuable, and I am humbled that you would spend it reading my work.

I would like to acknowledge St Francis Xavier College. It was here that I began to discover the amazing things that could be done with computers, and learned that my abilities were only limited by my imagination. I thank them for the support they continue to give me, years after graduation. I also thank them for all the things they taught me that I thought were useless at the time. I was wrong.

I thank my family, and in particular my parents. I am so grateful for your love and support, and I should tell you this more often. Also, thanks for not scolding me when I told you that I was living on a diet of LCM bars and Pizza Pockets, I promise to improve my eating habits after graduation.

Thankyou to the staff at NIMHR including Alison, Brendan and in particular Becca Randall and Lou Farrer. I will never be able to thank you enough for all of your help and guidance this year. This project is a testament to your open-mindedness.

I thank all participants of the project, who agreed to partake in the case study. If you had not been so generous with your time, this project would not have happened.

Thank you to the Department of Foreign Affairs and Trade, who took a chance on me three years ago, and have supported me throughout this degree.

To the friends that have supported me throughout this degree, you have done more for me than you'll ever know. You have my sincere and honest gratitude.

Finally to the friends I have here at ANU, including Sahan, Justin, Arrian, the Honours cohort, and everyone else I have met! Thanks for making the last four years so enjoyable. And to Emily McAlister. For all of those late night chats. For every time you proofread my work. For every group project, every time we laughed, every time we fought, for every time you pushed me to be my best. You are a true and honest person, and I consider you to be one of the most admirable people I know. Thank you for all of your support throughout this degree, and I look forward to many years of friendship ahead.

Abstract

The discipline of software engineering has historically been inward facing in nature. Limited research has been performed regarding the adoption of practices from external disciplines for the purposes of improving research methods. Virtually no research has been performed regarding the adoption of external practices for the purposes of improving software development techniques. This project serves to fill this gap.

This project investigates the feasibility of sharing knowledge between software engineering and other disciplines for the purposes of improving software development practices. The discipline of mental health is identified as a potential candidate from which to learn, in particular the practice of Community Based Participatory Research, which was recognised as being similar in nature to agile software development. An exploratory case study is designed in order to identify differences in implementation across these practices, generate hypotheses regarding the implementation of these practices, and identify possible opportunities for knowledge sharing between the disciplines.

The case study consists of interviews conducted with experts in software engineering and mental health research. Participants are questioned on their understanding and experiences with the investigated practices, along with their opinions towards the adoption and sharing of interdisciplinary practices. A general systems model is created in order to facilitate these interviews, and was found to be effective for gathering data from participants.

Participants from software engineering and mental health research believed that practices from within their disciplines could be improved by adopting adapted practices from external domains. Participants from both disciplines suggested practices that could be offered to other disciplines from within their domain. Participants generally identified that their research areas were inward-facing and silo based. Opportunities for software engineering and mental health to collaborate were identified, in the areas of communication, approaches to development and approaches to handling time constraints. The feasibility of interdisciplinary knowledge sharing within software engineering was established and recommendations were made regarding future interdisciplinary work within software engineering.

Contents

Acknowledgements **vii**

Abstract **ix**

1 Introduction **1**
1.1 Objective . . . 3
1.2 Project Plan . . . 3
1.3 Contribution . . . 4
1.4 Document Outline . . . 4

2 Background and Related Work **7**
2.1 Interdisciplinary attitudes and approaches to Software Engineering . . . 9
2.1.1 Historically inward-looking nature of Software Engineering Research . . . 9
2.1.2 Interdisciplinary work to date . . . 12
2.2 Software Engineering Practices and Research . . . 16
2.2.1 Laws of Software Engineering . . . 17
2.2.2 Software Engineering Practices and Methodologies . . . 18
2.2.3 Agile Development . . . 19
2.2.4 The inherent Complexity of software . . . 21
2.2.5 Object-Oriented Design, Abstraction and Modelling . . . 22
2.2.6 Systems Thinking and Systems Modelling . . . 24
2.3 Relevant literature in Mental Health . . . 26
2.3.1 Current state of Mental Health . . . 26
2.3.2 Participatory Design and Community Based Participatory Research . . . 27

3 The Agile and CBPR Systems Model **31**
3.1 Approach to modelling . . . 33
3.2 User-focused, Iterative Model . . . 33
3.3 Limitations of Model . . . 41

4 Methodology **43**
4.1 Approach to research . . . 43
4.1.1 Qualitative vs Quantitative Research . . . 43
4.1.2 Determining a qualitative research tradition . . . 44
4.1.3 Interviews for data collection . . . 45

4.2 Case Study Design . . . 46
4.2.1 Objectives . . . 46
4.2.2 Procedures and Protocols for data collection . . . 47
4.2.3 Collecting evidence . . . 48
4.2.4 Analysis of collected data . . . 50
4.2.5 Reporting . . . 52
4.3 Validity of Case Study . . . 52

5 Results and Discussion **55**
5.1 Overview of Interviews . . . 55
5.2 Analysis of Data . . . 56
5.2.1 Major themes through agile case . . . 61
5.2.1.1 Deviant and Outlying cases . . . 67
5.2.2 Major themes through CBPR case . . . 67
5.2.2.1 Deviant and Outlying cases . . . 72
5.2.3 Cross-case comparison . . . 73
5.2.4 Analysis of Systems Model . . . 75
5.2.4.1 Deviant and Outlying cases . . . 77
5.2.5 Identified areas for further research . . . 78
5.3 Implications for Interdisciplinary work within Software Engineering . . . 84
5.3.1 How this work fits into existing research . . . 86
5.4 Strengths and Limitations of Research . . . 87
5.4.1 Influence of the researcher on the data . . . 89
5.4.2 Concluding remarks . . . 89

6 Conclusion and Future Work **91**
6.1 Project Findings . . . 91
6.2 Future Research Directions . . . 93
6.3 Conclusions . . . 94

A Project Change of Direction **95**

B Interdisciplinary Research within Software Engineering **99**

C Case Study Information Sheet **101**

D Case Study Invitation E-mail and Consent Form **103**

E Software Engineering Interview Questions **105**

F Mental Health Interview Questions **107**

G Systems model for Software Engineering Interviews **109**

H Systems model for Mental Health Interviews **111**

I Initial Coding Schema from Open Coding Activities **113**
I.1 Software Engineering Open Codes 113
I.2 Mental Health Open Codes 115

J Results of cluster analysis on open codes **117**

K Annotated Systems Models **121**

Bibliography **127**

Chapter 1

Introduction

Software engineering research has a well-defined history of being insular, seeking inspiration almost exclusively from within the self-contained walls that the discipline has established for itself (Glass et al. 2002). Limited and sporadic research has been undertaken in order to discover techniques from other disciplines for the purposes of improving research practices (Sim et al. 2001). However, there has been virtually no effort to date for seeking inspiration regarding the practices for developing and maintaining software. As a discipline, software engineering is concerned with the practices surrounding the design, building, testing and maintenance of software applications (Parnas 1999). However research within software engineering takes an inward-facing view towards these practices, making little effort to learn from other disciplines regardless of the potential two-way benefit that could follow. This thesis explores the feasibility of looking to other disciplines for inspiration and learning opportunities, and in turn sharing knowledge across interdisciplinary boundaries.

Software systems are becoming increasingly prevalent throughout society, and the need to create quality software has never been higher. It may be fair to assume that given a $1 billion budget, a fully qualified software development team and a seemingly clear set of requirements, the delivery of a successful software project is all but given. Unfortunately, as the United States Air Force experienced, even projects with large budgets and qualified team members are still susceptible to underperformance and ultimately failure. The US Air Force spent over $1 billion on an enterprise resource planning (ERP) software development project before it decided to shut the project down in 2012, as the software failed to deliver 'any significant military capability' (Kanaracus 2012). Even worse, this type of story is not an anomaly in the software development industry. A survey of software projects in 2000 found that only 28 per cent succeeded outright, approximately 23 per cent were cancelled prior to delivery, with the remainder being delivered substantially late, over budget, lacking features, or a combination of the three (Standish Group 2001).

It is through innovation and exploration that software engineering can discover ways to improve practices within the domain (Schneiderman 2007; Taylor 2010). The issue with the US Air Force's ERP software was not that it was deficient in a technical sense, it was that it did not fulfil the needs of the users of the software. Too often this is the familiar story with software development, where software is created with completely functional capabilities that aren't suited to fulfil the users' needs. As a dis-

cipline, software engineering is moving towards user-focused, iterative development practices such as those undertaken within agile software development methodologies, in order to mitigate these issues (Stepanek 2005), however there is still ample capacity to improve in this area.

An uncharted area of interest is the use of multidisciplinary approaches in order to improve practices within the discipline of software engineering. Glass et al. (2002) identify that software engineering is generally an inward-focused discipline, where a survey of literature over a five year period found that 98 percent of papers did not use a reference discipline outside of software engineering. Sim et al. (2000) propose that software engineering could benefit as a discipline by harnessing the theories and methods from fields outside of software engineering itself, while McGrath & Uden (2000) propose that other disciplines may benefit through the sharing of advances in knowledge representation that have developed within software engineering. It is this multidisciplinary approach to knowledge sharing that will be investigated throughout this project, in order to identify the feasibility for sharing knowledge across interdisciplinary boundaries, as well as identification of possible practices that may be beneficial for adoption within the discipline of software engineering from an outside discipline.

An identified discipline that software engineering may be able to learn from is mental health research. Firstly, behavioural sciences have been identified as a potential source of knowledge for software engineering (Jeffery 2000; Wells & Harrison 2000; Conradi et al. 2000). Secondly, the emerging practice of Community Based Participatory Research (CBPR), a technique for studying and addressing intractable health and social problems within mental health research (Israel et al. 2008), has been identified as being undertaken in a very similar manner to agile software development, despite a lack of literature identifying a connection between the two sets of practices. Agile software development has also been identified by Bjornson & Dingsoyr (2008) as an area that warrants further research due to its influence on industry software development practice. It is proposed that the two disciplines may be able to benefit from the knowledge held by the other regarding the implementation of these user-based, iterative approaches to solving problems. Questions that can be asked include:

- Are these methodologies similar enough in implementation that they can indeed be compared? And if so,
- At what points in implementation do agile and CBPR practices diverge? Are there correlations between the issues/benefits experienced by each discipline, the extent of user engagement within each discipline and the point of divergence in implementation of these practices?
- What other practices or methodologies within these disciplines may be of benefit to the other?

Software engineering is concerned with the practices surrounding the design, building, testing and maintenance of software, and this project therefore places emphasis

on research surrounding practices within the software engineering process. Hence knowledge sharing regarding the practices undertaken within agile software development and CBPR will be the focus of this thesis.

1.1 Objective

The purpose of this project is to identify the feasibility of sharing knowledge between software engineering and other disciplines for the ultimate purpose of identifying practices that may be generalised and translated across interdisciplinary boundaries. This thesis will investigate the extent to which software engineering may be able to share knowledge with and learn from the discipline of mental health research, and the extent to which modelling techniques are an effective mechanism for the communication of knowledge between these disciplines.

The specific research questions that are addressed in this thesis are:

- What are the main challenges faced in practice when undertaking user-focused, iterative methodologies?
- Can an effective, general model that represents agile and CBPR methodologies be created and used to communicate knowledge?
- Are there correlations between user engagement at particular lifecycle stages and issues experienced when implementing user-based, iterative practices?
- What is the general attitude towards interdisciplinary knowledge sharing in the software engineering and mental health research domains?
- Which practices may be able to be transferred between mental health research and software engineering?

1.2 Project Plan

The primary research component for this project is an exploratory case study using the framework provided by Runeson & Host (2008). This case study investigates the real-world implementation of agile and CBPR practices by way of interviews with domain experts in each area, and compares and contrasts practices within the two disciplines of software engineering and mental health. The case study also investigates attitudes towards interdisciplinary collaboration felt by the domain experts that acted as participants for the study.

A general model that encapsulates the phases within agile and CBPR practices is created using Systems Modelling techniques. This model is used in the interviews as a mechanism for identifying differences in implementation across the two disciplines, as well as identifying the stages in the lifecycle that demonstrate the most and the least amount of user engagement.

The data collected as a result of this case study is analysed and reported using a grounded theory approach to qualitative data analysis. The analysis will be primarily focussed at hypothesis generation and for explanatory building purposes.

It is noted that there was a change in direction for the project plan partway through the project period. A description of the original plan, along with an explanation for the change can be found in Appendix A.

1.3 Contribution

This thesis contributes to the practice of interdisciplinary knowledge sharing through the demonstration of a mechanism which can be used to transfer knowledge across interdisciplinary boundaries. This thesis also contributes to the software engineering body of knowledge by way of addressing the identified inward facing nature of the discipline and providing a demonstration of how practices from external disciplines may be identified for use within the software engineering domain. Contributions are made through:

1. The application of Systems Modelling techniques to agile software development and CBPR practices in order to develop a general model that is applicable to both practices;
2. A presentation of interdisciplinary literature produced within software engineering to date;
3. The implementation of a case study in which the application of the developed model is demonstrated and information regarding knowledge sharing is gathered from experts in the field;
4. The analysis of collected case study data, leading to identification of practices that may be beneficial for adoption across interdisciplinary boundaries; and
5. Formally comparing agile software development to CBPR, literature to date has failed to do so.

1.4 Document Outline

Chapter 2 provides an overview of the multidisciplinary research that has been undertaken within software engineering. The inward facing nature of the software engineering discipline is explored, along with an overview of related methodologies and practices. The inherently complex nature of software is addressed, along with an introduction to Systems Engineering, Systems Thinking and Systems Modelling techniques. It also briefly explores the use of Participatory Research (such as CBPR) in mental health research, along with identifying grounds as to why mental health research could benefit from knowledge sharing with the discipline of software engineering.

Chapter 3 describes the processes undertaken in order to create a general model that represents the phases within agile software development and CBPR practices. Systems modelling was used for this process, with an informal general systems model of the two sets of practices produced. This approach to modelling represents an original method contributed by the author, and a guide is provided describing the steps undertaken in producing the general systems model.

Chapter 4 introduces the methodology for a case study involving interviews with experts in software engineering and mental health research. The purpose of this case study is to identify areas in which software engineering and mental health research may benefit from knowledge sharing activities, with a particular focus on agile software development and CBPR activities. The design of the case study is outlined in this chapter, with details regarding protocols used for collecting, analysing and reporting data. Potential concerns regarding the validity of the case study are also addressed in this chapter.

Chapter 5 presents the findings for the case study, along with an analysis and discussion of results. Hypothetical models are generated based on the data gathered in the case study, along with a comparison of agile software development and CBPR. A discussion regarding the implications for future interdisciplinary work within software engineering is given, along with identified strengths and weaknesses of the case study.

Chapter 6 reviews the contributions made by this project, presents a summary of the major findings for this research project and identified future work that follows from these findings. This chapter contains the concluding remarks for the project, including the implications for future interdisciplinary research within software engineering.

Chapter 2

Background and Related Work

The literature presented in this chapter gives context for this research project as a whole. The general attitude towards interdisciplinary work within software engineering is presented in order to highlight the insular nature of the discipline. This insular position presents an opportunity for researchers within software engineering to investigate the ways in which other disciplines solve problems of a similar nature, in order to explore and discover new ways to approach familiar problems within software engineering. Through new approaches, research can identify which practices are most effective within software engineering, and in turn, software engineering can contribute back to the disciplines from which it is learning from. In order to set the context for how software engineering will learn from and contribute to another discipline within this research project, three sections are presented in this chapter: the attitudes and approaches to interdisciplinary practices within software engineering; current software engineering practice which are both utilised in this project and may be beneficial to other disciplines; and relevant literature from the discipline of mental health.

The first section of this chapter presents evidence regarding the inward-facing nature within software engineering, and a summary of literature for interdisciplinary work performed to date. The attitudes to interdisciplinary work are included to identify that while there has been a historical tendency to be insular within this discipline, there is growing support towards interdisciplinary practices within software engineering. This support has been met by a handful of research projects which have included an interdisciplinary aspect to them, however have had a tendency to be focused towards software engineering research practices with a particular emphasis towards empirical software engineering research. The research in this project is in contrast to these projects as the focus is on software engineering practices rather than software engineering research methods. This is described visually in Figure 2.1, where majority of interdisciplinary research within software engineering is represented by the red arrow, which is for improving research methods within the discipline (which in turn make recommendations to software engineering practices). The research in this project instead sits along the blue arrow, where the focus is on improving software engineering practices themselves through the investigation of interdisciplinary practices. This chapter will explore literature across both of these areas, as the lessons

learned about using interdisciplinary practices for research methods and for development methods are applicable across the whole discipline, despite the difference in context for application.

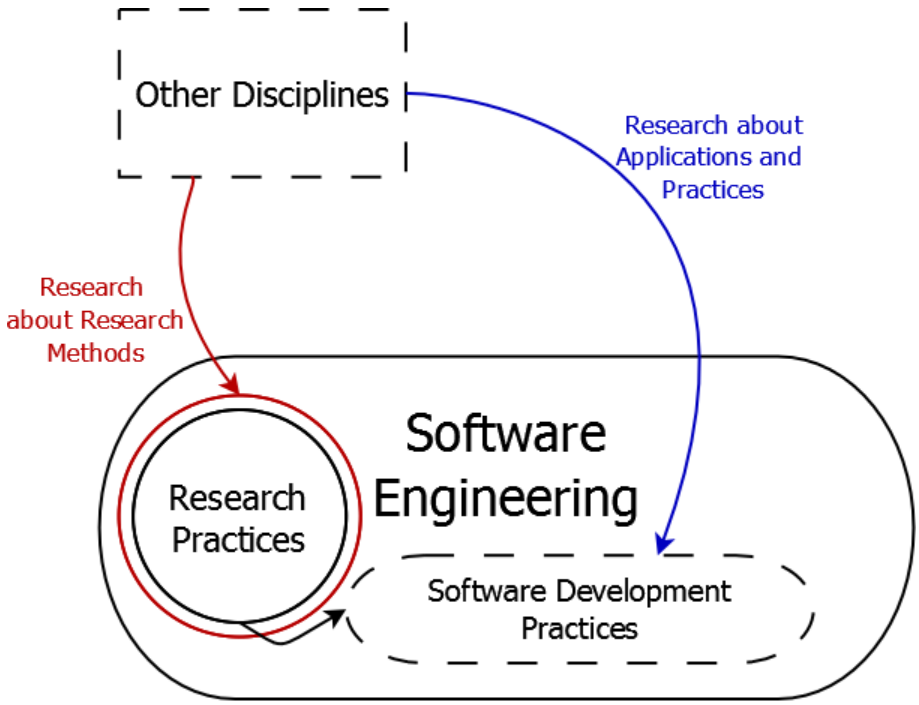


Figure 2.1: Distinguishing between interdisciplinary research towards research and development practices

Following the section on interdisciplinary approaches within software engineering, existing software engineering practices and research are presented. The first reason for including this section is that many of the practices explored through the literature are implemented or used throughout this research project. Practices that are implemented include abstraction, systems thinking and modelling, while practices and research that are used for reference include the software engineering lifecycle, agile software development and the complex nature of software. The second reason for including this section, is that it provides the reader with a reference for some of the practices and research which may benefit other disciplines through interdisciplinary collaboration. While this section may not exhaust all software engineering literature, it provides a core set of research that has been undertaken in the discipline to date. However, in order to present the case for a truly interdisciplinary project, software engineering research alone cannot provide enough context for the project, and as such this section is followed by relevant literature from the domain of mental health.

The final section of this chapter presents evidence supporting the notion that mental health in Australia is in need of reform, where interdisciplinary approaches may prove benefit, and explores the implementation of participatory design and CBPR within mental health research. The need for reform is highlighted as it demonstrates

that mental health is a discipline that may benefit form disruption, and as such, interdisciplinary approaches may provide value to mental health. Current attitudes to interdisciplinary research within mental health research is also explored, highlighting that this discipline is far more outward-looking than software engineering is. Despite having a more outward-focussed attitude, interdisciplinary projects are often restricted to health-based and social science disciplines. This leads to an argument that other disciplines may prove beneficial to mental health, and given their open mindedness, are an appropriate discipline to collaborate with for this project. This section concludes with an outline of participatory design and CBPR methodologies, which bare resemblance to agile software development methodologies. This analysis leads into a comparison and model generation of these multidisciplinary techniques in Chapter 3.

2.1 Interdisciplinary attitudes and approaches to Software Engineering

Research within the discipline of Software Engineering has historically been almost entirely inward facing, relying solely on theories and methodologies that have originated within the discipline itself. A pattern of interdisciplinary work is slowly emerging for research methods, but has not yet gained widespread acceptance within the Software Engineering research community nor appears to have bridged the gap into Software Engineering industry. This section presents evidence of the historically inward facing nature of Software Engineering, along with significant interdisciplinary work that has been undertaken within the discipline in recent years.

2.1.1 Historically inward-looking nature of Software Engineering Research

The majority of Software Engineering research performed to date has not included any reference discipline outside of Software Engineering itself. Evidence of this behaviour is presented within this subsection, along with research that recommends that Software Engineering Research could benefit by increasing the amount of outward-facing research performed within the discipline.

Glass et al. (2002) performed a survey on Software Engineering literature produced within key journals between 1995 and 1999, and of the 369 papers examined, ninety-eight percent contained no reference discipline outside of Software Engineering. Glass et al. explain that the work undertaken in their study was performed with the intention of providing a descriptive report about the state of research within the Software Engineering domain (2002, p. 494). During the examination of 369 papers, a classification scheme containing 12 reference disciplines was used to identify the reference discipline for each paper (Glass et al. 2002, p. 496). In this paper, a reference discipline is defined as "*any discipline outside the SE field that the SE researchers have relied upon for theory and/or concepts*" (Glass et al. 2002, p. 498). Of the papers examined, 0.54% relied on Cognitive Psychology, 0.27% relied upon Social and Behavioural Sci-

ence, Management, and Management Science, while the remaining 98.1% relied solely upon theories and concepts from within the Software Engineering domain itself (Glass et al. 2002, p. 500). This paper concluded that Software Engineering research is predominantly inwardly focused, finding that "*SE research tends to be quite self-contained, not relying on any other disciplines for its thinking*" (Glass et al. 2002, p. 500).

Segal et al. (2005) further expanded upon the work produced by Glass et al. (2002) by investigating literature published within the journal of *Empirical Software Engineering* between 1997 and 2003. Segal et al. felt that the original paper authored by Glass et al. (2002) was narrow in scope and potentially missed a large portion of empirical Software Engineering research, particularly for studies that focus on people. After repeating the work performed by Glass et al. (2002), Segal et al. (2005) found that the original findings largely lined up with what they had found, particularly about the inward facing nature of Software Engineers. Segal et al. note that empirical software engineers share the same tendency to be insular as software engineers in general, finding that 85 percent of papers contain no reference discipline, 6 percent referenced Psychology, 5 percent referenced a Mathematical or Computer Science field and 4 percent referenced Social Science (2005, p. 3). Along with verifying the inward facing nature of the Software Engineering community, Segal et al. also identify that it is unlikely for studies conducted within Software Engineering to focus on people (2005, p. 4).

Leading from the findings made by Glass et al. and Segal et al. there is scope within the discipline of Software Engineering to investigate the theories and concepts developed within other disciplines. The remainder of this subsection will identify research within the Software Engineering domain that provides discussion around the use of interdisciplinary approaches within Software Engineering.

Sjoberg et al. (2007) identified the small extent to which theories are used within the Software Engineering domain, and explore methods that may assist in building theories within Software Engineering. The research identifies that theories in Software Engineering can come from three sources: theories that are used verbatim from other disciplines; theories from other disciplines that have been adapted to Software Engineering before use; and theories that are generated within Software Engineering itself (Sjoberg et al. 2007, p. 14). Sjoberg et al. identify the multidisciplinary nature of Software Engineering, citing theories that are used within the discipline that have been taken as they are from other disciplines, as well as theories from other disciplines that have been adapted for use within Software Engineering (2007, p. 14). Sjoberg et al. (2007) recommend the continued use of developing theories using multidisciplinary techniques, along with developing theories from within Software Engineering as the discipline develops. Recommendations such as these provide momentum for the type of work that is undertaken in this project, from which theories may be developed as a result of multidisciplinary collaborations.

Rus et al. (2001) identify the benefits of using interdisciplinary groups and approaches during their investigation into the use of knowledge management (KM) in Software Engineering. This investigation identified nine success factors for the implementation of a KM within Software Engineering. One of the success factors identified for successful knowledge management was the use of "*Interdisciplinary problem solv-*

ing working groups...[As] this will transfer the knowledge from one discipline to another, as well as provide solutions to interdisciplinary problems in decreased time" (Rus et al. 2001, p. 41). Just as Rus et al. (2001) recognise the importance of interdisciplinary work groups within Software Engineering, so too does this project aim to capitalise on the transfer of knowledge between disciplines, as well as look at potential solutions for interdisciplinary problems.

Herbsleb & Mockus (2003) acknowledge the appeal of using interdisciplinary approaches within Software Engineering for problems that have similar properties. The example provided by Herbsleb & Mockus describes the problems faced by humans competing for floor space and programs competing for memory, which both have similar constraints and are instances of resource conflict (2003, p. 139). Herbsleb & Mockus ultimately decide against using an interdisciplinary approach to the coordination problems they were facing, stating that it is their belief that Software Engineering is not yet sufficiently mature for such broad and general approaches (2003, p. 139). Little detail was provided by the authors of this paper regarding this position towards interdisciplinary practices. The work performed within this thesis acknowledges the infancy of interdisciplinary approaches within Software Engineering, and through an exploratory, empirical case study, aims to identify potential areas within the discipline that may benefit from such approaches.

Bennet et al. (2000) recognise the inherent interdisciplinary nature of software and Software Engineering. Bennet et al. held a software workshop, *Forecasting the Future*, in which academics from a range of disciplines (including organisational sociology, psychology, law, marketing, engineering and biochemistry) were invited to evaluate the work undertaken by the group (2000, p. 3). The results of this workshop were that the issue of interdisciplinarity is critical to developing a vision for the future of software, and that software does not exist in isolation but in a legal, political, social and economic context (Bennet et al. 2000 p. 3). Bennet et al. claim that in order to achieve the highest possible levels of productivity and quality, Software Engineering must break down the rigid walls that exists between software and its environment (2000 p. 3). This thesis aims at breaking down walls between Software Engineering and other disciplines, ultimately for the purposes of collaboration and learning. Bennet et al. (2000) have identified that software cannot exist in isolation, and it is with this mindset that we look to other disciplines for learning opportunities in order to better practices within our own domain.

As demonstrated by Glass et al. (2002), the majority of Software Engineering research provides no reference discipline outside of Software Engineering itself, leading to the conclusion that as a discipline, Software Engineering is generally inward facing. Sjoberg et al. (2007), Rus et al. (2001), Herbsleb & Mockus (2003) and Bennet et al. (2000) recognise that there are certainly various parts of Software Engineering that could benefit from an interdisciplinary approach. It is noted that Herbsleb & Mockus (2003) felt that as it currently stood, Software Engineering was not yet mature enough to adopt a multidisciplinary approach to the coordination activities they were undertaking, however in comparison, the aim of this exploratory project is to identify potential areas in which interdisciplinary practices may be of benefit to Software Engineers.

The remainder of this section will investigate theoretical and empirical research into the use of interdisciplinary approaches within Software Engineering.

2.1.2 Interdisciplinary work to date

Despite having a historical tendency to rely mainly on theories and methodologies that have originated within Software Engineering, there is an emerging trend of interdisciplinary work being produced within the Software Engineering domain. A notable contribution comes from the interdisciplinary workshop *Beg, Borrow or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research* (Sim et al. 2001) which gathered experts to discuss the applicability of using approaches from other disciplines within the Software Engineering Research domain. This subsection introduces both theoretical and empirical research which has been performed, investigating the use of interdisciplinary techniques within Software Engineering, and positions the work contributed within this thesis amongst these works. Figure 2.2 provides a graphical description of the work outlined within this subsection. As per the convention above, red arrows represent research work that focuses on research methods, while blue arrows represent research work that focuses on practices. This figure is provided to give the reader a complete picture of interdisciplinary work conducted within software engineering to date. For a version of the figure with authors and year labels attached to each arrow, see Appendix B.

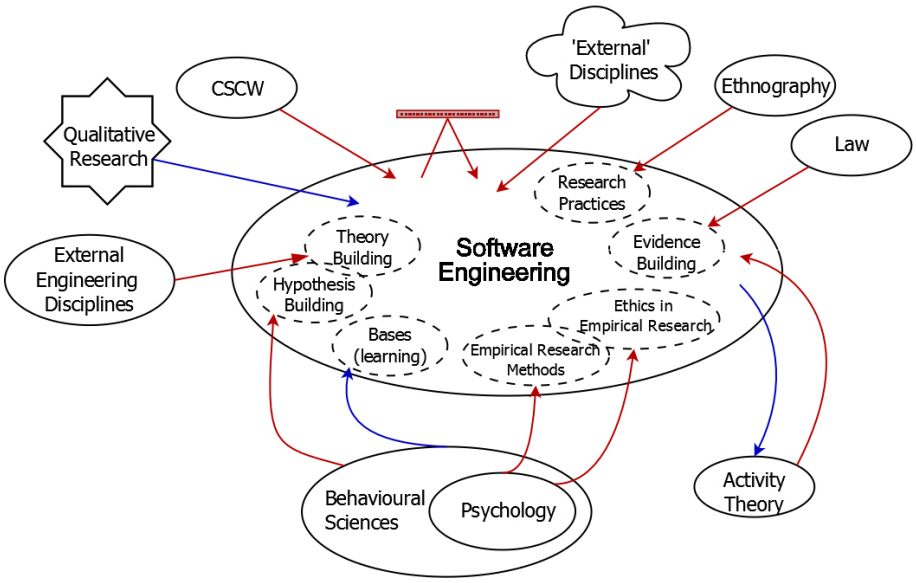


Figure 2.2: Model of interdisciplinary work undertaken in Software Engineering

Sim et al. (2001) facilitated a workshop that provided a forum for software engineers and researchers to discuss the feasibility of adopting methods from other disciplines for the purposes of Software Engineering research. Sim et al. argue that, just as we have learned to re-use code within software engineering, so too can we harness

theories and methods from other disciplines in order to further Software Engineering research (2001, p. 85). Examples of disciplines that Software Engineering may be able to learn off are given, including Anthropology for conducting ethnographies and field studies, as well as Psychology for setting up experiments, analysis of data and reporting of results. (Sim et al. 2001, p. 86). Sim et al. acknowledge that methods and practices from other disciplines cannot be applied wholesale, and must be adapted in some way to suit the needs of the Software Engineers applying them (2001, p. 86). The need to be educated about the nature of interdisciplinary approaches was highlighted, addressing the point that "*[Software Engineers] need to ask "why use a particular technique" not just "how can a technique be applied*" (Sim et al. 2001, p. 87), and that care must be taken to ensure that methods and techniques are nor applied out of context (Sim et al. 2001, p. 88). Conclusions from the workshop include the identified gap in Software Engineering education regarding empirical methods in Software Engineering, despite a need for application of these methods within many facets of the domain (Sim et al. 2001). With regard to the appropriate application of interdisciplinary and multidisciplinary techniques, it was proposed that collaboration can occur across and between disciplines, identifying that this process need not be constrained to just two fields, however when borrowing techniques and practices, it must be done responsibly and respectfully (Sim et al. 2001). Furthermore, when performing research into the adoption of interdisciplinary practices, the research must be of a high enough quality that contributions from Software Engineering to the domain in which the borrowed methods originate from is possible (Sim et al. 2001). This thesis accepts the recommendations made by Sim et al. (2001), in both their attitudes to adopting interdisciplinary and multidisciplinary approaches, as well as contributing back to the discipline in which the practices are being borrowed from. Of the following literature presented in this sections, several of these papers originated as position papers from the workshop Sim et al. (2001), along with further interdisciplinary work that has been undertaken within Software Engineering since.

McGrath & Uden (2000) present a case for the interdisciplinary applicability of knowledge representation methods used in Software Engineering, with particular reference to management science fields. While acknowledgement is made of the rich variety of relevant social science and management material that would benefit Software Engineers, the focus of the paper is on advocacy for the use of formal conceptual modelling techniques for representing models and theories within social sciences, and in particular, management science fields (McGrath & Uden 2000). The use of formal modelling techniques is deemed appropriate due to the clarity in chains of resonating and simplification of task verification that formal modelling provides, along with the clarification of concept overlaps, inconsistencies and ambiguities (McGrath & Uden 2000). McGrath & Uden conclude by addressing the appropriateness of interdisciplinary approaches within software development, and the need to address 'soft' factors when undertaking said development,

> *"In essence, systems are not built in a vacuum, but with organizational environments where outcomes are heavily influenced by a myriad variety of internal and external softer factors"*
>
> (McGrath & Uden 2000, p. 5)

The work presented by McGrath & Uden (2000) lays the foundation for case studies such as the one presented in this thesis, which uses both modelling techniques, along with an objective for providing back to the discipline with which we are collaborating.

Jeffery (2000) presents a report for work regarding the use of theories in disciplines such as behavioural sciences for the purposes of hypothesis formulation in Software Engineering. The author presents the case that developing collaborative approaches with external disciplines will assist in expanding the body of knowledge in Software Engineering, basing this assertion off experience on a developed program of empirical research that was based on behavioural theory (Jeffery 2000). Jeffery makes particular mention of the use of theories and models, however highlights the fact that researchers must be careful to query the appropriateness of each possible research technique over the others.

Along with Jeffery (2000), Conradi et al. (2000) also explore the broad use of behavioural sciences within the discipline of Software Engineering (presented below), while Harrison (2000) and Wells & Harrison (2000) explore a narrower field within behavioural sciences in the form of Psychology methods and practices. Harrison (2000) presents the case that studying large groups of subjects in Software Engineering (as is the traditional technique for empirical research) is not always appropriate. Instead, Harrison (2000) proposes the adoption of the single-subject experiment from Psychology, where factors of interest are alternatively introduced and withdrawn from a single subject over the course of the project. Challenges to this approach are addressed, however the benefits of this approach include overcoming issues with having an unrealistic environment, problems in 'task realism, inappropriate grouping of subjects and an over-reliance on students (Harrison, 2000). Wells & Harrison (2000) explore the idea that the entire software design and development process is affected by the impact of human, organisational and social issues, and as such collaborations with external disciplines (such as Psychology) is emerging as a necessity for fully understanding issues that do not traditionally fall within Software Engineering practices. As Wells & Harrison point out,

> *"In today's world of design and development of systems and software, not only are software engineers expected to provide state-of-the-art technology, but also to simultaneously tackle the inherent issues concerned with human cognitive and social behaviours that will affect the ultimate behaviour of the technology"*
>
> (Wells & Harrison 2000, p. 1)

Wells & Harrison (2000) also acknowledge the increasing complexity surrounding Software Engineering, identifying that the usefulness of a tool is dependent upon the

understanding of the context in which it is used, as well as how it will be perceived by the users. As research within social sciences and Psychology focus largely on human behaviour, they are therefore more familiar with the investigation of human ability (Wells & Harrison 2000). Methods and techniques from these disciplines may be of benefit to Software Engineers in terms of gathering data, as well as ensuring reliable and valid results and conclusions (Wells & Harrison 2000). Wells & Harrison also identify issues with communication across disciplines, highlighting the fact that there is the potential for confusion of terms between (and possibly within) disciplines (Wells & Harrison 2000, p. 4). These papers and findings are of particular interest for this project, due to the closely tied nature of Psychology and Mental Health Research.

Qualitative Research Methods have been gaining momentum in support throughout the Software Engineering research community (Sim et al. 2001; Wells & Harrison 2000; Runeson & Host 2008; Dittrich et al. 2009), and this survey notes the work of Becker-Kornstaedt (2000) and Conradi et al. (2000) in this area, with respect to the application of interdisciplinary techniques within Software Engineering. Becker-Kornstaedt (2000) discusses the existing of use of qualitative research methods within areas such as Requirements Elicitation, and proposes an exploration of qualitative techniques for their use in the elicitation of software process knowledge. Like Jeffery, Conradi et al. (2000) investigated the applicability of using qualitative methods from behavioural sciences, however the focus was directed towards Software Experience Bases (that is, mechanisms for software developers to share experiences with one another). Conradi et al. (2000) explore the theoretical possibility of using organisational theory and social anthropology from behavioural science in order to observe, understand and assess work patterns as well as for assisting with integration of new technology into existing organisations. Both Becker-Kornstaedt and Conradi et al. emphasise the use of qualitative research methods, which provides validation for the qualitative methods used through this project.

In the position paper *Borrow Fundamentals from Other Science and Engineering Disciplines* (Xia 2000), Xia argues that Software Engineering research must tend towards the same methodological rigor that is observed in other science and engineering disciplines. The historically theoretical-lacking foundation of Software Engineering is explored, followed by the recommendation that the Software Engineering community needs to rigorously define technical concepts, research hypotheses and testing and maintenance theories (Xia 2000). Xia's ultimate position is that Software Engineering can leverage off the theory building techniques that are present in other science and engineering disciplines (Xia 2000).

Menezes & Pfleeger (2000) investigate the way in which the discipline of Law treats evidence, identifying practices and ideas that are useful to Software Engineering researchers and practitioners. The rules used by the legal community to evaluate and combine evidence are proposed as potential frameworks for empirical research in Software Engineering, with particular emphasis on evaluation of results, design studies for the generation of new results and evaluation of bodies of knowledge, in the form of collections of results (Menezes & Pfleeger 2000). Menezes & Pfleeger (2000) investigate the feasibility of using these rules and potential benefits from undertaking

these practices. While this project is not specifically looking at the use of Law practices within Software Engineering, the work undertaken by Menezes & Pfleeger provides a good foundation for investigating the applicability of interdisciplinary practices within Software Engineering.

Traditional Ethnographic studies involve the investigation of cultures, including their customs, habits, and mutual differences (Ronkko et al. 2002). While an adaption of Ethnography is currently performed within the discipline of Software Engineering (Sim et al. 2001), Ronkko et al. (2002) propose that Software Engineering may be able to learn from the traditional implementation of Ethnographic practices, as from their original sociological roots. Specifically, as it currently stands, the quantification of ethnographic field studies undertaken within Software Engineering (in order to present results that are in agreement with traditional Software Engineering research standards) implies the loss of an important contribution made by Ethnographic studies, that is the inside perspective on how practices are performed (Ronkko et al. 2002). Ronkko et al. argue that by adopting a qualitative ethnographic approach, Software Engineers can investigate how methods are actually being performed in various industrial contexts, compared to the researcher's initial aims when creating the methods in the first place (2002, p. 6). The case study undertaken within this project investigates cultural issues addressed when using user-driven, iterative practices and analyses the results using qualitative methods. In this way, many of the recommendations made by Ronkko et al. (2002) are adopted within this project.

Computer Supportive Cooperative Work (CSCW) is the qualitative research technique concerned with the ways in which humans interact with technical artefacts (Dittrich et al. 2009). Historically, CSCW has had little interaction with the process driven discipline of Software Engineering, despite the fact that both disciplines are concerned with the way in which users interact with software (Dittrich et al. 2009). Dittrich et al. explain that as the number of failed software projects began to rise, it was identified that the complexity of software problems were not always solvable with strict linear processes, and thus flexible and End-used development approaches emerged (2009, p. 394). While collaborations between CSCW and Software Engineering have been emerging over recent years, Dittrich et al. identify the possibility further collaborations between the disciplines, and leveraging off the techniques based in each discipline in order to fully utilise the rich data produced by qualitative methods. The case study undertaken within this project utilises qualitative research methods, with analysis on the rich data set collected by undertaking this case study included in the reporting section of this thesis.

2.2 Software Engineering Practices and Research

Within the discipline of Software Engineering there exists a rich set of laws, practices and methodologies that are used for the development, testing and maintenance of software. While there is an identified communication gap between Software Engineering research and industry (Parnas 1995; Brown & McDermid 2007; Menezes

& Pfleeger 2000; Beckman et al. 1997), this section aims to present practices and research that have a historical basis in both domains. As identified by McGrath & Uden (2000), interdisciplinary work undertaken within the discipline of Software Engineering should aim to contribute back to any discipline with which it collaborates, and the work presented in this section identifies a set of laws, practices and methodologies which could be contributed by Software Engineering. This section also presents research on the inherent complex nature of developing software, as well as tools for handling this complexity such as abstraction, modelling and Systems Thinking. Research regarding the complexity of software is included in order to demonstrate the applicability of practices within Software Engineering towards complex problems. This work is included to help demonstrate areas in which interdisciplinary practices may be appropriate for transfer, that is, with other disciplines that also deal with complex problems.

2.2.1 Laws of Software Engineering

The discipline of Software Engineering contains a set of accepted laws which have been derived empirically from observation. Endres & Rombach explain that Software Engineering laws are derived when reoccurring observations occur, where laws are generalisations across situations or time periods (2003, p. 2). A key characteristic of Software Engineering laws are that they explain *how* things occur, without an explanation of *why* they occur (Endres & Rombach 2003, p. 2). In Boehm's 1984 article *Software Engineering Economics*, he observed the importance of the laws governing software development, noting that the laws have received general quantitative support however further research was needed to clarify the area at the time (Boehm 1984, p. 40). Much like the discipline of Software Engineering itself, the laws have grown and evolved over time, as demonstrated through the remainder of this section.

Initially eight laws existed to describe the Software Evolution process. These laws are fully outlined in Lehman's *Laws of Software Evolution Revisited*, having been collected between 1968 and 1988 (Lehman 1996, p. 108). Many of these laws (such as the law of Increasing Complexity, the law of Continuing Change and the law of Feedback System) carry on to be considered standalone laws today. Others have been decomposed, or modified, based on further examination and observation.

Over time, the number of accepted Software Engineering laws has grown to at least fifty, and continues to grow as observations are generalised. In Endres & Rombach's *A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories*, fifty laws which are generally agreed upon by experts are presented, with acknowledgement that this list is not considered a complete list (Endres & Rombach 2003, p. 5). For the sake of brevity the entire set of laws will not be reproduced in this thesis, and instead only laws which directly affect each section will be reproduced where appropriate. The laws presented by Endres & Rombach are categorised by author and deal with topics such as software-related complexity, gathering requirements, software quality, system architecture and other general topic areas. While these laws are generally accepted by experts, historically there has been some opposition to

the Software Engineering laws.

Criticism of the term *laws* have historically been prevalent in the Software Engineering community. Lehman addresses several of these criticisms, including that the observed phenomenon reflected human behaviour and not laws, that the behaviour could be tied to particular organisations or technologies and the lack of statistical significance of the laws, the latter of which is presented in the conference paper *An Examination of Evolution Dynamics* (Lawrence 1982). Lehman refutes these criticisms, highlighting that these laws are the product of many individuals, with analysis of the laws being conducted by disciplines outside of computer science. Furthermore, the criticised behaviour was that observed within the environment in which software engineering operates, highlighting the validity of these observations. Finally, the laws do not claim to be derived from statistical models rather that the inputs for the laws come from the reality of software development. (Lehman, 1996, p. 110) Endres & Rombach further this argument, by stating that they "*mean law in the sense of 'law of nature' [...] an order or relation of phenomena that, so far as is known, is constant under certain conditions*" (Endres & Rombach 2003, p. 4).

While criticism of the laws has been taken under consideration, it is the position of this project that the laws of Software Engineering, particularly those presented by Endres & Rombach, will be generally accepted in the same way that they have been accepted by experts in the field. The laws of Software Engineering will provide a basis for the Software Engineering components throughout this project, as well as, where appropriate, offer potential benefit to disciplines outside of Software Engineering.

2.2.2 Software Engineering Practices and Methodologies

The discipline of Software Engineering is comprised of several practices, which operate together to produce and maintain quality software. The *Guide to the Software Engineering Body of Knowledge – SWEBOK V3.0* (SWEBOK) is an international standard (ISO) for Software Engineering practices, produced by the Institute of Electrical and Electronics Engineers (IEEE) which specifies these practices (Bourque & Fairley, 2013). While several sub practices are specified in the guide, there are five broad categories within which these sub practices sit. They are Software Requirements, Software Design, Software Construction, Software Testing and Software Maintenance (Bourque & Fairley, 2013). While other practices exist concerning Software Engineering (such as Project and Quality Management), the above mentioned categories represent the practices undertaken during the Software Development lifecycle. Currently there are no works as comprehensive as SWEBOK for capturing the practices undertaken within Software Engineering, and thus work within this research project will leverage off the work produced in the guide.

Software Engineering Development Methodologies can be broken down into three categories, Incremental, iterative and agile. SWEBOK outlines these methodologies as follows: Incremental models can be considered *"linear models in which the phases of software development are accomplished sequentially with feedback and iteration"* (Bourque & Fairley 2013, p. 8-5), also known as waterfall; iterative models are ones *"in which*

software is developed in increments of increasing functionality on iterative cycles" (Bourque & Fairley 2013, p. 8-5); and *"agile models that typically involve frequent demonstrations of working software to a customer or user representative who directs development of the software in short iterative cycles"* (Bourque & Fairley 2013, p. 8-5). As mentioned in the *Software Engineering Practices* subsection, SWEBOK is considered the international industry standard, and as such, these definitions will be used throughout this research project, unless otherwise stated.

A different approach to Software Engineering methodologies comes from Stepanek's *Software Project Secrets: Why Software Projects Fail*. Stepanek investigates traditional Project Management methodologies that have been used for software development which have been shown to yield a success rate of 28% (Stepanek 2005, p. 3), and different approaches to a strictly linear style (such as agile) can be used to increase the chances of success in a project. This literature critically analyses different Software Development methodologies, and investigates which practices within the methodologies are more successful and why. Stepanek's work addresses why particular methodologies are able to tackle the complexity associated with software better than others, and as such, provides part of the foundation for *generalising* the various practices and laws presented in the following section. In this way, this research project will build on the work of Stepanek. As this project investigates the comparison of agile development methodologies compared to practices within Mental Health Research, further background into these methodologies will be explored.

2.2.3 Agile Development

Agile Development Methods are a set of iterative and incremental development (IID) methodologies that advocate the use of iterations throughout the software development lifecycle. Agile methods differ from the traditional single-pass "requirements, design, implementation, testing" waterfall lifecycle approach due to two key factors. Firstly, Agile methods specify that iterations of development occur throughout the software project's lifespan, incrementally evolving the software through each iteration (Larman & Basili 2003). Secondly, Agile methods advocate for customer and client involvement throughout the development process, not simply for requirements elicitation and evaluation (Larman & Basili 2003). A graphical model of the general Agile Development Process can be seen in Figure 3.8.

Agile Development Methods were formally introduced by a group of 17 process experts through the *Manifesto for Agile Software Development* in 2001 (Fowler & Highsmith 2001), though IID development techniques had been used throughout the Software Engineering industry since as early as the 1960s (Larman & Basili 2003). Before Agile, iterative development practices had received support from Software Engineering thought leaders such as Glass' support of incremental development practices (Glass 1969) and Boehm's Spiral Model of Software Development (Boehm 1986). Larman & Basili identify an increased public awareness in IID methodologies, with several recommendations made by industry representatives to move towards adopting evolutionary development methods (2003, p. 53-54).

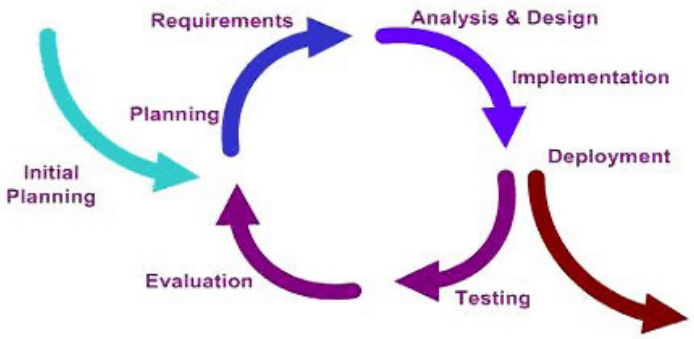


Figure 2.3: General Agile Development Process

Since the conception and release of Agile, the Software Engineering industry has adopted Agile Software Development as the most widely used software development lifecycle approach (Heusser 2013). A 2012 survey of over 4000 individuals in the Software Engineering industry found that 84 percent of respondents believed that their organisation practiced agile development (VersionOne 2012). The widespread adoption of Agile may be related to the identified benefits of Agile, which include the ability to work effectively in radically different environments, increased developer satisfaction with their jobs and the product, better engagement with the client and reports of increased code quality when agile methods are used (Dyba T & Dingsoyr 2008). The widespread use of Agile may also derive from the principles of agile, whose purpose is to identify better ways of developing software (Fowler & Highsmith 2001).

The principles of agile specify that the highest priority for any project is to satisfy the customer, allowing for changing requirements to develop throughout the course of the project as a result of frequently delivered software (Fowler & Highsmith 2001). The values of Agile, as specified in *Manifesto for Agile Software Development*, are

> *"Individuals and interactions over processes and tools.*
>
> *Working software over comprehensive documentation.*
>
> *Customer collaboration over contract negotiation.*
>
> *Responding to change over following a plan."*
>
> (Fowler & Highsmith 2001, p. 2)

Each statement is given as a preference; although the part on the right is important, the part on the left is deemed *more* important by the authors. These values underpin the Agile Development Process, and provide the context within which agile methodologies are expected to operate in implementation. It should be noted that while Agile has been widely adopted across the Software Engineering industry, there still exist issues within this software development lifecycle approach.

Identified challenges with Agile Development methods include difficulties in knowledge representation and transfer, disruption for traditional management processes, increased pressure for customers of such projects and the limitations placed on team dynamics. The preference for working software over documentation has been identified as a challenge for knowledge management within organisations, as documentation typically serve as useful artefacts to facilitate the transfer of knowledge between developers (Nerur et al. 2005). This is particularly troublesome for teams implementing Agile methodologies as IIDs are typically more complex to understand that traditional development methodologies (Larman & Basili 2003). Additionally, by increasing the tacit knowledge that resides in the heads of developers, the potential to make the organisation heavily dependent on the development team increases, shifting the balance of power within an organisation from management to development teams (Nerur et al. 2005). This also leads to developers being less interchangeable in agile teams (Dyba & Dingsoyr 2008), all of which contributes to a disruption to traditional management practices. Finally, the engagement of customers in an Agile project has been identified as a critical success factor for these projects (Nerur et al. 2005), yet it is known that the role of the on-site customer can be stressful, and is ultimately unsustainable for long periods of time (Dyba & Dingsoyr 2008). The identification of these issues within Agile demonstrate that Agile is not perfect, and this thesis makes no such claims. Instead, this thesis is interested in Agile due to its current dominance within the Software Engineering industry, and the identified benefits that may be inherited by undertaking such methodologies.

2.2.4 The inherent Complexity of software

There is consensus among experts that the very nature of software makes it inherently complex. Brooks argues that "*the complexity of software is an essential property, not an accidental one*" (Brooks 1987, p. 11). Endres & Rombach introduce *A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories* with the notion that Software Engineering is a subset of the discipline of Systems Engineering, the discipline that "*deals with the planning, development, and administration of complex systems*" (Endres & Rombach 2003, p. 1), where Software Engineering deals specifically with software. Brooks explains that "*Software entities are more complex for their size than perhaps any other human construct because no two parts are alike*" (Brooks 1987, p. 11), highlighting one of the key attributes of software that makes it inherently complex. Stepanek explains that when software increases in size (by lines of code), the complexity of the code increases by the *square* of the number of lines of code added (due to the communication between components), making software unique in that complexity is its most significant issue (Stepanek 2005, p. 9-10). There is little to no literature that disputes the complex nature of software.

In Booch's *Object-Oriented Analysis and Design with Applications*, the inherent complex nature of software is further explored with identification of four key elements; the complexity of the problem domain; difficulties of managing the development process; the affordances of flexibility possible through software; and problems with character-

ising the behaviour of discrete systems (Booch 1994). Booch explains that complexity in projects is further exacerbated because along with the tendency of software requirements changing during its development, the very presence of the software development project alters the rules of the problem domain (Booch 1994, p. 9). The work performed by Booch reinforces the work above done by Brooks, Stepanek, Endres and Rombach in highlighting the complex nature of software. The characteristics that are highlighted by the above authors will be used throughout this project to draw comparisons between software and other complex systems. In this way, the project will be leveraging off the work previously performed investigating the complexities surrounding Software Engineering, in the pursuit of other, comparably complex disciplines to collaborate with.

2.2.5 Object-Oriented Design, Abstraction and Modelling

Object-Oriented design lends itself to directly mitigate part of the complexity that is inherent in software. In Booch's *Object-Oriented Analysis and Design with Applications*, object-oriented design is shown to be effective in creating software resilient to change. Object-oriented design also increases the level of confidence in the software being built, and reduces the inherent risks of developing complex software systems (Booch 1994, p. 27). A key characteristic of complex systems is that they are comprised of hierarchies, whereby the system is "*composed of interrelated subsystems that in turn have their own subsystems, and so on*" (Booch 1994, p. 12). These hierarchies lend themselves to two distinct types: ones in which objects are a *part* of a bigger system, and ones in which objects are a *type* of a broader system. These two types of systems can be seen in below in Figure 2.4.

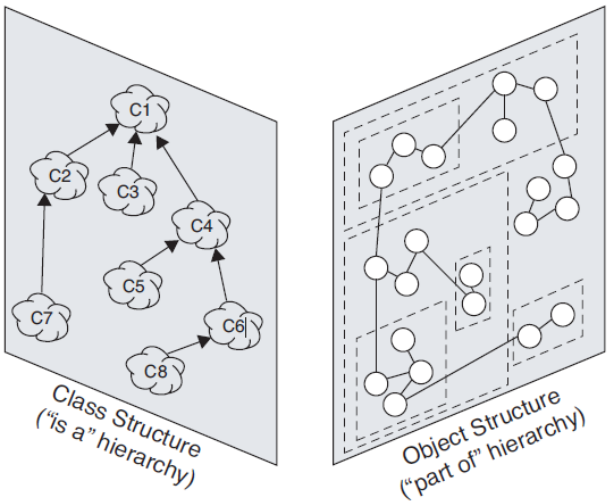


Figure 2.4: Booch's Key Hierarchies of Complex Systems (Booch 1994, p. 16)

Object-oriented design therefore lends itself to decomposition (decomposing a complex software system into smaller parts, each of which can then be refined inde-

pendently (Booch 1994, p. 19)), and can be used to create Class and Object Hierarchies of the system. A complex system can be viewed as a hierarchy of inter-relating components. Hierarchies remove some of the intricate details of the system, and instead focus on the relationships between components which make it easier to understand the system. The concept of removing details to make elements easier to understand is known as abstraction, and is one of the key principles on which object-oriented design is founded.

Abstraction, the act of removing detail through generalisation, is a key tool at a Software Engineer's disposal to reduce the complexity of a software system. Dahl et al. describe abstraction as the process of identifying and focussing on similarities between objects, processes or situations in the real world (Dahl et al. 1972, p. 83). Booch expands upon this description, stating that abstraction

> "*denotes the essential characteristics of an object that distinguish it from all other kinds of objects and thus crisply defined conceptual boundaries, relative to the perspective of the viewer*" (Booch 1994, p. 41).

Information hiding is act of 'hiding' the information and data inside an object. Abstraction utilises this concept by often describing the interface and behaviour of an object without detailing its inner workings. In this way, abstraction is able to 'hide' the inner (or irrelevant) complexities of an object, and allow the viewer to focus on the aspects of the object that matter to them. By applying abstraction techniques to software, and object-oriented hierarchies, Software Engineers are able to sanction different levels of complexity to make them more manageable. Booch identifies that "*the task of the software development team is to engineer the illusion of simplicity*" (Booch 1994, p. 9). This technique is illustrated in this literature review, as it is not a tool reserved for software engineering. Indeed any discipline needing to deal with complexity could utilise abstraction, and as such this will be a recurring theme throughout this project. Along with abstraction, modelling is also a powerful tool for Software Engineers to deal with complexity.

Models are powerful tools used in Software Engineering because they incorporate abstraction, decomposition and hierarchical structures, and allow us to test functions of software systems in controlled environments. Endres & Rombach explain that the purpose of models are to "*explain our understanding [...] They are indirect views or abstractions, leaving out those things that are not considered important for the time being*" (Endres & Rombach 2003, p. 24). Booch explains that the reason models have a broad acceptance across all engineering disciplines is partly due to the fact that they appeal to the principle of decomposition, abstraction and hierarchy (Booch 1994, p. 26). While models are a useful tool, one of the laws of Software Engineering is Davis' Law (L4):

> "*The value of a model depends on the view taken, but none is best for all purposes*" (Endres & Rombach 2003, p. 22)

Booch further emphasises this point by noting that "*in order to express all the subtleties of a complex system, we must use more than one kind of model*" (Booch 1994, p. 26).

The notations used for modelling in Software Engineering has been standardised, due to the work of the Booch, Rumbaugh & Jacobson in *Unified Modeling Language (UML) 1.0* (1997). Standardising the notation in models allows analysts and developers to unambiguously describe design decisions to one another, as well as remove unnecessary work done to understand the semantics of a model and instead concentrate on the content (Booch 1994, p. 172). Models will therefore be used as a tool throughout this project to help articulate ideas and mental models, as well abstract complex systems in order to remove some of the difficulties in understanding the system. One type of modelling that will be focussed on in particular is Systems modelling, which are used to describe causal relationships within a system.

2.2.6 Systems Thinking and Systems Modelling

Systems thinking and systems modelling has been introduced into this thesis for two key purposes. Firstly, in order to undertake a case study across disciplines, a technique for representing processes, systems and relationships is important to avoid ambiguity (as much as possible) when communicating across several disciplines. Systems modelling has been designed for interdisciplinary communication purposes, and is therefore a fitting tool for this project. Secondly, the use systems thinking and systems modelling may prove beneficial to disciplines external to software engineering who are not currently using them, and the promotion of these methods is therefore important.

Systems thinking and systems modelling are techniques used throughout Systems Engineering in order to explicitly express a person's understanding of a system (Aronson 1998). As a subset of systems engineering, these techniques are equally applicable across the discipline of software engineering (Endres & Rombach 2003). The key function of systems thinking lies in the ability to expand one's thinking of a system in order to take into account larger numbers of interactions that exist both inside the system and between the system and other systems (Aronson 1998). This is in contrast to traditional form of analysis which specifies breaking a system into parts and studying those parts in isolation. Systems modelling is a manifestation of systems thinking, where the mental models of a system are abstracted and captured in a diagrammatic form.

Systems modelling techniques were first introduced by Richardson & Pugh's *Introduction to system dynamics modeling with DYNAMO* (1981) and made popular by Senge's highly cited *The fifth discipline: The art and practice of the learning organization* (1990). Causal relationships between variables within a system are modelled through the use of arrows with noted polarities. Plus ('+') and minus ('-') signs indicate polarity, where a plus sign represents that an increase in the independent variable causes the dependent variable to also increase (that is, a proportional relationship), whereas a minus sign indicates that an increase in the independent variable causes the dependent variable to decrease, thus an inversely proportional relationship (Senge & Sterman 1992). An example of this can be seen in Figure 2.5. This figure represents a savings bank account, where all blue arrows represent proportional relationships (for

example, increasing the frequency or amount of deposits leads to an increase in total principal) and the red arrow represents an inversely proportional relationship (that is, an increase in withdrawals will cause a decrease in principal). Note the use of red and blue arrows to represent proportional and inversely proportional relationships respectively.

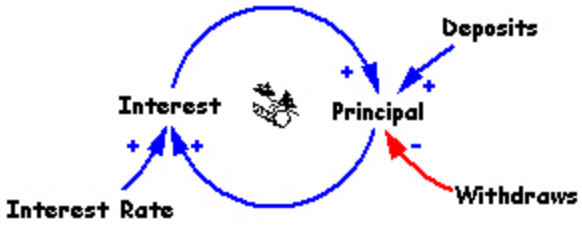


Figure 2.5: Systems model of a savings bank account (Bellinger 2004)

Along with representing arbitrary systems, there are several system archetypes that Senge (1990) identifies that can be modelled with systems modelling. Two archetypes that are of particular interest within this project are the reinforcement feedback and the balancing loop systems. These archetypes are the foundation structures of systems thinking (Bellinger 2004).

The reinforcement feedback (or reinforcement loop) system are systems in which small actions can lead to large consequences due to the compounding and reinforcing nature of the system itself (Senge 1990). An example of a reinforcing feedback system when customers have a positive experience at a retail store, and spread the word about their satisfaction, sending more customers into the store, and thus reinforcing the cycle. A visual example of this type of system is given in Figure 2.6. While the reinforcement feedback system involves some sort of compounding or snowballing effect, the balancing loop instead is constantly in a state of trying to balance opposing variables.

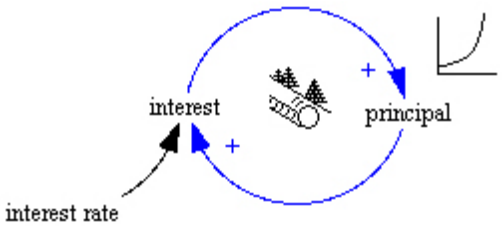


Figure 2.6: Reinforcement feedback system for bank account (Bellinger 2004)

The balancing loop is any system that is concerned with moving from a current state towards some sort of goal state, implicit by the system (Senge 1990). An example of a balancing loop is a hungry human being, who eats until they are full, but then as the body uses energy from the food they have previously eaten, they must eat again in order to continue having energy. Senge notes that all goal-oriented behaviour is underpinned by feedback loops, and can be found within all complex organisms

(Senge 1990). A visual description of this type of system is given in Figure 2.7.

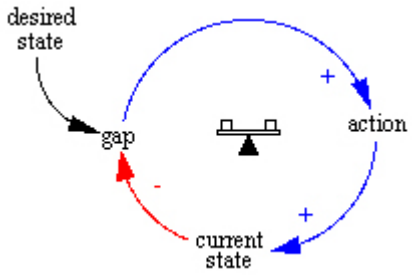


Figure 2.7: Visual description for balancing loop system (Bellinger 2004))

Though they may appear simple, these techniques are powerful ways of representing relationships within a system and can be used across many disciplines (Senge 1990). A systems model to user-focused, iterative software development is presented in Chapter 3, using the techniques and structures described throughout this section. It is the power in simplicity of representation that appeals to the use of systems thinking and systems modelling throughout this project, as they facilitate a reduction in ambiguities within interdisciplinary communication, and may prove beneficial for other disciplines who are currently not utilising these techniques.

2.3 Relevant literature in Mental Health

An interdisciplinary approach to a topic will never be complete if research from only one discipline is considered, and as such a review of relevant mental health literature is presented in this section. This research project does not attempt to produce an extensive report of all literature produced in the discipline of Mental Health, but instead presents two key areas that are appropriate for this project. Firstly, the current state of the discipline is presented, highlighting the fact that reform has been identified as a necessity within the Mental Health domain. Secondly, community based participatory research is explored, giving an overview of the practice, and presenting information that is relevant to this research project.

2.3.1 Current state of Mental Health

Australia's mental health system is at risk of becoming ineffective and unsustainable without intervention. A review of Australia's mental health system from 1983 to 2013 found that despite significant support towards improving mental health across the nation, the life expectancy of people with mental illness has not improved in 30 years, and the number of reports for service issues and failures outweigh reports highlighting positive changes (Mendoza et al. 2013). Mendoza et al. also found that those with mental illness receive inadequate access to health care, and for those that do receive care, it is often inappropriate or ineffective (2013, p. 41). Another review of the state of mental health in Australia found that there is a pressing need to reconceptualise the

nation's mental health system due to the unsustainable nature of increasing health-costs within the system (Hosie 2014). Rosenberg & Hickie note that there appears to be few areas as complex as mental health (2013, p. 1). These issues indicate that as an industry and as a discipline, mental health is in need of disruption. As with software engineering, an interdisciplinary approach to practice may prove beneficial to mental health.

Mental health has historically had a much more outward-focusing nature than software engineering, having a whole branch of health care that is inherently interdisciplinary. The American Psychological Association describes Integrated Health Care as an interdisciplinary approach which is characterised "*by a high degree of collaboration and communication among health professionals…[with] sharing of information among team members related to patient care*" (APA 2014). These health care professionals come from a range of disciplines, including physicians, psychologists, and social workers, making integrated health care a branch of health care that is interdisciplinary by nature. Integrated health care is not the only instance of interdisciplinary work being performed within the discipline of mental health by any stretch. In the 1960s the idea of having an interdisciplinary approach or interdisciplinary team within a mental health context was not new, indeed it was described as an idea that had been around for some time (Auerswald 1968). It is therefore the belief of the researcher of this project that it would be appropriate to undertake an interdisciplinary project with the discipline of mental health research. The identified open-minded nature of mental health, coupled with their need for disruption makes them a suitable discipline with which to undertake collaboration.

2.3.2 Participatory Design and Community Based Participatory Research

Community Based Participatory Research (CBPR) is a methodology within the discipline of mental health that is of particular interest to this project. CBPR has been identified as a methodology that has interdisciplinary applications, and whilst being research focused, puts emphasis on producing outputs that will benefit the community that have participated in the research project (Minkler & Wallerstein 2008). The characteristics of interdisciplinary application and tangible output production, along with the similarities explored in Chapter 3, make CBPR an approach of particular interest to this research project. The core phases undertaken within CBPR can be seen in Figure 2.8.

CBPR is a collaborative approach to research in which participants of the study assist in the design of research questions, as well as being active contributors from the design of the study through to evaluation of results (Minkler & Wallerstein 2008). CBPR is not a technique in itself but rather a methodology to research, encompassing approaches such as action research, participatory action research and mutual inquiry (Cornwall & Jewkes 1995). Participatory design is a framework within CBPR, which aims to integrate the ideas and experiences of people into mental health based interventions (Hagen et al. 2012). Figure 2.9 provides a high level overview of the stages within the participatory design process.

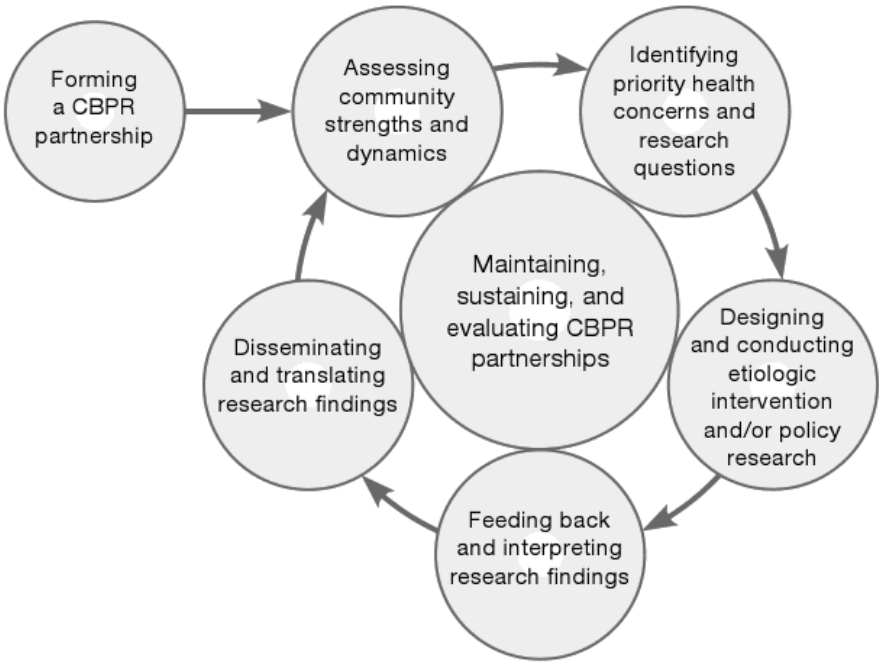


Figure 2.8: Core Phases when conducting CBPR (Israel et al. 2012, p. 11)

The key characteristic of interest in Participatory Design, along with CBPR in general, is the iterative nature of the process. Once an intervention is created, it is evaluated and remaining problems are identified in order to improve the original design. As described above, this process is undertaken with the participants of the research.

The foundations for CBPR date as far back as the 1930s, as described by Faridi et al.

> *"The seminal work [for CBPR] of Kurt Lewin and Paul Freire — to name just two early researchers — dates back to the 1930s and emphasizes an iterative process of action, reflection, and experiential learning."*
>
> (Faridi 2007, p. 1)

Despite such early beginnings, CBPR has not always been accepted by the academic community, and has only received increasing public recognition relatively recently (Minkler & Wallerstein 2008). This increasing trend has been partially attributed to the Institute of Medicine (IOM), which named CBPR as one of eight emerging content areas for which training should be offered by all schools of public health (Gebbie et al. 2003).

CBPR and participatory design will be further investigated throughout this thesis, in contrast to agile software development techniques. Several of the characteristics of CBPR practices bare similarities to agile software development practices, and as such,

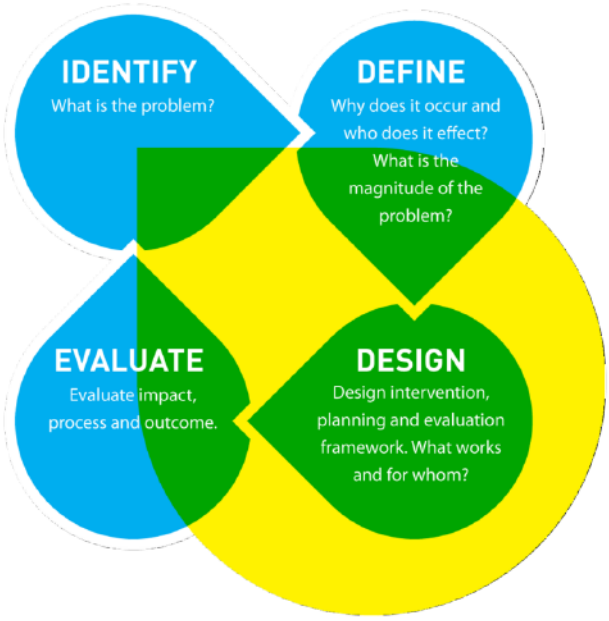


Figure 2.9: Stages within Participatory Design process (Hagen et al. 2012)

a common generic model for encompassing both methodologies is created using the systems modelling technique in Chapter 3.

Chapter 3

The Agile and CBPR Systems Model

This chapter describes the processes undertaken in order to create a general model that represents the phases within agile software development and CBPR practices. Systems modelling was used for this process, with an informal general systems model of the two sets of practices produced. This approach to modelling represents an original method contributed by the author, and a guide is provided describing the steps undertaken in producing the general systems model. The intended applications of this model are described, along with limitations of using this modelling technique.

This project compares and contrasts the practices of agile software development and CBPR due to their observed similar nature. The similarities were identified through observation of design and implementation, as there does not appear to be a link between the practices in literature. This led to the natural question of, how similar are agile and CBPR techniques implemented in practice? And if they did indeed share substantial similarities, would it be possible to create a general model that represented both practices that would allow a comparison of implementations to be made? It was the belief of the researcher that the two disciplines may have been able to benefit from observing how the other implemented the practice differently. This is because such interdisciplinary comparisons have scarcely been made in Software Engineering previously, and thus represented a potential new area from which to learn about effective development practices. An example of a potential benefit could be if agile practices typically performed a particular lifecycle phase differently to the corresponding lifecycle phase in CBPR, then the two disciplines could look at the reasons behind these differences. These differences could be further investigated to determine if they were context specific, or if there is a key reason why one discipline undertakes there practice in a particular way, and if such, could the other discipline investigate the same consideration within their own domain. In this way, learning and collaboration could occur between the disciplines, with a general model acting as a tool to facilitate this. In order to create such a model, a method for generalising and representing the two methodologies needed to be identified.

Models were deemed to be an appropriate tool to facilitate a comparison between the methodologies due to their ability to convey meaning and understanding to the

viewer. As discussed in Chapter 2, models are used throughout software engineering to explain understanding, allowing for abstractions to leave only the most necessary components of the entities that are being modelled (Endres & Rombach 2003). Identified alternatives to using models for the purposes of comparisons included verbal and written descriptions of the methodologies. These methods, whilst valid, were felt that by themselves would allow too much variability in interpretation, as well as reintroduce some of the ambiguity that is addressed through the use of models (Booch 1994). The advantage of using models is that they are designed to articulate ideas about mental models, as well abstract complex systems in order to remove some of the difficulties in understanding the system. One modelling method in particular that was further investigated was the use of Systems Modelling.

Systems modelling was recognised as an appropriate mechanism for creating representations of interdisciplinary practices. There are several advantages to using systems models, including their ability to organise key elements of a system into a structure that can highlight the connections, interactions and relationships between the elements of the system (Bennet et al. 2005). Systems modelling has a historically been used for understanding and representing multidisciplinary processes (Senge, 1990), and the simplicity of the rules necessary for creating and understanding the model were appealing for use across a multidisciplinary audience. The rules for creating a systems model were covered in Chapter 2, and thus will not be repeated here. Other modelling techniques such as UML were also considered, however the time needed to explain the notation and rules to those who do not operate within the software engineering domain was seen as a disadvantage compared to the simplistic rules of systems modelling. While for this activity systems modelling was seen as advantageous over UML, as Davis' Law points out, no type of model is best for all purposes (Endres & Rombach 2003).

By taking a systems perspective to the task of comparing agile and CBPR, emphasis is placed on the interactions between the components of the systems, which in this case, are lifecycle methodologies. Therefore, it is these interactions, these causal relationships, which will be modelled using the systems modelling approach described in Chapter 2. As such, it will be the lifecycle stages and the relationships between these stages that are modelled using the model below.

It should be noted that while systems thinking and systems modelling has a history of working in multidisciplinary contexts, the author could not find evidence of this technique of using systems modelling to compare two similar practices having ever been used or documented before. Therefore, the method below represents an original approach for representing and comparing processes from two different disciplines. This 'appears' to be an original contribution to the software engineering body of knowledge, however is not restricted just to this discipline, and forms part of the contribution for this thesis.

3.1 Approach to modelling

The approach to modelling used in this research project followed the process to systems modelling provided by Senge (1990). The process outlined by Bennet et al. also provided direction in terms of creating a systems model, and the questions that needed to be addressed throughout the process of developing such a model (2005, p. 949). Parts of Checkland's soft systems methodology, as described by Williams (2005), was also used for the purposes of identifying systems characteristics necessary for building a systems model. Ultimately, an informal model was sought in order to utilise the qualitative data that is derived from such a model. While formalisation of the model would have been possible, obtaining quantitative data was not the purpose of this exercise, and instead the general understanding of the implementation for each lifecycle phase was the primary focus.

The first stage was identifying the key elements of the system that were to be modelled (Bennet et al. 2005). As the methodologies of agile and CBPR were the subjects of this model, each phase of each lifecycle model was considered a key element. Secondly, the relationships between elements were defined, identifying the positive or negative causal relationships between elements (Senge 1990). Finally, feedback loops were identified (Senge 1990) along with any intervening factors that influence or control these feedback loops (Bennet et al. 2005).

After the two models were created, corresponding elements were merged in order to create a general model of the two methodologies. Where nodes did not perfectly align, the nodes were abstracted to a more general label that encompassed all relevant nodes. This abstraction allowed for all nodes to be mapped to the general systems model, albeit at the cost of specificity. For discussion on the limitations of the modelling approach, see the final section of this chapter.

3.2 User-focused, Iterative Model

To create a user-focused, iterative model that has a basis in both agile software development and CBPR, individual systems models of these practices were initially created. From these models, a general systems model was derived. This section outlines the steps taken and decisions made in creating this set of models.

In order to create a systems model for the agile software development, the first step was to identify key phases in the lifecycle approaches that sit within agile. The Disciplined Agile Delivery (DAD) framework (Ambler 2012) was used to identify these key phases, which is a non-prescriptive agile framework that supports several lifecycle approaches. The lifecycle phases from two agile approaches are provided to demonstrate to the reader how the general phases from the DAD framework map back to these prescriptive approaches. As Scrum and XP were identified as the most commonly practiced methodologies within agile (VersionOne 2012), the life cycle phases from these methodologies were used. This decision was made based on the assumption that the domination of these methodologies within Software Engineering were likely to mean that the average software engineer's mental model for an agile lifecy-

cle would largely line up with these practices. This position is further justified by the fact that Scrum is the most widely recognised agile software development methodology (Holmström et al. 2006; Project Lifecycle Services 2007). The common identified phases of these agile approaches are displayed in Table 3.1. Processes in the left column represent phases of the Scrum lifecycle, as outlined by Ambler (2012). Processes in the middle column represent equivalent phases of the XP lifecycle, as described by Nayab (2011). The final column represents a general agile phases, as taken from the DAD framework.

Table 3.1: Key elements within general agile methodology

Scrum Lifecycle Phase	XP Lifecycle Phase	Agile Lifecycle Phase
Planning	Planning	Planning
Requirements, Product Backlog	User Stories	Requirements
Architectural vision	Design	Architecture and Design
Construction	Coding	Construction
Iteration Review and Demonstration	Testing	Testing
Enhancement Requests and Defect Reports	Listening	Evaluation

Once key elements (in the form of lifecycle phases) had been identified, the second step was to identify causal relationships between these elements (Senge 1990; Bennet et al. 2005). A basic flow-on effect was recognised between each sequential phase of the lifecycle approach (for example, general relationships such as 'the more requirements that are identified, the greater the overall design efforts that will be required' were identified). These relationships were extracted based on the DAD framework. The outcomes of this exercise can be seen in Figure 3.1.

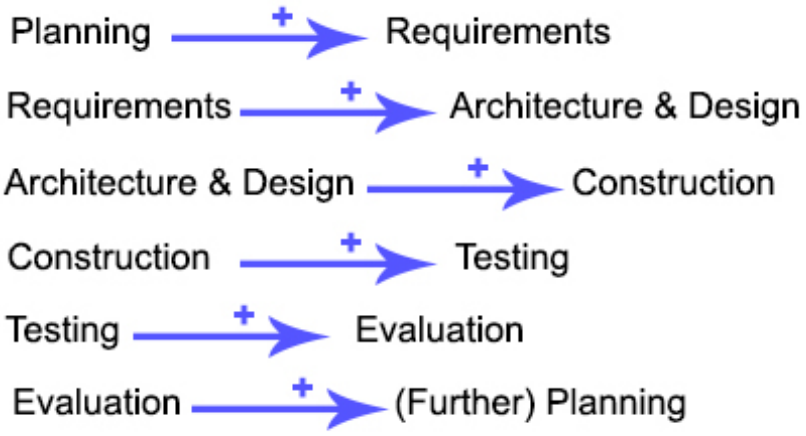


Figure 3.1: Initial relationships between key elements

Finally, feedback loops within the model were identified (Senge 1990) along with intervening factors that have influence or control over these feedback loops (Bennet et

al. 2005). It is clear from Figure 3.1 that there exist overlapping relationships between key elements of the system. For example, while the outputs of the planning phase will positively impact the requirements phase, the outputs of the requirements phase will positively impact the Architecture and Design stage. These types of relationships create a cycle amongst the phases, highlighting the iterative nature of the agile software development approach.

In regards to intervening factors that influence the feedback loops within the system, two main factors were identified by the researcher. Firstly, the remaining work items (known as the product backlog in Scrum) that were identified by the DAD framework have a direct impact on the work that needs to be undertaken. For example, if upon evaluation, many work items are identified, then there will be a large amount of planning that will need to be undertaken in order to complete them. Similarly, if upon evaluation, there is only a small number of work items remaining, the amount of planning that needs to be undertaken is much less. In this way, the number of work items positively impacts the amount of planning that needs to be undertaken. Similarly, by completing the planning, requirements, design and so on, the number of work items that are remaining should be reduced. This reduction should be as a result of customer feedback, and thus would be a product of the evaluation phase of the cycle. Therefore a new relationship is introduced into the system, as seen in Figure 3.2.

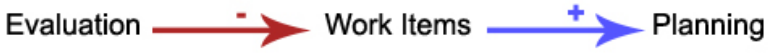


Figure 3.2: Relationships connecting Work Items

The second intervening factor is the budget of any agile software development project. The nature of agile allows for ongoing scope definition, and as such, budget is used as a constraint throughout the project's lifecycle (Stepanek, 2005). Note that deadlines are also used to constrain agile projects, however due to the reoccurring theme of budget constraint through the literature, this was chosen as the constraining factor in this model (see Limitations of Model section below). Budgeting within an agile development process entails feature trade-off, that is, allowing customers to refine and change requirements during throughout the project but only adding new ones if others are removed (Stepanek 2005). In this way, the budget of the project directly influences the number of work items that can be completed throughout the life of the project. This adds a further relationship to the work items factor, as seen in Figure 3.3.

Figure 3.3: Causal relationship between Budget and Work Items

Finally, by combining this set of relationships, we are left with an informal systems model of the agile development process, identifying causal relationships within the

process, as inferred by the researcher. This final model can be seen in Figure 3.4.

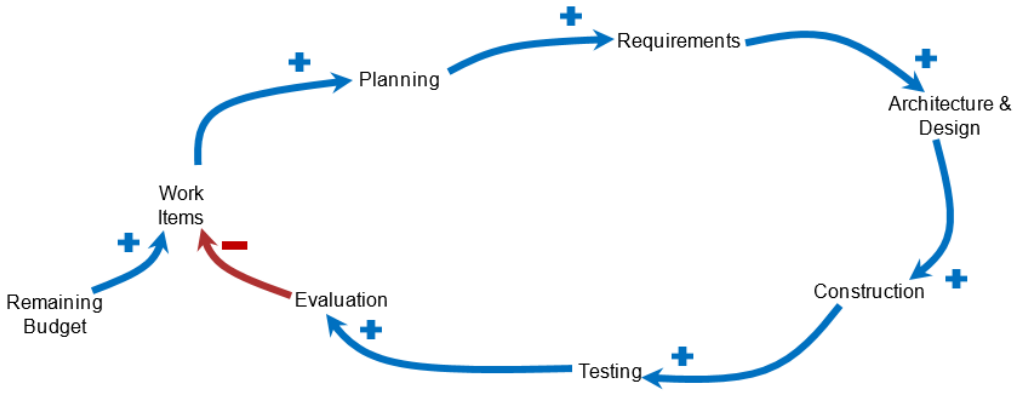


Figure 3.4: Derived agile systems model

After constructing the agile systems model, it was necessary to create an equivalent systems model for CBPR. As above, the first step in this process was to identify key phases within the CBPR process. The phases within the model presented by Israel et al. (2005) as seen in Chapter 2 were used to identify key elements within CBPR. The phases from the CBPR process of Participatory Design as presented by Hagen et al. (2012) are included in order to demonstrate to the reader how general phases within the CBPR model can be mapped back to prescriptive phases within a CBPR approach. Participatory Design was chosen as an exemplar CBPR practice due to its recognition by the participatory research community (Minkler & Wallerstein 2008). The identified approaches within these approaches are displayed in Table 3.2, with Participatory Design phases in the left hand column and CBPR phases in the right hand column.

Table 3.2: Key elements within general CBPR methodology

Participatory Design Phase	CBPR Phase
Identify	Assessment
Define problem space and objectives	Identify priority concerns and research questions
Position and Concept	Design Intervention of policy research
Create	Conduct Intervention of policy research
Use and monitor	Feedback and Interpretation of findings
Evaluate	Translate Research Findings

Once the key elements of the process had been identified, causal relationships between these elements were identified. For this process the work of Israel et al. (2005) was again consulted, identifying the process from phase to phase within the CBPR context. Much like agile, a basic flow-on effect was evident between CBPR phases,

and these relationships were documented. The outcomes of this exercise can be seen in Figure 3.5.

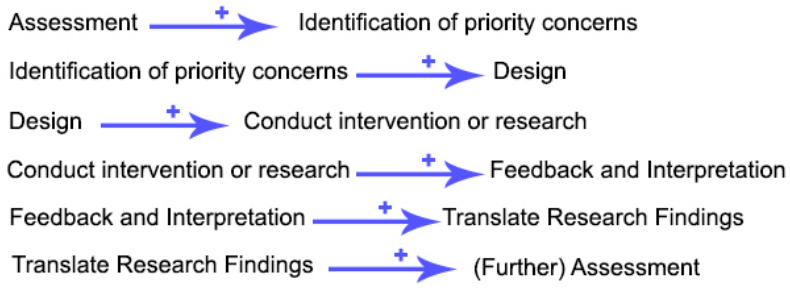


Figure 3.5: Initial relationships between key elements

The final stage of this modelling process was to identify feedback loops and factors that influence the feedback process. As in the agile model above, there is clear overlap between the phases and relationships identified in Figure 3.5. When combined, these phases create a feedback loop within CBPR, highlighting the iterative nature of CBPR. It is noted that this iterative characteristic is also highlighted in the model presented by Israel et al. (2012). Along with identifying feedback loops, influencing factors must also be identified.

The main driving factor within the CBPR process as identified by Israel et al. (2012) and Minkler & Wallerstein (2008) are priority health concerns within a given community. These health concerns represent issues that must be resolved for the betterment of the community of interest, and thus drive everything from the outcomes of the initial assessments through to the priority of the research questions and findings. However, it should also be the case that by undertaking CBPR practices, the severity and magnitude of health concerns within the community should be reduced. This reduction should be evident through the research findings, and thus a new relationship is formed. This new relationship can be seen in Figure 3.6.

Figure 3.6: Relationships connecting Health Concerns

In the literature reviewed by the researcher, there did not seem to be any constraint in particular that is common when undertaking CBPR. Parker et al. (2003) identify cost as one of the difficulties in conducting CBPR, however this was not necessarily a common theme in other publications. It was the decision of the researcher to tentatively include this as a constraining factor to the system, which would then be validated or refuted throughout the case studies conducted with practitioners. The reason for this decision is that given that CBPR practices were also undertaken as projects (just as agile is), it was believed to be a fair assumption that such activities would be constrained by a budget (see Limitations of model section below). Through

the combination of all relationships and constraints, an informal systems model of CBPR was produced, identifying causal relationships within the process, as inferred by the researcher. This final model can be seen in Figure 3.7.

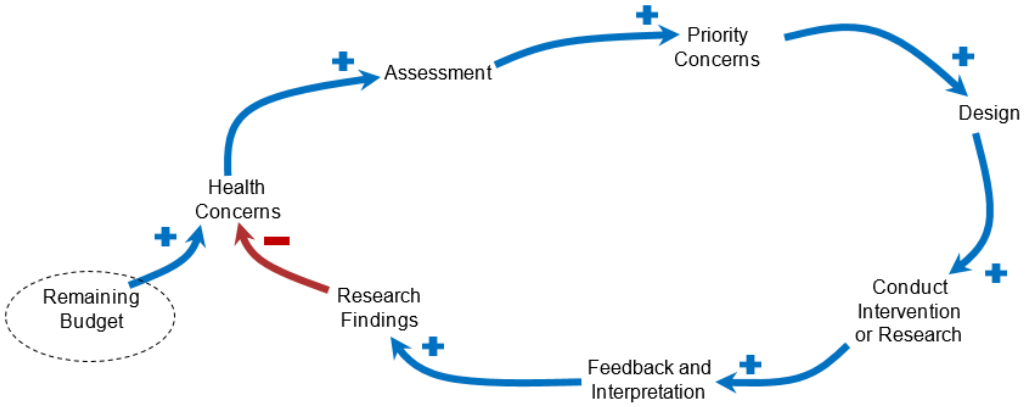


Figure 3.7: Derived agile systems model

It can be seen that agile software development and CBPR practices share a similar shape within their systems model. More than just being similar, they actually share the same pattern as the balancing feedback loop structure described by Senge (1990) as demonstrated in Chapter 2. While not initially obvious why this is the case, the fact is that both processes are designed to achieve a given goal, and therefore by their nature will follow the balancing feedback structure (Senge 1990). Now that the two models have been created, it is necessary to combine the two into a general systems model that can be mapped back to both practices.

In order to create a general model of agile software development and CBPR practices, corresponding nodes within the two derived systems modelled were abstracted to a common element that encompasses both activities (as described above). This process involved exploring the similarities between nodes, and defining an appropriate label that represented these similarities. This process of comparing and labelling is described below for each set of corresponding nodes between the agile and CBPR systems models.

The first set of nodes that were compared were the planning and assessment nodes for the agile and CBPR systems respectively. The focus for the planning node in the agile model is to work with stakeholders to scope the iteration, modelling an initial architecture of the system and creating estimates for the iteration (Ambler 2012). The emphasis for the assessment node of the CBPR node is on assessing community strengths and dynamics, where the power lies within the community and which key stakeholders need to be involved to ensure a community voice (Israel et al. 2012). As all activities described by both agile and CBPR could be constituted as planning activities, the general node for this phase was assigned the label 'Planning'.

The requirements node for agile and Identify priority concerns and research ques-

tions (shortened to 'Priority Concerns' in the model) for CBPR were then compared. The focus for the requirements node for agile was to determine what the users needed the system to do (generally in the form of user stories). The emphasis for the Identify priority concerns and research questions phase was to identify the major health problems that were impacting the community that the project may address, the factors that contribute to said health problems and the key research questions necessary for these health concerns (Israel et al. 2012). In a general sense, both practices are gathering the requirements necessary for undertaking their particular work for the project, and thus the general node was named 'Requirements'.

The architecture and design node for agile and Design node for CBPR were compared, with similarities noted even in the name of the nodes. The architecture and design node for agile is concerned with the design of the features that are being built or added to the system (Ambler 2012), while the design phase in CBPR is concerned with designing interventions and/or policy research for the given health concerns (Israel et al. 2012). This node was generalised to 'Design'.

The construction phase in agile and the phase concerned with conducting the designed intervention or research were compared, due to their similar nature of implementation. The construction phase in agile puts emphasis on developing high quality software artefacts, with close collaboration of stakeholders (Ambler 2012). The conducting stage in CBPR is concerned with actually undertaking the process or method designed in the previous phase (Israel et al. 2012). As both sets of practices are concerned with implementing the design, the corresponding node was labelled 'Implement Design'.

The testing phase in agile and feedback and interpretation phase in CBPR were compared due to their emphasis on ensuring the design and implementation were performing in the way that were expected. The testing phase in agile is concerned with finding defects in the given software artefact (Ambler 2012), while the feedback and interpretation phase in CBPR is concerned with sharing the results of the research with all stakeholders to ensure what was found makes sense (Isreal et al. 2012). Due to the correspondence with stakeholders to test both functionality and sanity checks between these phases, the label 'Testing' was given to this node.

The evaluation phase in agile and translate research findings in CBPR (shortened to 'Research Findings' in the model) were then compared. The evaluation phase in agile is concerned with working with stakeholders to ensure that the software meets the requirements of the users, while disseminating and translation of research findings in CBPR evaluate exactly which findings should be disseminated to the community and what roles community partners should play in publishing the results of the research. As both phases are broadly involved with evaluation activities, this node was given the label 'Evaluation'.

Finally the nodes which drive the feedback loops, namely the work items node for agile and health concerns node for CBPR were compared and generalised. As described above, the work items in agile constitute the features or issues that need to be addressed by the project team, while health concerns are issues that must be resolved for the betterment of the community. These nodes both represent the idea of

elements that need to be addressed and resolved based on some existing view of the environment or community, and thus were given the label of 'Remaining Issues'.

A summary of the descriptions above are given in Table 3.3. This table provides the original agile and CBPR labels, along with the newly defined general labels defined above.

Table 3.3: Newly created generic labels

Agile Phase	CBPR Phase	Generic Label
Planning	Assessment	Planning
Requirements	Identify priority concerns and research questions	Requirements
Architecture and Design	Design	Design
Construction	Conduct Intervention or Research	Implement Design
Testing	Feedback and Interpretation	Testing
Evaluation	Translate Research Findings	Evaluation
Work Items	Health Concerns	Remaining Issues

After identifying the key elements in the generic systems model, the relationships between these elements must be established. Fortunately, the new node structure reflects the original structures within both the agile and CBPR models, and therefore can be directly translated to the general model. Along with the labels identified in Table 3.3, the Remaining Budget Node will also be included, with the purpose of future validation as discussed above. The overall model can be seen in Figure 3.8.

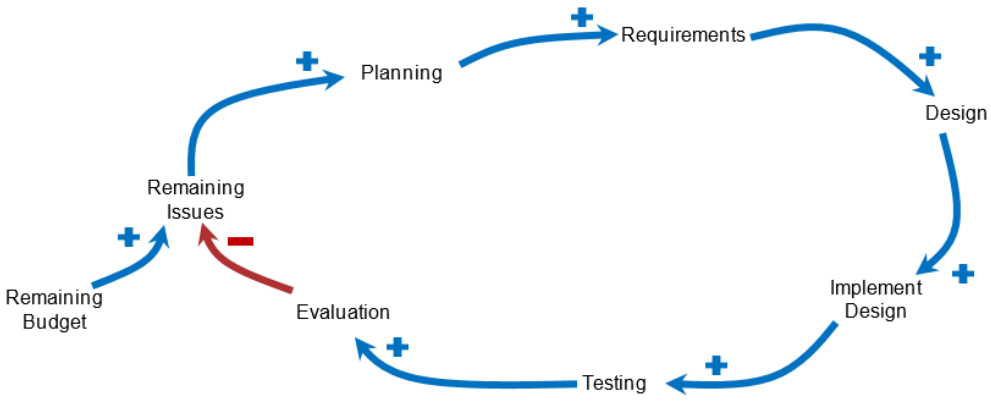


Figure 3.8: User-focused, iterative systems model

This general model can now be used in order to compare and contrast agile and CBPR, as described in the next chapter. The case study aspect of this project (whose methodology is described in Chapter 4 and results are described in Chapter 5) involves questioning experts in the two disciplines to describe certain characteristics of agile and CBPR, using the model as a tool for obtaining this information. Two key

pieces of information are being sought through these methods. The first is to distinguish differences in implementation between these two sets of practices, in order to determine if there are things that each discipline can learn about the other. Differences between agile and CBPR may highlight activities that have potential applicability for transfer between the disciplines, and thus represent learning opportunities in this project. The second significant piece of information being sought is the investigation of exactly where each discipline is engaging their users the most while undertaking these practices. This information is being investigated in order to answer the research question presented in Chapter 1 of this thesis, 'Are there correlations between user engagement at particular lifecycle stages and issues experienced when implementing user-based, iterative practices?'. In order address this question, participants of the case study will be asked to identify how much user engagement generally occurs at each stage of the model for their disciplines practices, along with what they feel are the biggest challenges posed when undertaking these particular activities.

As identified in the next section, there are identified limitations of the general model created within this section. Of course, it is possible that limitations exist that have not occurred to the researcher, or may not be immediately obvious to someone who is not used to undertaking the practices that it represents. For this reason, when conducting the case study in this project, practitioners of agile and CBPR will be queried on the applicability of the general model to their experiences in the field, and improvements to the model will be sought from these experts.

3.3 Limitations of Model

The general model described in this chapter is limited due to its inherent abstract nature. While all efforts have been made to ensure that each element of the model accurately reflects corresponding phases of the agile software development and CBPR approaches, this does not come without cost. It was necessary to remove and abstract details of specific practices in order to create the general model. As such, it is possible that details or pieces of information about certain practices that are potentially useful for the project may be lost when interviewing experts in relevant fields, simply because the particular phase which encompasses those practices are no longer present. It is also possible that the abstractions taken have removed information that is potentially important for practitioners, and as such, this will not be known until the model is already being used within the case study itself. These limitations refer to issues inherent when abstracting information from a system.

A second limitation of the model is that the only resource constraint specifically identified is that of the budget. There are two potential issues surrounding this design decision. Firstly, the idea of having fixed deadlines for agile is not represented at all, which could potentially mean that issues surrounding deadlines are missed during case study interviews. This decision to include budget as the constraining factor was based on the amount of discussion that surround the constraints of money in agile literature compared to the amount of discussion regarding time constraints. The sec-

ond issue is that CBPR does not generally attribute budget as a constraint (based on literature), however this has been inferred by the researcher. In order to address this, all CBPR practitioners will be questioned about the existence of this node, and asked if it appropriate to include such a constraint on the system.

The third limitation is that these models are based on the researcher's interpretation of the literature, resulting in two potential issues. The first is that the researcher may have misunderstood parts of the literature, resulting in an incorrect assumption or design decision to be made. The second is that there may be differences in implementation for agile and/or CBPR than what is presented in the literature. In order to address these issues, the models will be verified by experts in agile and CBPR, asking them to make alterations to the model where they see appropriate.

The final limitation is that the models presented here follow Senge's and Bennett et al.'s methodologies only as far as to create informal models. While it is possible to extend this general model into a formal model, it was felt that due to the qualitative nature of the research, an informal model that was directed towards gathering qualitative information was more appropriate for this project. While this isn't necessarily a limitation for this project itself, anyone who wanted to reproduce the steps undertaken in this chapter for creating a systems model would be limited to the creation of informal systems models and not formal models.

Despite identified limitations, it was the hope of the primary researcher of this project that this general model would serve as an appropriate tool for identifying similarities, differences and user engagement across the practices of agile software development and CBPR. The design of the case study which would utilise this model is described in the following chapter, with results presented in Chapter 5.

Chapter 4

Methodology

This chapter outlines the processes that were undertaken in order to gather data regarding the implementation of agile software development in software engineering and CBPR in mental health research. The purpose of this work is to identify the feasibility of software engineering and mental health research learning from one another. This fits in with the overall objective of this study to address the historically inward-facing nature of software engineering, and to identify the feasibility of sharing knowledge between software engineering and other disciplines. An exploratory case study is designed using the framework provided by Runeson & Host (2008) in order to compare agile software development practices against CBPR. The case study consists of interviews performed with a sample of experts from each discipline, and the data from which is analysed using a grounded theory approach to analysis. The results of this analysis are presented in Chapter 5.

4.1 Approach to research

As outlined by the research questions in Chapter 1, this project was concerned with identifying areas in which software engineering and mental health may be able to learn from one another. In order to identify these areas, an exploratory attitude to research was needed, as there was no precedent within the literature for comparing practices across disciplines. Indeed, there was no precedent for this type of interdisciplinary research within software engineering. The first step in this process involved identifying the type of data collection process that would be most effective for identifying areas for interdisciplinary knowledge sharing. This process involved comparing the strengths and weaknesses of qualitative and quantitative data collection processes.

4.1.1 Qualitative vs Quantitative Research

Whilst an extensive comparison between qualitative and quantitative data is outside the scope of this project, the considerations taken by the researcher when choosing a research approach within this project are presented below.

Qualitative research is defined as research that is undertaken in a natural setting, where the researcher gathers words or pictures, and inductively analyses them in or-

der to derive meaning from the participants of the study (Creswell 1998). This is in comparison to quantitative methods, which involve empirical investigations involving statistical, mathematical or numeric data and analysis (Given 2008). Software engineering has a historical legacy to lean towards quantitative research projects, however the researcher felt that this was not reason enough to dismiss the possibility of using a qualitative approach.

Creswell (1998) explains that in qualitative projects, research questions typically ask 'what' or 'how' questions. This is in contrast to quantitative projects, which often ask 'why' questions, and search for explanations or cause and effect relationships. Clearly from this simple measure, the research questions in this research project lean towards a qualitative research approach (such as 'What are the main challenges faced when conducting user-focused, iterative methodologies?' and 'What is the general attitude towards interdisciplinary knowledge sharing in the software engineering and mental health research domains?').

In order to present a balanced view, however, the strengths of quantitative analysis must be identified. Quantitative research often allows for results to be quickly and numerically analysed (Given 2008). Analyses within quantitative projects are also less susceptible to being overwhelmed by data than qualitative research, with less difficulty in maintaining and demonstrating rigor (Anderson 2010). Quantitative research is also appropriate for testing hypotheses, as numerical methods can be effective for presenting explanations for certain phenomena (Given 2008).

Whilst the strengths of quantitative research were considered, it was ultimately decided that a qualitative approach to data gathering and analysis was appropriate for this project. This was due to the fact that this project was investigating the feasibility of interdisciplinary approaches within software engineering, and was not trying to prove or disprove a pre-existing hypothesis. This project therefore used qualitative data as its primary data source, in order to compare the way agile software development and CBPR practices are performed. Along with deciding on the type of data that was going to be collected, a research tradition in which to perform the qualitative research also needed to be chosen.

4.1.2 Determining a qualitative research tradition

In order to perform a qualitative research project, a framework within which to operate had to be chosen. Creswell (1998) presents five qualitative research traditions for qualitative research, namely a biography, phenomenology, ethnography, grounded theory and case study. Each of these traditions are presented briefly, along with justification for why they were or were not appropriate for this study.

A biographical study involves research surrounding a single individual, and their set of experiences (Creswell 1998). This research project was concerned with investigating a set of practices within two disciplines, which is concerned with several individuals and complex behaviours between these individuals. Therefore the biographical approach to research was not deemed appropriate for this study.

A phenomenological study is concerned with understanding the essence of expe-

riences about a phenomenon, generally from a philosophical perspective (Creswell 1998). While this project was concerned with studying two phenomena (namely the implementation of agile and CBPR practices within their domains), it was the actual implementation and issues felt that were being sought, rather than philosophical perspectives behind the practices. Therefore, this approach was also not deemed as appropriate.

An ethnographic study is concerned with the study and description of a cultural or social group (Creswell 1998). As the nature of this project was on particular practices rather than groups of people, this was deemed an inappropriate research technique.

Grounded theory is concerned with developing a theory grounded from the data in the field (Creswell 1998). This project wasn't actually concerned with creating a theory *per se*, and was more aligned with the exploratory ideas of interdisciplinary feasibility within software engineering. It is noted that a grounded theory to analysis did appeal to the researchers, and is further explored below.

A case study is used for creating an in-depth data analysis of one or more cases bound within a system (Creswell 1998). As this study wanted to investigate the way in which agile was implemented (within the software engineering domain) and the way in which CBPR was implemented (within the mental health domain), this approach seemed the most appropriate out of the five traditions. It is for this reason that a case study approach was taken towards the research, which would consist of two cases (agile and CBPR).

For the sake of this project, the definition of an exploratory case study provided by Runeson & Host (2008) will be used. An exploratory case study is one in which the aim is to generate new hypotheses and ideas for research through the investigation of what is currently happening with the investigated case. This is in contrast to an explanatory case study, which seeks an explanation about a given problem or situation (Runeson & Host 2008). This project adopted an exploratory case study approach in order to explore the native implementation of agile and CBPR, and to explore any possible opportunities for learning between the two practices.

4.1.3 Interviews for data collection

The survey technique described by Robson (2002) was used, with particular emphasis on the process of collecting information from a sample of a population by means of an interview. Interviews were identified as effective mechanisms for data collection as they allowed the researcher to further explore the investigated practices, along with any other interesting or unexpected topics that arose.

The primary data source for this project will come in the form of interview transcripts from experts in the relevant domains. The methodology for collecting, analysing and reporting this data was developed in consultation with the supervisors of this project and the NIMHR staff, and is described further in this chapter.

Through discussions with relevant project stakeholders, it was decided that the interviews would be conducted with experts in each domain. Experts were chosen for interview in this case study due to their extensive knowledge in their given do-

main. Based on the project advisors' experience, it was expected that there was a good chance that experts would be aware of the foundations of particular practices and issues being experienced across their discipline. Experts were also chosen due to the limited timeframe that the project was under, and thus a larger case study with several users of varied experience was simply not possible. The potential bias that may have been introduced by only including experts in the participant pool is addressed below. In order to improve the validity of the project, data source triangulation was used throughout the project.

Triangulation is a method for improving the precision of research by selecting different sources of data to confirm results in order to build a coherent picture (Easterbrook et al. 2008). Runeson & Host (2008) describe data source triangulation for empirical research methods as the process of collecting data from more than a single data source, and in doing so, providing a broader picture of the investigated phenomenon. The case study presented in this project will employ data source triangulation techniques by considering the perspectives of experts across two different disciplines. Additionally, within each group of experts, it will be ensured that there is a range of academic and industry experience in order to examine a wide context for how these practices are being applied.

The case study design presented below describes the actions taken in order to collect data in a consistent manner across all participants. Elements of the Case Study Protocol used for this project are described below, and attached as appendices where appropriate. The Case Study Protocol acts as a guide for data collection, provides an explicit description for the research to be undertaken and allows for relevant feedback (Runeson & Host, 2008). Along with ensuring consistency across the data gathering activities, the Case Study Protocol also acted as a guide for the Ethics Proposal required to undertake this type of research.

4.2 Case Study Design

This section describes the case study which involved the investigation of agile software development techniques within the context of software engineering and the implementation of CBPR techniques within mental health research. These are considered separate cases and are investigated independently. A cross-case comparison performed at the conclusion of the interview period to identify opportunities for interdisciplinary knowledge sharing to occur between the disciplines. The following subsections address each component of the case study design, as identified by Runeson & Host (2008).

4.2.1 Objectives

The purpose of this project is to identify the feasibility of knowledge sharing between software engineering and mental health research. This involved attempting to identify areas from which the disciplines could learn from one another. One train of thought

regarded the ideas of sharing 'lessons learned' when undertaking the practices of agile and CBPR. In this way, pain points that one discipline was facing may have been able to be remediated by solutions that the other had come up with. Along with this idea, the researcher identified that the participants themselves may have had ideas about where their practices may be effective. Therefore, participants would also be questioned about knowledge sharing and potential areas for interdisciplinary collaboration. These intentions culminated in the five main questions that will be addressed throughout the case study, which are as follows:

- What are the major challenges and pain points felt when implementing agile Development practices/CBPR activities?
- Is the model created in Chapter 3 an effective description tool for both agile Development practices and CBPR activities?
- Were there influences from other disciplines in the creation of iterative, user-based development methodologies such as agile Development and CBPR?
- What is the general attitude to knowledge sharing across software engineering and mental health research, with regards to current techniques and practices?
- What techniques do experts feel are particularly effective in their domain that could be translated to other disciplines?

The ultimate objective of searching for the answers to these questions is to demonstrate whether or not the methods proposed in this project are an effective mechanism for facilitating a two-way flow of knowledge between disciplines. The secondary objective is to identify practices that may be beneficial to implement within the domain of software engineering from the discipline of mental health research, as well as conversely identifying methods that may provide benefit within the domain of mental health research from the discipline of software engineering.

4.2.2 Procedures and Protocols for data collection

As this project intended to use human subjects, it was a requirement of the Australian National University that the proposed case study was approved by the university's Human Research Ethics Committee (HREC). The considerations that were addressed during this ethics approval process are addressed throughout this section.

Key ethical considerations that needed to be addressed in order to conduct this study included that all participants gave their written consent to participate; that the project was approved by the HREC; there was an agreed level of confidentiality for participants; results would be handled sensitively; and feedback would be provided to participants by the researcher upon the completion of the project.

Potential participants in the software engineering and mental health research domains were identified through a combination of requesting project advisors in both disciplines for potential candidates who would be appropriate for the study, and

through the websites of organisations that performed agile or CBPR. As advised by Runeson & Host (2008), due to the qualitative nature of this type of research, it is generally better to find subjects that display a difference in characteristics. These include those who have experience in different roles as well as displaying personality types, rather than try and replicate similarities across participants. For this reason, a selection of potential participants were chosen that had a range of academic and industrial experience in practicing agile development and CBPR activities. This decision also allowed data source triangulation to be performed, providing a broader picture of the current practices across industry and academia as well as increasing the precision of the research.

Formal invitation emails were sent to all potential participants (see Appendix D) prior to any commitment being made to participate to the study. A consent form (see AppendixD) and Information Sheet (see AppendixC) were given in this e-mail, so that participants were provided with as much information possible before participating in the study. Those who were e-mailed were also invited to e-mail the primary researcher with any queries they had in order to find out any additional information, or clarify any concerns that may have arisen.

Upon receiving written consent from participants, an interview time and location was set up with each participant. Consent forms allowed for participants to choose their preferred level of confidentiality (being addressed under a pseudonym, having their occupation disclosed or complete anonymity), and requested permission to audio record the interview. These interviews were to be conducted either in person or online using Skype.

At the commencement of each interview, the interviewer confirmed that the participant was still happy for the interview to be recorded (where applicable), and if so, recorded the proceedings using an in-built laptop microphone, and a backup recording using the in-built microphone and recording feature of a mobile phone. The decision to record the interviews was based off a recommendation from Runeson & Host (2008), who advise that even when taking notes, it can be difficult for a researcher to record all details, and the researcher should therefore record the interview in an audio or video format.

As outlined in the approved Ethics proposal for this project, raw data from the interviews (in the form of recordings and transcripts from the interviews) will only be accessible by the primary researcher, and the supervisors who are overseeing this project. Interview recordings and transcripts will be stored for one year following the submission of this thesis, after which time they will be destroyed.

4.2.3 Collecting evidence

The primary form of data collection for this project was through the use of interviews. In each interview, the primary researcher asked a series of questions to each participant about practices and experiences in their respective discipline. For a list of indicative questions for experts in software engineering and mental health research, see Appendix E and Appendix F. The questions asked were based on the case-study ob-

jectives stated above. These questions were intentionally open in nature, in order to allow participants the opportunity to provide a broad perspective on the given topic, based on their experience in the domain.

The interviews were semi-structured, in that for each interview a set of questions were planned, but not necessarily asked in the same order each time (Runeson & Host, 2008). It was occasionally the case that a participant addressed a point that was more aligned with further question, and in these instances, the further question may have been asked sooner than was previously planned. In this way, the direction of the conversation in the interview had influence on the order of the questions, which allowed for further exploration for topics of interest.

During the interview, participants were also asked to adapt the general model created in Chapter 3 to better reflect the practices they had witnessed in the past. For the software engineering experts, this meant changing the model to better suit their observations of how agile has been performed, and for mental health researchers, this meant changing the model to align with the ways they had witnessed CBPR and Participatory Practices being performed. The general model that was given to each participant is given in Figure 4.1. It is noted that the researcher decided to remove the pluses and minuses from the model for sake of the interview. This was based on the feeling that the diagram may have been too busy and thus confusing for the participants. Colours were seen as being sufficient for the sake of this exercise (accompanied by an explanation from the researcher). As discussed in Chapter 5, this decision did not appear to reap any benefit, and for the next iteration of the model, the researcher would reinstate the pluses and minuses for the sake of clarity.

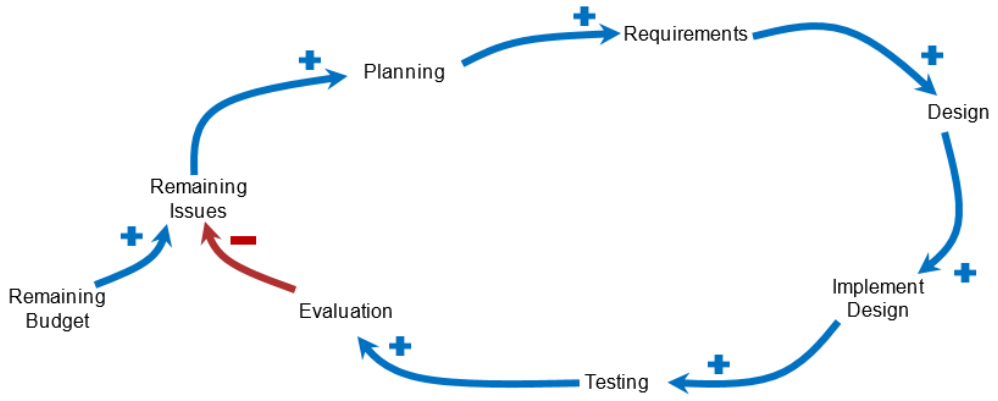


Figure 4.1: General model given to participants for adaptation

Along with adapting the model to align with their experiences, participants were also asked to mark on the diagram the places that they felt had the most user engagement. Participants were asked to put two crosses next to the phases they felt the generally always engaged users, one cross for stages that occasionally or usually engaged uses, and leave blank and stages that generally did not involve user engagement.

As the researcher was in direct contact with participants and data was collected in real time from this interaction, this approach falls under Lethbridge et al.'s (2005) classification of being a first degree data collection technique. The differences between first, second and third degree data collection are that in first degree the researcher is in direct contact with the subjects, in second degree the researcher collects the results from the subjects, but is not in direct contact with them, and third degree data collection as an independent analysis of research artefacts (Runeson & Host 2008). Runeson & Host (2008) explain that while first degree methods are generally the most expensive data collection technique to apply, they have the advantage of being able to control what data is being collected and the method for collection. In this case, conducting the interviews was fairly inexpensive, and as such, this was an appropriate technique for collecting data for this project. As discussed above, the benefit of using several participants is that the effects of interpreting the data from a single data source are limited. This allowed for results to be triangulated, which will generally produce a stronger conclusion than when conclusions are based on a single data source (Runeson & Host, 2008).

Upon the conclusion of each interview, the recordings were transcribed into an electronic format by the primary researcher for review by the participant. As observed by Runeson & Host (2008), sending back transcripts and observations allows participants to correct any mistakes in the raw data. Each participant of this study received the transcript of their interview via e-mail, and responded with any desired changes to the data. All changes were made and transcripts updated as requested.

4.2.4 Analysis of collected data

The purpose of the analysis presented in this case study is to explore the responses given by participants and generate hypotheses from the data. As highlighted by the research questions of this project, the primary concern is in identifying the feasibility of interdisciplinary learning and sharing knowledge across disciplinary boundaries. It is through this lens that themes and relationships are analysed.

Runeson et al. (2012) identify the importance of using a structured approach to analysis, highlighting the need to plan and record decisions made throughout the process. The primary framework for data analysis was a specialised framework for analysing case study data through a grounded theory approach, in the form of Runeson & Host's editing approach to analysis (Runeson & Host 2008). Runeson & Host present four techniques for case study data analysis (where each is an adapted version of a grounded theory approach to data analysis). The four techniques presented by Runeson & Host (2008) are:

- Immersion approach: in which there is little structure in analysis, and results are reliant on the skill of the researcher.
- Editing approach: Which exists within a structured framework, however starts off with at most a few predefined codes, and instead allows codes to emerge from the data.

- Template approach: A more formal approach that editing, where codes are pre-defined based on research questions.
- Quasi-statistical approach: A formal approach with pre-defined codes, and others based on calculation of word frequency within the data set.

Runeson & Host identify Editing and Template approaches as the most commonly used throughout software engineering case studies. As this case study is exploratory in nature, and areas for potential learning were expected to emerge from the data, an editing approach to data analysis was chosen. While this approach was largely prescriptive, there were times throughout the analysis when supplementary theory to analysis was sought, and during these times Robson's *Real World Research* (Robson, 2002) was consulted for reference.

Analysis begins through the process of initial, or *open*, coding (Runeson & Host 2008). To this end, the researcher printed off each transcript, marking words (or 'codes') next to each sentence that seemed appropriate. These words generally related to the theme of the sentence, as well as any phenomenon that that seemed important to the participant, or unexpected from the perspective of the researcher.

Following this initial coding session, the transcripts were imported into the commercially available analysis tool *NVivo qualitative data analysis software* (QSR 2012). Each identified code from the initial round of coding was put into NVivo as a 'node'. Following this, each digital copy of each transcript was coded again. The difference was that when a sentence fit within codes that existed within the software that the researcher hadn't originally picked up on, the sentences were also attributed these codes. In this way, sentences were retrospectively fitted to codes, until the researcher was satisfied that all codes lined up with appropriate corresponding sentences.

This represented the process of open coding within the system. The next phase of analysis involved assembling codes into broader themes, within the process of axial coding (Runeson & Host 2008). Further direction was sought from Robson (2002) regarding the steps taken during the stage of axial coding. A coding paradigm was developed, which identified one or more central phenomena for each case, casual conditions for these phenomena, strategies that have arisen form the presence of these phenomena, intervening conditions to the system and consequences that flow from the phenomena. Examples of these coding paradigms for the two cases can be seen in Chapter 5.

Following this process, the final stage of selective coding (Runeson & Host 2008) was performed. This involved the integration of the categories identified through the coding paradigm when performing axial coding. As recommended by Robson (2002), hypothetical propositions were presented, in the forms of models. These models were discussed, along with implications and future work that flow from them.

The process outlined above describes Runeson & Hosts's editing approach to qualitative data analysis. This approach is a specialised version of grounded theory, tailored for case studies within software engineering. The analysis in Chapter 5 steps through each of these stages in more detail, giving examples of the data that was collected and needed at each stage of the process.

4.2.5 Reporting

This thesis acts as the reporting mechanism for the case study undertaken during this project. Robson (2002) identifies that a case study report should give context to the case, provide a history of related work, provide data in a basic form and articulate the researcher's conclusions. This information is portrayed across chapters 1, 2, 5 and 6 of this thesis respectively.

Summarised versions of the data are provided in the next chapter, along with key quotes from relevant interviews. For information involving the model in Figure 4.1, aggregate results are presented in Chapter 5, where results from participants in the same discipline are laid over one another (copies of the original diagrams provided in interviews can be found in Appendix K for software engineering and mental health research participants).

Specific care has been taken through the reporting process of this project, in order to keep the identities of participants confidential for reporting purposes. As identified by Amschler-Andrews & Pradhan (2001), the researcher has a responsibility to maintain the integrity for all participants whilst balancing their duties as a researcher in their attempt to be published. Participants are always referred to under gender-neutral pseudonyms in this thesis for this reason.

4.3 Validity of Case Study

The validity of a case study dictates the trustworthiness of the results, where threats to validity can negatively affect the impact of the research project (Runeson et al. 2012). In this section, potential threats to the validity of this case study are identified, with specific reference to the countermeasures taken throughout the course of the project. Specific reference will be made to the construct validity, internal validity, external validity and reliability of the case study, each of which belong to the validity classification schemes recommended by Runeson at el. (2012) and Yin (2003) for controlled experiments in software engineering.

The construct validity of the project was explicitly addressed before the commencement any data collection. The construct validity of this project refers to the extent in which operational measures that are being investigated actually represent the research questions for the project. Runeson & Host (2008) give the example of construct validity being threatened because the researcher has interpreted the constructs discussed in an answer to an interview question differently than the way the participant intended it. Two explicit steps were taken in order to address the construct validity of the case study. Firstly, the supervisors of the research project were briefed on the interview questions prior to the commencements of any interview to ensure that they aligned with the phenomena being researched. Secondly, all participants had their transcripts returned to them, and were entitled to make any changes they wanted to the transcripts to make sure that what the researcher had recorded accurately reflected their position on any particular topic. In this way, the construct validity of the project was addressed before and after the commencement data collection.

Internal validity is concerned with the examination of causal relationships, specifically when the investigated factor is affected by an unknown factor, of which the investigator is not aware (Runeson & Host 2012). This investigation is not specifically investigating causal relationships, and has a greater focus on relationships between observed factors (e.g. investigating whether there is a relationship between user engagement and project success across the two investigated disciplines). However, for any identified causal links that are identified will be used instead to form hypotheses rather than form conclusions. This is due to the exploratory nature of the project, and any hypotheses that are created will be presented for further investigation.

The broad perspective taken by this project is a natural defence against threats to external validity. Runeson & Host (2008) identify that external validity is the extent to which findings can be generalised, and the extent to which findings will be of interest to those outside of the investigated cases. It is apparent from the interview questions given in Appendix E and Appendix F that no specific project or organisation was focused on in any particular part of the interview. The interviews were intentionally designed to have participants draw upon multiple areas and experiences, and in general participants spoke about several different organisations and projects within each interview. Additionally, given that participants were identified from different areas of academia, industry and even across disciplines, the possibility that the findings will be specific only to those who participated in the case study is less likely.

The reliability of a case study refers to the extent that data collection and analysis are dependent on the specific researcher, which can be threatened if the coding process or interview questions are unclear (Runeson & Host 2008). In Chapter 5, the initial set of codes used in the analysis are mapped back to the original research questions. Whilst further codes were created based on emerging themes from respondents' answers, the links to codes which have been based off the research questions presented in Chapter 1 are explicitly made. Additionally, the base set of questions used in the semi-structured interviews are given in Appendix E and Appendix F, allowing for repeatability of these interviews by a researcher outside of this project. Therefore appropriate measures have been taken in order to ensure that the reliability of this project is fitting for a software engineering case study.

In this section, potential threats to the validity of the project have been addressed, highlighting countermeasures which have been taken in order to prevent these threats from eventuating. The validity classification scheme recommended by Runeson at el. (2012) and Yin (2003) for controlled experiments in software engineering was used, addressing threats to validity across construct validity, internal validity, external validity and reliability. As demonstrated, appropriate measures have been taken in order to reduce threats to the validity of this project as much as possible.

Chapter 5

Results and Discussion

This chapter presents the results of undertaking the case study described in Chapter 4. Findings from the case study are provided, with appropriate discussion distributed amongst these results. A discussion regarding the implications for future interdisciplinary work within software engineering is given, along with any identified strengths and weaknesses of this case study.

5.1 Overview of Interviews

The case study performed as a part of this research project involved interviewing three experts from the discipline of software engineering and three experts from the discipline of mental health. These interviews were conducted between August and September of 2014 by the primary researcher of this project. Protocol was undertaken as outlined in the case study description within Chapter 4 of this thesis and a consent form was obtained from all participants prior to the commencement of their respective interview. All participants gave permission for the interviews to be audio recorded for the purposes of future analysis within the project, and all gave permission to be referenced by occupation.

The end of data collection for this project was dictated by the timeframe that is necessary for an undergraduate program of this nature. A second round of interviews would have been preferable for the purposes of validating the findings found, however this was simply not feasible in the project timeframe (see section 5.4). This time pressure was further exacerbated by the change in project direction during the research period. For a description of this change of direction, see Appendix A. This chapter therefore describes the analysis performed on the initial set of 6 interviews performed with experts in software engineering and mental health.

Throughout each interview experts were asked a series of verbal questions, and were asked to annotate the systems model presented to them when appropriate. The interviewer recorded handwritten notes during the interviews, which were later used as field notes, as well as audio recordings of the interviews. The researcher created detailed summaries of each interview (generally in the form of extensive, yet partial or full transcripts of the interviews). These detailed summaries were provided to each participant to confirm the accuracy of the records, and provide any alterations or addi-

tional information they felt necessary. It is noted that these transcripts were manually created by the researcher, by listening to and typing out the contents of the interview audio files. Attempts were made to use Dragon Speech Recognition Software (Nuance Communications 2014) to transcribe interviews, however this proved to be ineffective due to the lack of accuracy produced by the tool.

The approved detailed summaries were used for qualitative analysis, as presented throughout the remainder of this chapter. NVivo qualitative data analysis software (2012) was used to for open and axial coding activities, and models were produced from within NVivo as a result of these activities. The approach used for analysis was the Runeson & Host's editing approach to analysis, a general framework for analysing case study data through a grounded theory approach (Runeson & Host 2008). The following section presents the results of these analysis activities, presenting each case individually, along with a cross-case synthesis between agile and CBPR.

5.2 Analysis of Data

The purpose of the analysis presented in this chapter is to generate hypotheses from the data collected within the exploratory case study. As highlighted by the research questions of this project, the primary concern is in identifying the feasibility of interdisciplinary learning and sharing knowledge across disciplinary boundaries. It is through this lens that themes and relationships were analysed, and hypotheses referring to these topics were sought. It is noted that while the analysis of this data was guided by the research questions of the project, the method undertaken was flexible enough to allow unexpected themes to emerge from within the data, as demonstrated below.

The primary framework for data analysis used was a general framework for analysing case study data through a grounded theory approach, in the form of Runeson & Host's editing approach to analysis (Runeson & Host 2008). There were times throughout the analysis when supplementary theory to analysis was sought, and during these times Robson's *Real World Research* (Robson, 2002) was consulted for reference.

It is noted that throughout this chapter, participants' names are removed for confidentiality purposes. When it is necessary to refer to a specific participant, they are referred to by an alphanumeric pseudonym. This pseudonym will be of the form 'S######' for software engineering participants and 'M######' for mental health participants. The '#' sign is replaced with a set of digits based on the time that the interview was conducted. A general description of the participants is given in Table 5.1 for contextual purposes, when reading the responses below.

The first step of the analysis process came from coding the participant approved interview transcripts using an open coding process. To do this, the researcher printed off each transcript, highlighting and annotating the sentences that contained a repeated idea, an idea that the researcher found unexpected, or any comments that participants specifically mentioned being important. Codes were based on the content of the sentences, allowing codes to emerge from within the transcripts themselves.

Table 5.1: Overview of participants' background

Pseudonym	Discipline	Experience	Occupation
S081912	Software engineering	Industrial and Academic	Software Engineering Lecturer
S082210	Software engineering	Industrial and Academic	Software Engineer
S082711	Software engineering	Industrial	IT Consultant
M082615	Mental health	Academic	Mental Health Researcher
M091013	Mental health	Industrial	Youth Participation and Technology Coordinator
M091514	Mental health	Industrial and Academic	Product Manager

The reason that codes were initially generated from paper copies of the transcripts is that during the next phase, each sentence was attributed to one or more codes using a software-based tool. By creating an initial set of codes on paper, it meant that during the next phase of the process, sentences could be checked against all possible codes, rather than just the ones found initially.

The next step in the analysis process involved transferring these codes from paper into computer-aided with qualitative analysis software NVivo. This allowed codes to be attributed to participants, and a more coherent picture began to be formed. To do this, each sentence was highlighted and attributed to one or more nodes (where each node is representative of exactly one code). As mentioned above, if there were new codes that sat within sentences that weren't picked up on by the researcher in the initial round of open coding, these codes were also assigned to these sentences. In this way, this was a second iteration of the open coding process.

There were instances where codes could quite quickly and easily be grouped into hierarchies during this process. An example would be the codes "Issues with Agile – Lack of Understanding", "Issues with Agile – Lack of Documentation" and "Issues with Agile – Attitude of Development Team". Quite clearly, these codes can all fit under the category of "Issues with Agile". Using NVivo, these initial hierarchies were created to help the researcher organise and make sense of the node patterns that were emerging. This resulted in the initial, hierarchical coding schema that can be found in Appendix I. This initial grouping of hierarchies was for organisational purposes, however helped form the basis of the next phase of analysis-axial coding.

Axial coding involved assembling data in new ways after open coding, identifying connections between codes based on their content and emerging themes. An initial step in this process was to perform cluster analyses on the data (based on the Pearson correlation coefficient), which grouped codes with similar content together. These cluster analyses resulted in hierarchical trees that can be seen in Appendix J. These trees immediately gave rise to codes that could be jointly associated by simple ob-

servation, such as 'Culture and Management Support' and 'People don't like change' for interdisciplinary attitudes within software engineering, and 'Communication' and 'Feedback' when undertaking CBPR in mental health. While they did give an indication of which nodes were similar to others, this was not sufficient to perform axial coding, and thus a coding paradigm was used.

A coding paradigm was created based on the initial codes found through the open coding process during the axial coding phase. A coding paradigm specifies the central phenomenon (or phenomena) of the study, explores causal conditions that influence the central phenomenon, specifies identified strategies that interact with the central phenomenon, identifies the intervening conditions that influence the strategies and delineates the consequences for this phenomenon (Robson 2002). For both the software engineering and for mental health interviews, two central phenomena emerged from the data. The first was unsurprisingly the cases themselves (that is, agile software development and CBPR). The second were the attitudes and ideas towards interdisciplinary collaboration, which was not necessarily directly connected to the case itself. Therefore a coding paradigm was created for each central phenomena in each discipline. These phenomena can be seen in Table 5.2, Table 5.3, Table 5.4 and Table 5.5.

Table 5.2: Axial coding paradigm for agile software development

Component	Open Codes	Axial Code
Central Phenomena	**Chaos** **Loops within Loops**	"Agile Software Development"
Causal Conditions	**Over Documentation within Software Engineering** **Requirements Volatility** **Adaptation to Iterative**	"Issues with Traditional Approaches"
Strategies	**User Engagement** **Adherence to Strict Timeframe** **Reduce Documentation**	"Emphasis on user engagement and satisfaction"
Intervening Conditions	**Extent of User Engagement** **Lack of Understanding** **Lack of Documentation** **Attitude of Development Team** **Culture and Management Support**	"User, Management and Team support necessary for success"
Consequences	**Dealing With Complexity** **Quality** **Benefits of Automation**	"Conditionally Improved Practices"

Table 5.3: Axial coding paradigm for software engineering attitudes to interdisciplinary collaboration

Component	Open Codes	Axial Code
Central Phenomena	**Interdisciplinary attitudes**	"Attitudes to Interdisciplinary collaboration"
Causal Conditions	**Silo Based Discipline** **People don't like change**	"People like what they know"
Strategies	**Push Out** **Project Work** **Educational Material** **Conceptualisation of Problems** **Modelling** **Law Enforcement** **Environments with Volatile Requirements**	"Practices applicable for pushing out and pulling in"
Intervening Conditions	**Culture and Management Support** **Open-mindedness**	"Willingness of areas to adopt new approaches"
Consequences	**Push Out** **Pull In**	"Improved Practices"

Table 5.4: Axial coding paradigm for CBPR

Component	Open Codes	Axial Code
Central Phenomena	**CBPR**	"CBPR"
	Loops within loops	
Causal Conditions	**Understanding needs of users**	"Understanding the needs of communities"
	Working with communities	
Strategies	**User engagement**	"Emphasis on user engagement and feedback"
	Feedback	
	Strong customer focus	
Intervening Conditions	**User engagement**	"Time is significant constraint"
	Time constraints	"User engagement and management"
	Access to stakeholders	"Understanding of practices necessary"
	Balancing requirements with feasibility	
	Conflicting requirements	
	Communication and common language	
	Dependent on stakeholder engagement	
	Lack of understanding	
Consequences	**Dealing With Complexity**	"Fulfilled needs of communities"
	Give users sense of ownership	
	Generating creativity in users	
	Fulfilled Requirements	

Table 5.5: Axial coding paradigm for mental health attitudes to interdisciplinary collaboration

Component	Open Codes	Axial Code
Central Phenomena	**Interdisciplinary attitudes**	"Attitudes to Interdisciplinary collaboration"
Causal Conditions	**Silo Based Discipline**	"Silo Based"
Strategies	**Health Groups** **Knowledge Sharing** **Science & evidence based approaches** **Marketing** **Genuine engagement** **CBPR**	"Areas that rely on well-formed solutions for their users"
Intervening Conditions	**Culture and management support** **Open-mindedness**	"Willingness of areas to adopt new approaches"
Consequences	**Push Out** **Pull In**	"Areas for improvement"

The final stage of analysis for this exploratory case study was to integrate the categories identified throughout the axial coding process. This stage of analysis is known as selective coding: the product of which are hypotheses based on the data gathered (Robson 2002). These hypotheses are given in the form of models, and can be found in the following subsections. Theses subsections describe the major findings for this case study.

As described throughout this section, themes were generated from within the data itself. In this way, this case study was exploratory in nature (that is, was not intended to confirm some pre-existing hypothesis), and is considered an inductive approach.

5.2.1 Major themes through agile case

The major themes uncovered during the agile case are brought together into the theoretical model found in Figure 5.1. These themes were derived during the selective coding process described above. In this process, themes were drawn from axial codes and the relationships between these codes were specified.

There are two key phenomena identified in Figure 5.1, both indicated by a thicker outline than the other themes in the model. From this model, two broad hypotheses have been generated, as follows,

1. Issues with traditional software development approaches led to the establishment of agile software development practices. These practices emphasise user satisfaction in order to improve the way software is developed. However, this

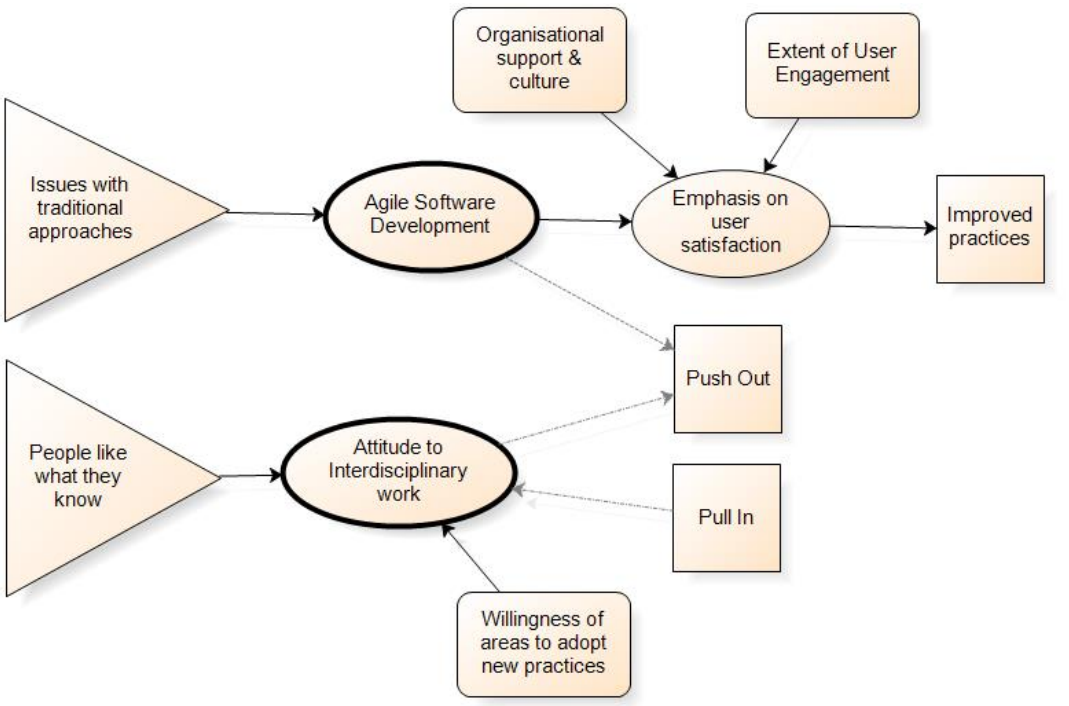


Figure 5.1: Theoretical model of agile software development from case study

improvement can be affected by the support of the organisation developing the software, and the extent to which the customer is willing to be engaged.

2. People in software engineering have typically stuck to what they know, leading to an insular view towards interdisciplinary work. The extent to which software engineering will accept interdisciplinary work is dependent on peoples' willingness to adopt new practices.

Note that there are themes in the model connected by dashed arrows that are not present in either hypothesis. This is not because they reflect the current state of software engineering, but rather they were themes that arose through the interview regarding possible avenues of thought and research for the future. These themes are further discussed in subsection **5.2.5** below.

In order to justify the above hypotheses, evidence will be provided regarding the relationship between each connected pair of themes. This will involve breaking down each hypothesis into its constituent parts, and providing evidence for each relationship in the model. We will begin with the first hypothesis.

> *Issues with traditional software development approaches led to the establishment of agile software development practices. These practices emphasise user satisfaction in order to improve the way software is developed, however this improvement*

can be affected by the support of the organisation developing the software, and the extent to which the customer is willing to be engaged.

The first sentence justifies the establishment of agile based on issues with traditional software development approaches. This theme emerged through the participants' responses, particularly in relation to the tendency for software engineers to produce too much documentation and not being able to handle requirements volatility using traditional approaches. Key quotes justifying this observation are given below,

> "So the issues of over-documenting, and over-everything has occurred, and there certainly needed a different approach taken. And I guess Agile was at a point in time when people were throwing up their hands and saying 'I don't want to spend five years on a project, when all I'm doing is creating documentation."' (Participant S082210)

> "The motivation behind it is that on the first approximation, nothing works. People saw that the problem with methodologies like waterfall and so on is that, on the one hand clients often change their mind along the way and get better ideas." (Participant S082711)

> "[Agile was created] To overcome a lot of problems we were having with requirements, and to overcome what we'd call 'Requirements Volatility'. Overcome a lack of understanding and lack of communication." (Participant S081912)

These responses indicate a need within the software engineering community to address these issues of over-documenting and handling volatile requirements. From the participants' descriptions, it appears that agile software development was established for this purpose.

The second part of the first hypothesis states that '*These practices emphasise user satisfaction in order to improve the way software is developed*'. User satisfaction was a broad theme, used to encompass categories such as engaging the user through the development process, adhering to a strict timeframe and reducing the amount of documentation that is produced. These themes that encompass a focus on user satisfaction emerged throughout the interviews, as shown below,

> "[I]t's verifying that you're actually doing what they want you to... That's why agile says that you have the user very closely involved." (Participant S082210)

> "Agile done well is far more structured. You've got the beat of the drum, every two weeks, every month, you've got reports that have to be done, you're acknowledging that work has to be finished" (Participant S081912)

> "One of the tenets of the [agile] manifesto is to value working software over documentation." (Participant S082210)

The conversations surrounding these themes led into the consequences of undertaking an agile approach, which was the goal of improving the practices and products produced. The theme of improvements came from the way in which agile practices can handle complexity, and from the focus on quality that came from parts of the community that advocated for agile,

> "It's trying to recognise that as software projects have gotten more complex, and bigger and bigger, the idea of playing out these mega projects, which has always been ridiculous, but even more so, and how we can do best practice." (Participant S081912)

> "There was a group there that was all about quality, you know measuring and collecting information about systems, and where they failed and why they failed. And they were doing the research work, coming up with ideas for ways to overcome." (Participant S082210)

> "I think it works well when your major focus is building features." (Participant S082711)

While these themes were seen as possible improvements to the issues faced by traditional methodologies, they were conditional to various factors that could affect the development team. The final part of the hypothesis states '*however this improvement can be affected by the support of the organisation developing the software, and the extent to which the customer is willing to be engaged*'. The software engineering participants all made note that agile is not a 'fix-all' solution, and that under certain situations it does not in fact improve processes at all (and in some instances, actually makes them worse). The broad theme of organisational support encapsulated such themes as the culture of the organisation, the support the development team has from management to use agile techniques, the attitude of the development team and any lack of understanding the development team had regarding agile techniques. The broad theme of the extent of customer engagement is self-explanatory. Evidence of these factors is given below,

> "You need mature people there who understand the need to celebrate finding errors, and the positive ego-less feeling of where people are. Encouraging people to look at their estimates of how long it's going to take them to do things, in a positive way, in terms of how can we do better rather than beating people into the ground." (Participant S082711)

> "The main reason I've seen them fail is the attitude of the development team. Of really just saying 'we'll just hack away here. We'll create the requirements for the customer, and convince the customer what his or her requirements are, then let's go do it' and they hack something together and then deliver something that really doesn't explain how they've developed it, what's in it, how to change it when it comes to doing that." (Participant S082210)

> "Logically, agile should give you a better means of getting people up the experience curve more rapidly, providing you've got enough more experienced people who have the right attitude, who want to bring people up. And not everybody does." (Participant S082711)

> "There's misunderstanding out there. It's tied to misunderstood rigor, of the methodology." (Participant S081912)

> "If you can't get an involvement [with customers] happening, then the team ends up just talking to itself, and you're back to where you where before, except worse in that you haven't really got any initial requirements to work off." (Participant S082711)

This set of excerpts is presented in order to demonstrate the many factors on which agile development methodologies are dependent in order to be successful. Together with the above evidence, these quotes are used to justify how the researcher determined the first hypothesis within the model above.

The second hypothesis generated from the findings is as follows,

> *People in software engineering have typically stuck to what they have known, leading to an insular view towards interdisciplinary work. The extent to which software engineering will accept interdisciplinary work is dependent on peoples' willingness to adopt new practices.*

The first part of this hypothesis states that software engineers have typically stuck to what they have known, leading to an inward-facing attitude. It has been established throughout this thesis several times that literature supports this position, as did the experts interviewed in this case study. In particular, the themes of being 'silo-based' and the fact that people generally dislike change were expressed by the participants,

> "I think sometimes you may well find that other disciplines, because we exist in our solid, cement wall silos are actually doing a lot of the same sort of thing, we're just calling it something different." (Participant S081912)

> "Getting things embedding into new environments is difficult. In fact, on first approximations, it is impossible." (Participant S082711)

While there were several examples provided by participants where software engineering could hypothetically benefit from taking an interdisciplinary approach, there was clear acknowledgement that this is currently not the case (these examples are discussed in subsection **5.2.5** below). It was acknowledged that in order for interdisciplinary approaches to be adopted within a software engineering context, it would require both management support and a culture of open-mindedness,

> "If people don't want to have something imposed on them, the only way you're going to be able to do that is to get somebody at the top of the organisation to go with it, whatever the top of the organisation is" (Participant S082711)

> "It will only work if people are open-minded in an environment, and are prepared to make it work." (Participant S082711)

Together, these examples set the context for the attitude that the discipline of software engineering has had to date, and thus form the basis for the second hypothesis within the software engineering case.

Finally, within each interview participants were asked to rate the amount of user interaction that they felt occurred during the each stage of the agile lifecycle (the raw forms of these diagrams can be found in Appendix K). These were rated using the general systems model produced in Chapter 3.

The researcher has aggregated the inputs from each participant into a synthesised model, in order to highlight the amount of user interaction believed to occur at each lifecycle. In this newly created model, the darker the green for a particular lifecycle phase, the more user interaction that is believed to occur there. The results of this activity can be seen below in Figure 5.2.

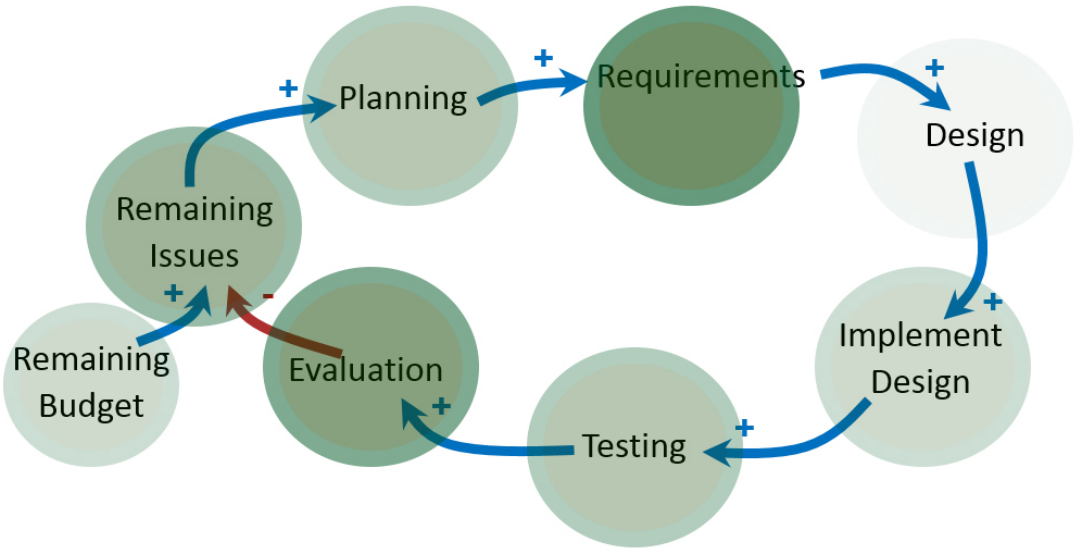


Figure 5.2: Aggregate user interaction within software engineering

From the data collected, it is clear that the participants of the study believed that the most user interaction occurred at the 'Requirements' and 'Evaluation' phases. This will be contrasted to the results from the mental health case, presented below.

The hypothesised model presented in this subsection represents the output from analysing the data from the agile case within this project's case study. The issues identified with traditional methodologies, and the focus on user satisfaction, is consistent with the justification for the inception of agile software development (Fowler & Highsmith 2001). Similarly, the inward facing nature of software engineers corresponds to findings through the literature (Glass et al. 2002; Segal et al. 2005). However, in order to provide the reader with a fair basis from which to judge this study, deviant

and outlying evidence that do not fit within the central interpretation are presented below.

5.2.1.1 Deviant and Outlying cases

Evidence which does not fit within the central interpretation of the software engineering case is presented below. The core deviation derived from the theme of dealing with complexity. While two of the participants felt that there was merit in using agile techniques for dealing with the development of complex systems, one participant did not feel the same way. When asked "Do you think that agile is an effective way for dealing with complexity?" the participant responded,

> "No I don't, not in its rawest form... I know from documents that I've read, people have said that they completed their project, very rarely from what I've seen will people produce a summary of what the system is actually about and the totality if you like of the problem that we're solving. It's always presented in little bits without giving a big picture, and that's often what's missing." (Participant S082210)

While the researcher acknowledges the validity of the above point, because this theme was only one constituent part of a broader theme (Conditionally Approved Practices), it was felt that this single outlying attitude did not undermine the broader theme as a whole.

5.2.2 Major themes through CBPR case

The major themes uncovered during the CBPR case are brought together into the theoretical model found in Figure 5.3. These themes were derived during the selective coding process described above. In this process, themes were drawn from axial codes and the relationships between these codes were specified.

As in the software engineering case, there are two key phenomena identified in Figure 5.3, both indicated by a thicker outline than the other themes in the model. From this model, two broad hypotheses have been generated, as follows,

1. A need to understand community needs led to the establishment of CBPR practices. These practices emphasise engagement and feedback with users in order to fulfil community needs, however this process can be affected by the level of understanding the research team has, the ability to engage with and manage users, and time constraints that are imposed on the project.

2. People in mental health research have typically stayed within a silo-based environment, leading to an insular view towards interdisciplinary work. The extent to which mental health researchers will be open to the idea of interdisciplinary work is dependent on peoples' willingness to adopt new practices.

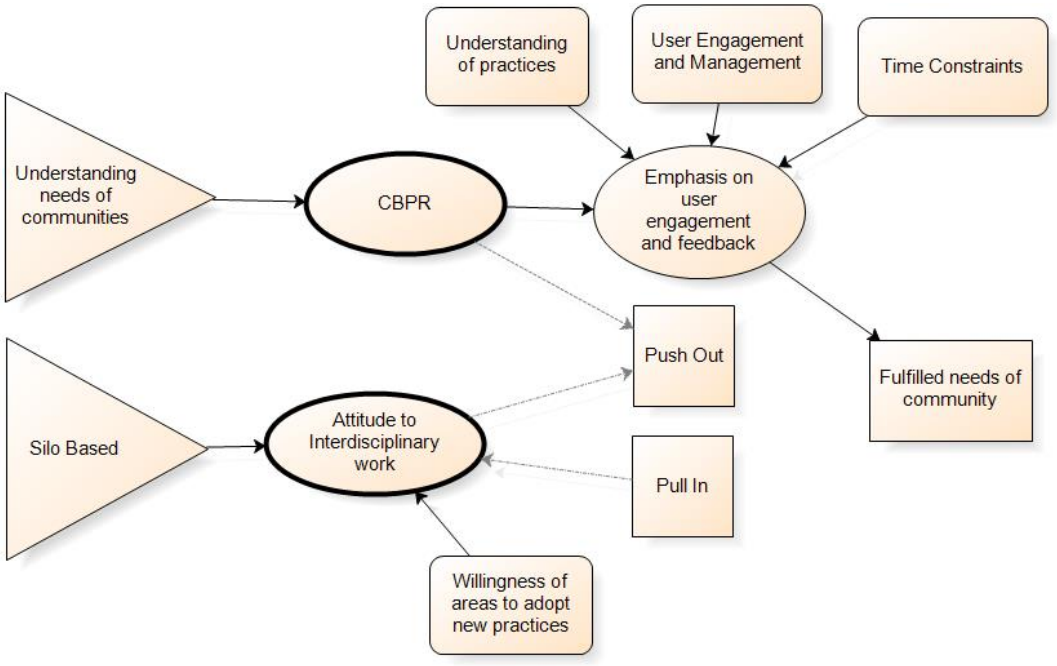


Figure 5.3: Theoretical model of CBPR from case study

Note that, as in the software engineering model, there are themes in the model connected by dashed arrows that are not present in either hypothesis. This is because they do not reflect the current state of mental health, but rather were themes that arose through the interview regarding possible avenues of thought and research for the future. These themes are further discussed in the **Identified areas for further research** subsection below.

In order to justify the above hypotheses, evidence will be provided regarding the relationship between each connected pair of themes. This will involve breaking down each hypothesis into its constituent parts, and providing evidence for each relationship in the model. We will begin with the first hypothesis.

> *A need to understand community needs led to the establishment of CBPR practices. These practices emphasise engagement and feedback with users in order to fulfil community needs, however this process can be affected by the level of understanding the research team has, the ability to engage with and manage users, and time constraints that are imposed on the project.*

The first sentence justifies the establishment of CBPR based on the need for those working in mental health to understand community needs. This theme emerged through the participants' responses, particularly in relation to the need for understanding their users' requirements and the nature of working with communities. These responses were gathered when discussing the foundations of CBPR,

> "In terms of community development... I imagine from research that tries to work with communities to understand what their needs are, and understand what's going to work for them in their context" (Participant M091514)

> "It's not just about organisations coming in and saying 'come and complete this survey'. Particularly with certain communities, it's not going to work... particularly if you are building something that needs to be contextually relevant." (Participant M091514)

These examples set the context for the establishment and use of CBPR in mental health. This leads into the next part of the hypothesis that reads '*These practices emphasise engagement and feedback with users in order to fulfil community needs*'. This part of the hypothesis was created when clear themes regarding user engagement and a strong customer focus emerged from the data. A selection of examples includes the following,

> "At every sort of point there's been end users coming in, rather than just being at the end and helping us test it." (Participant M082615)

> "[T]here is a strong consumer focus in a lot of the work that we do here." (Participant M082615)

> "[In CBPR] Ensuring clear feedback processes iteratively throughout the process." (Participant M091013)

Through this strong user focus, several benefits to end users were identified, broadly put under the theme of 'fulfilled community needs'. Aspects of these fulfilled needs related to themes such as giving users a sense of ownership over the service, dealing with some of the inherent complexity that comes with these sorts of problems and generating creativity within the community regarding the types of services they would want. Examples of these themes include the following,

> "I've found it to be incredibly worthwhile for being able to generate a lot of creativity around ideas. For the first reason, it makes complete logical sense to me as to why you want to engage the people who are going to be using something from a ground level." (Participant M082615)

> "I think in terms of understanding the complexity of the needs of the end users and developing something that more accurately meets those needs, then absolutely." (Participant M091514)

> "It gives them [the community] a sense of ownership over it, and you feel confident in yourself that what you're building and what you're delivering to somebody is going to be somewhere around what they want and what their needs are." (Participant M082615)

However, just as with the software engineering case, there are several factors which can influence the success of CBPR projects. This is highlighted in the final part of the hypothesis '*however this process can be affected by the level of understanding the research team has, the ability to engage with and manage users, and time constraints that are imposed on the project*'. There were several factors which participants indicated can affect the success of CBPR activities. Notable themes that arose include conflicting requirements, engagement and communication with stakeholders, the project team's understanding of CBPR, management support and the issue of time management. Examples of these constraints are given below,

> "And sometimes it can fall down too when you have a group of heterogeneous users, where they don't actually all want the same thing. You might have people involved in the process that are very diverse in what they want. How then, do you synthesise those views with all the other players as well? So that's where I think the challenges can happen." (Participant M082615)

> "I think the biggest trouble is when people don't understand why they're doing it [CBPR]. So they kind of hear about it, or someone tells them that they have to use it, but it doesn't come about because of a fundamental principle that they want to engage people in that process or they believe that it's going to be a better project and a more meaningful outcome" (Participant M091013)

> "And speaking a common language is really difficult too. Like you will be talking in some terms and someone will come to you providing their ideas in a different language, and how do you make sure you're speaking about the same things and you understand each other properly." (Participant M082615)

> "What I think it comes down to is culture and it needs to be driven from the top down in a way. If you've got the organisations that recognise the value of community participation, it's going to set everything up to follow." (Participant M091514)

> "It actually more of a time issue than a budget issue for us. Because we found that participatory design activities, such as this, just take such a long time." (Participant M082615)

This set of excerpts is presented in order to demonstrate the many factors on which CBPR practices are dependent in order to be successful. Together with the above evidence, these quotes are used to justify how the researcher determined the first hypothesis within the model above.

The second hypothesis generated from the findings is as follows,

> *People in mental health research have typically stayed within a silo-based environment, leading to an insular view towards interdisciplinary work. The extent to which mental health researchers will be open to the idea of interdisciplinary work is dependent on peoples' willingness to adopt new practices.*

The first part of this hypothesis states that mental health researchers typically stay in self-contained research silos, leading to an inward-facing attitude. This view was presented by two of the three participants, when they self-identified as existing within what they described as a silo,

> "I'm sure there others, I don't know because I work in Mental Health so I'm not as familiar with external practices... I think it's interesting how silo based a lot of practices are, even at universities" (Participant M091514)

> "I think that's a problem, because you get so bogged down in your own field, that I'm sure a multidisciplinary perspective could be useful, however a lot of people don't have it. It's just like 'this is my slice, and this is what I do'." (Participant M082615)

While there were several examples provided by participants where mental health could hypothetically benefit from taking an interdisciplinary approach, there was acknowledgement that this is currently not the case (these examples are discussed in subsection **5.2.5** below). It was acknowledged that in order for interdisciplinary approaches to be adopted within a mental health context, it would require a culture of open-mindedness. One participant shared their experience with trying to undertake a multidisciplinary project within a university context,

> "My PhD is multidisciplinary, and it seems to be a major issue. It's not a major issue for me, but it seems to be a major issue for academics, and it makes me question, is this a reflection of the wider world?" (Participant M091514)

Together, these examples set the context for the attitude that the discipline of mental health has had to date, and thus form the basis for the second hypothesis within the CBPR case.

Finally, within each interview participants were asked to rate the amount of user interaction that they felt occurred during the each stage of the CBPR lifecycle (the raw forms of these diagrams can be found in Appendix K). These were rated using the general systems model produced in Chapter 3.

The researcher has aggregated the inputs form each participant into a synthesised model, in order to highlight the amount of user interaction believed to occur at each lifecycle. In this newly created model, just as in the software engineering case the darker the green for a particular lifecycle phase, the more user interaction that is believed to occur there. The results of this activity can be seen below in Figure 5.4.

From the data collected, it is clear that the participants of the study believed that the most user interaction occurred at the 'Planning', 'Design' and 'Testing' phases.

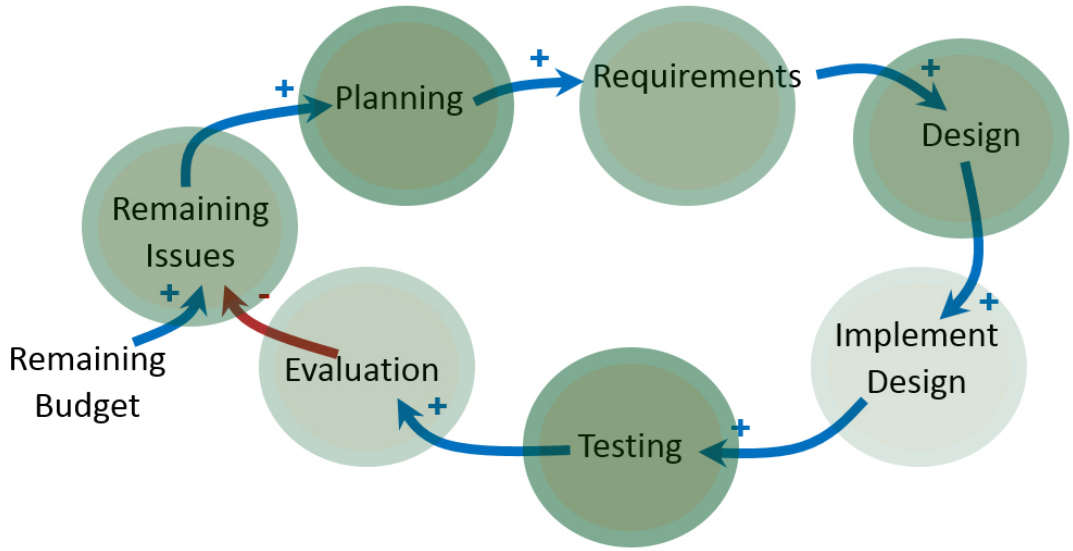


Figure 5.4: Aggregate user interaction within CBPR

This will be contrasted to the results from the software engineering case, presented below.

In order to provide the reader with a fair basis from which to judge this study, deviant and outlying evidence that do not fit within the central interpretation are presented below.

5.2.2.1 Deviant and Outlying cases

Evidence which does not fit within the central interpretation of the CBPR case is presented below. While all participants acknowledged that the discipline of mental health may be able to benefit from adopting interdisciplinary practices, one participant did acknowledge that their workplace was currently collaborating with non-mental health organisations in order to share their techniques regarding CBPR,

> "Yeah, so we're already getting contacted by groups as broad as [removed health organisation]. Lots of other health groups are contacting us to find out how we're doing what we do." (Participant M091013)

This is in contrast to the other two respondents in this regard. Rather than considering this in opposition to the findings, this potentially presents a learning opportunity for software engineering, to observe how other disciplines are sharing information. Areas for further identified research are discussed below.

5.2.3 Cross-case comparison

In this subsection, relevant components from the agile software development and CBPR cases are compared, followed by discussion. In order to perform a cross-case

comparison, parts of Burn's *Cross-Case synthesis and Analysis* (2010) has been followed. The researcher notes that this is not a true cross-case synthesis since both cases have been analysed individually prior to the comparison, however the use of a comparative matrix is used nonetheless for the purposes of generating discussion. The results of this comparison can be seen in Table 5.6. The themes used in this comparative matrix are those that have been presented in the analysis above.

Table 5.6: Comparison of common variables between two cases

	Agile	CBPR
Identified causal conditions	Issues with traditional approaches	Understanding the needs of communities
Intended outcome	Improved Practices	Fulfilled needs of community
Main issues experienced	Lack of understanding; Lack of documentation; Attitude of development team; Extent of user engagement; Culture and management support	Time constraints; Access to stakeholders; Balancing requirements with feasibility; Conflicting requirements; Lack of understanding; Communication and common language
Phases of most engagement	Requirements, Evaluation	Planning, Design, Testing

Firstly, despite the similarities in implementation, these practices appeared to be caused from two very different phenomena (in the opinions of the participants). The software engineering participants explained that agile was born out of issues that were being experienced with traditional approaches to software development (such as a tendency to produce too much documentation and not being able to handle volatile requirements). In contrast, participants from mental health appeared to believe that CBPR was created based on the requirement to fully understand community needs. It is interesting to note the difference in perceptions the two sets of participants had. While both saw that these practices were created out of necessity, one set views it from the perspective of mending something that is broken, while the other sees it as a way to better position themselves for service delivery. After conducting these interviews and analysing the data, it is not clear whether or not these differences stem from the creation of these practices themselves, or perhaps a difference in culture between the two disciplines. Without performing further investigation into this topic, it would be inappropriate to make a judgement call one way or the other.

Secondly, the intended outcomes for each practice seemed to relate to the identified causal condition in both cases. The software engineering participants noted that improved practices (including the possibility for improved quality and being able to compartmentalise the complexity of software engineering) were the intended outcome for using agile over traditional methods. The participants were very clear that these improvements were conditional on the execution of the practices and the culture surrounding the project. This is contrasted to the mental health participants who believed that CBPR was able to fulfil the needs of the community, by handling the complexities that came with their requirements, giving the community a sense of

ownership over the product and aligning their output to what the community actually needed. It is hardly surprising that participants presented the intended outcomes of their relevant practice in relation to the need that caused that practice in the first place.

Thirdly, the main issues experienced had some common crossover, however there seemed to be a difference in overall theme for these issues. The software engineering participants noted that the attitude of the development team, a lack of understanding surrounding agile, organisational culture, management support, a tendency to produce too little documentation and the extent to which users would engage were all major pain points when undertaking agile practices. Many of these themes (such as attitude of development team, culture, management support and lack of understanding around agile) seemed to stem from an organisational culture origin. This is in contrast to the issues felt by the mental health researchers when undertaking CBPR, who noted that their major pain points generally stemmed from time constraints, lack of access to stakeholders, balancing requirements, communication with stakeholders and a lack of understanding surrounding CBPR. The first two issues of time constraints and access to stakeholders stem from a more resource-focussed origin in comparison to the issues felt by the software engineering participants. It is interesting to note that the software engineering participants did not raise time constraints as an issue, but rather as a necessity for operating with in a fixed iterative structure. This may have been due to the fact that while agile specifies fixed iterations (such as SCRUM), which sets a known time boundary for each stage of the project in advance, CBPR does not appear to have an equivalent time-specified framework. However this is speculation from the researcher, and further work should be performed to investigate this phenomenon further. There were also issues that were predominant in both disciplines, most notably a lack of understanding by those undertaking agile and CBPR. This was one theme that consistently appeared across both disciplines, with participants noting this was a significant issue in their field. One software engineering participant eluded to the fact that there was a lack of emphasis towards training in the industry,

> "I think we haven't learnt yet is the fact that you really do need well trained people to think about and develop some of these systems. And it may not be just on the development, I think it is more on the thinking side of things, it's where our software engineering should be concentrating I think." (Participant S082210)

The issue regarding a lack of understanding around the investigated practices may be alleviated if more emphasis is put on software engineers to be sufficiently trained, however further investigation as to the reason behind this lack of understanding may also be beneficial to the two disciplines.

Finally, the phases of the lifecycle that engaged users the most differed between the two disciplines. Software engineering participants believed that there was an emphasis on user engagement at the Requirements and Evaluation stages within agile,

> "They'd definitely be involved in the requirements and planning phases." (Participant S081912)

> "At the beginning of each iteration the customer should agree on what the requirements will be for that iteration based on the remaining issues" (Participant S082210)

This is in contrast to the mental health participants who believed that the most user engagement happened during the Planning, Design and Testing phases of the lifecycle (the evidence for which can be seen in Appendix K). While it is interesting to note that the stages of most engagement are mutually exclusive between the two disciplines, there does not appear to be any clear relationship between the stages of user engagement and the issues felt by each discipline.

While some interesting comparisons can be made when comparing the results of the two cases, there are certainly aspects that appear to be related to the culture of the disciplines themselves rather than to the specific practices within agile or CBPR. However some areas for possible translation can be identified from these findings, for example the way in which agile treats the constraint of time compared to the issues that faced mental health researchers regarding time. Another opportunity for learning could come from the issue regarding a lack of documentation in agile teams (which was not identified as an issue for CBPR teams). One mental health participant noted that when undertaking CBPR, the documentation they produce is not only for the benefit of the team, but also for dispersal throughout the mental health community,

> "Documentation. That's the other thing too. We are about knowledge translation and sharing. So the only reason that we know how we can do community-based participatory research in a good way is because we've read other peoples' published work on how they've done it... everything is a research opportunity. Everything is a piece of research that we do. So we can put it out there, and share it, and disseminate it." (Participant M082615)

This attitude to documentation may well prove beneficial to those undertaking agile software development, whereas at the moment this still appears to be an issue.

Following this cross-case comparison, an analysis of the effectiveness and limitations of the systems model is presented below. This analysis is based on the responses given by participants, largely through annotations on the model provided to them.

5.2.4 Analysis of Systems Model

Each participant was provided with the general systems model that was derived in Chapter 3. Along with using this model to generate the findings above, participants were also queried on the applicability of the general model to their experiences in the field. Participants were asked the following question (or a close adaptation) "Have a look at the simple [agile/CBPR] diagram. From your experience, is there anything you would add or change (can be in your own notation)?" The responses to this question were largely that it was representative of the participants' given practice, however there were two recommended alterations that repeatedly emerged.

Firstly, the theme of 'loops within loops' and back links within implementation were common across participants from both disciplines. From the annotated models in Appendix K, it can be seen that two participants from software engineering and two participants from mental health draw arrows to create smaller loops within the diagram. This theme is further supported by some of the comments received from participants,

> "I think there are lots of relationships between parts, and therefore I would say loops within loops. It isn't a linear process, or circular, it's chaos in here." (Participant S081912)

> "[I]f this is your overarching loop [the systems model], then you might see things kind of like that again [draws arrow from Design to Requirements] and like [draws arrow from Requirements to Planning]. Sort of mini or micro loops happening within a larger loop." (Participant M082615)

> "There are microcosms of that diagram within each of the processes." (Participant M091013)

While the annotations are not the same in every case, one thing was made clear from the responses- in order to accurately represent the way these practices are undertaken in industry, inter-process connections were necessary. The term 'chaos' was used, and the sentiment expressed by participants is represented quite succinctly by one software engineering participant,

> "There's a nice term, that Agile needs to be 'chaordic', in that there has sufficient order not to be chaotic, but sufficient chaos to allow innovation and change to take place." (Participant S081912)

Therefore, for the next iteration of development for the systems model, it is recommended that inter-process links are used across the model (and potentially sending loops from each node to itself, to represent the microloops recognised at each stage).

The second theme that arose, particularly for the mental health participants (although not exclusively), was the idea that time and organisational support were seen as additional constraints to the system as a whole, and that budget was potentially a lesser constraint than originally thought. Two of the three mental health participants noted 'Time' next to the 'Remaining Budget' node, and the other noted 'Staff/organisation support' as a constraint to the system. This is further reinforced by the following excerpts, where one example of time being a restriction is actually from one of the software engineering participants,

> "It actually more of a time issue than a budget issue for us. Because we found that participatory design activities, such as this, just take such a long time." (Participant M082615)

> "Budget is one thing, but time is so often, maybe not 'more' important, but definitely the big factor drives things." (Participant M091013)

> "In traditional, you can justify slipping one month (you'll just make it up in the next). [Compared to agile] You haven't got that tight look on what's going on, you're not planning every month." (Participant S081912)

> "I think a lot of it has to do with the culture of the organisation. If you don't have a commitment throughout the organisation, such as given sufficient budget to support certain activities, understanding the cycle of what it is, understanding what the challenges are, and putting in practical supports to minimise any challenges." (Participant M091514)

Based on these responses, it is reasonable to consider adding the constraints of 'Time Restrictions' and 'Organisational Support' to the systems diagram, particularly given that these constraints are not contained to either discipline. We note that upon reflection, the original decision to omit 'Time Restrictions' (as stated in Chapter 3) is no longer seen as the best decision, and would be recommended to be added into the next iteration of the model.

It is also noted that while all effort was taken to explain any confusions or questions that participants had about the model, it is still possible that opportunities for misunderstandings existed. These opportunities existed because of any pre-conceived definition that participants may have had with respect to the names of phases chosen by the researcher (such as 'Requirements', 'Design' etc.) that conflicted with the intended meaning of the phase. While this represents a limitation of the methodology used, this issue was not raised by any participant during the interviews.

While improvements to the model have been identified throughout this analysis, the strength of the model is also highlighted. Ultimately, the model was effective for evaluating the amount of user engagement seen in practice by the two disciplines, and also proved effective as a tool for participants when explaining where major issues occurred when undertaking agile/CBPR (as evidenced by the examples given in the previous 3 subsections). For the purposes intended, the model was an effective tool for gathering information and comparing the two similar practices. Where improvements have been recognised, these are addressed in the following Chapter as future work.

5.2.4.1 Deviant and Outlying cases

There were no cases that directly conflicted with the interpretation presented in this subsection, however there was that was an outlier regarding the attitude towards budget within the mental health participants. While two participants rated time as equally, or more, important constraint than budget, one mental health participant identified budget as a highly restrictive constraint when undertaking CBPR. This participant explained that in their experience, budget was the key determinant for how many iterations were able to be performed, and thus was a very appropriate constraint to have on the system,

> "Perhaps it's the case that it may be quite a while before it goes around in another iteration. So you may get funding to do this work, and you

> go through the stages and evaluate it, and then there's actually no more money after that... It's actually new budget that comes in at that stage, but that might just be my experience." (Participant M091514)

This does not conflict with the results found above, but merely re-iterates the need to keep 'Remaining Budget' on the systems model (and indeed, validates its presence for mental health research).

5.2.5 Identified areas for further research

Several areas for further interdisciplinary research were identified by participants from both software engineering and mental health. As this research project is concerned with interdisciplinary approaches to (and from) software engineering, these sets of responses were considered particularly important. For this reason this subsection summarises these responses, identifying practices which may be able to be adopted from the given discipline to an external discipline (under the code 'Push Out') or external practices that may be able to be adopted within the given discipline (under the code 'Pull in'). Responses from mental health participants are also included for two main purposes. Firstly, the premise for this project (as established in Chapter 1) was that by undertaking interdisciplinary work, learning opportunities should be available for any discipline with which we collaborate. These responses constitute a learning opportunity for mental health researchers. Secondly, by investigating areas that mental health experts believe they may be able to collaborate with, potential areas that software engineering may be able to collaborate with might also emerge. Therefore this subsection covers responses from both sets of participants.

All software engineering respondents felt that there were practices within software engineering that external disciplines could potentially benefit from. The question that relates to this topic asked "In your opinion, do you think other disciplines could benefit from using generalised, adapted practices and techniques from the Software Engineering domain?" All participants answered yes to this question,

> "As a broad generalisation, yes. It depends on what they're doing and where they're doing it. If what they're doing is a project, and almost anything and everything we do can be seen as a project." (Participant S081912)

> "I do. Especially, from what I'm finding at the moment, is the conceptualisation of problems." (Participant S082210)

> "Yes, I think they could. They obviously have to be projects." (Participant S082711)

A clear theme regarding project work was evident in the participants' answers. This immediately gives context to the type of areas that software engineering should be looking to collaborate with. Communication was another theme that was evident through the participants' answers, with two candidates identifying communication techniques as an area that software engineering may be able to contribute to external disciplines,

> "If you capture a lot of this stuff just by using these techniques [referring to Modelling] you can easily translate what you've collected and transcribed into a full requirements model... it's all about simplification of concepts, and getting customers interested, and that applies not just to software, but I've found that the software industry is ahead of the game in being able to provide those conceptual models" (Participant S082210)

> "[T]here may be some of the concepts of communication and collaboration, because that's core to almost anything we do" (Participant S081912)

As participant S082210's answer highlights, modelling was a topic that emerged throughout the interview with respect to communication tools within software engineering. Further comments from within this interview identified that these tools were particularly effective in the participants' experience for talking to customers and helping them understand and conceptualise their problems in real world software projects. Along with communication, identification and engagement with stakeholders were also areas identified as appropriate for pushing out to other industries. The participant notes that explicit way in which software engineers identify and manage stakeholders, particularly external stakeholders. This is identified as something that is currently deficient with the way education programs are created, and an area that software engineering may be able to provide benefit,

> "So for people in a history degree, where are you going to see your graduates going, what do they need? That to me has got to come from somewhere external. You do have external constraints that come from the broad stakeholder of society." (Participant S081912)

Another area that emerged as interdisciplinary applicability was agile software development (the researcher notes the potential bias that the software engineering participants may have had to agile after previously discussing it in the same interview, see section **5.4** below). Areas that agile were deemed to have potential in helping were projects with several stakeholders and potentially volatile requirements, including electronics and large scale infrastructure,

> "Areas like electronics, or large scale infrastructure, where people get stuck with their first idea, and by the time it's been implemented the technology's moved on and it's no longer optimal. You'd have to assume that in that situation it [agile] would be good." (Participant S082711)

Discussions regarding the interdisciplinary applicability of agile were had with two participants, and it was noted that careful consideration would need to be made for any project that wished to introduce agile, including the cultural and management support for introducing such practices. Along with providing possible areas for pushing out software engineering practices, the participants also considered practices from external disciplines that may be adopted within software engineering itself.

All software engineering respondents felt that software engineering could benefit from adopting generalised, adapted practices that originate within other disciplines. The question asked regarding adoptions of external practices was "In your opinion, do you think that Software Engineering could benefit from using generalised, adapted practices and techniques from other disciplines?" One respondent noted the opportunity for two-way share knowledge across interdisciplinary boundaries,

> "I think there could be learning, because they've worked out that in certain instances, this doesn't work because this happens. I'm positive that we could learn as much from them as they could learn from us." (Participant S081912)

Automated and model driven paradigms were common responses to this question. Two of the three software engineering respondents noted that that learning form disciplines that implement standardisation and automation across could teach something to software engineering. Key quotes from these responses are provided below,

> "I suppose the push towards automation that I've seen it in other manufacturing industries, when you look at the history of the production of cars for example... you look at that and think of how things have become totally automated, such as car production, and you look at the quality of the vehicles now. I mean you can basically choose any car and be happy with its quality these days" (Participant S082210)

> "I still think that architects in the building environment, and all the things that flow on from that, I think we still have a lot to learn from that area. That's an area that's been model driven, lots of existing standards. They've been lucky that they have a standardised brick for a few hundred years, so you don't have to go off and build new moulds for that." (Participant S082711)

Along with automation and standardisation, another area that was identified for potential learning was that of scientific approaches to developing and evaluating problems. The software engineering participant identified the willingness for software engineers to get to the market as soon as possible without fully understanding the problem or solution space. The participant recommended that by adopting a more rigorous, scientific approach to software engineering we may improve our understanding on the impact that the software is having,

> "[W]e do tend to perhaps advance too quickly in guessing ideas out there and don't realise the impact of the ideas until something bad happens...So if it was anything, it would probably be putting more scientific approaches into actually how we develop the problems that we want to solve. That would be something that I think we should be taking on more strongly." (Participant S082210)

These responses demonstrate that the software engineering experts interviewed during this case study support the notion of bringing in practices and lessons from other disciplines, in order to improve practices within software engineering. Furthermore, there were no outlying or deviant cases to the examples above for these questions, further strengthening support for interdisciplinary research to be performed within software engineering. See **Implications for Interdisciplinary work within Software Engineering** below for further discussion regarding these findings. A summary of these findings can be seen in Table 5.7.

Table 5.7: Identified areas for interdisciplinary collaboration within software engineering

Push Out	Pull In
Communication and modelling	Automation and standardisation
Stakeholder management	Scientific approaches
Agile techniques	

Like their software engineering counterparts, all mental health respondents felt that there were practices within mental health and mental health research that external disciplines could potentially benefit from. The question that relates to this topic asked "In your opinion, do you think other disciplines could benefit from using generalised, adapted practices and techniques from the Mental Health domain?" All participants answered yes to this question,

> "Yeah, so we're already getting contacted... Lots of other health groups are contacting us to find out how we're doing what we do." (Participant M091013)

> "I do think CBPR, absolutely." (Participant M091514)

> "If there was one thing that I would say that we do, that other disciplines should do, is make sure that what you're doing has a strong research backing. So making sure that it is based on evidence-based principles" (Participant M082615)

Each participant expanded on their responses, identifying areas that they believed could benefit by adopting practices that are currently being performed within the discipline of mental health. A common theme that came from the participants was that of health groups. Two participants identified that other health disciplines may benefit from considering the use of CBPR practices within their domains. Along with the response from Participant M091013 above, Participant M091514 further expanded upon this point,

> "I think more broadly in health, it's going to be relevant for any intervention. Any new product, or any new program, it's just a bit of common sense around how you develop something.... in terms of implementing stuff, in community or organisational settings, and in a wide range of settings, I would think it's really relevant." (Participant M091514)

Health was not the only area mentioned by participants. Another industry that has shown interest in CBPR is that of marketing, with participant M091013 noting that market researchers had actually been in contact with the organisation affiliated with this participant in order to learn more about genuine user engagement,

> "I'm always impressed when it seems like we're ahead of them [market researchers] for once... I think there's a lot more acknowledgement about the importance of genuine engagement. Market researchers have always been very good at going out and hearing from enough people in order to provide the minimum thing that they need in order to make some more money. But, I guess as you get into more and more complex problems, government and corporates have seen that the simple solutions don't fix the problem." (Participant M091013)

The above comment was made in relation to the interest market researchers had shown in CBPR. Along with industries that were identified as possible beneficiaries of collaboration with the discipline of mental health, certain practices were also identified as appropriate candidates for generalisation and translation into other disciplines. Firstly, scientific and evidence based approaches to tackling problems and evaluating solutions were identified by one participant as being particularly effective within the domain of mental health. The participant believed that other disciplines could benefit by following a similar mindset to the establishment of practices and evaluation,

> "And if there was one thing that I would say that we do, that other disciplines should do, is make sure that what you're doing has a strong research backing. So making sure that it is based on evidence-based principles. That you also have a proper, rigorous scientific way of evaluating it... I think this is particularly important for technology because there is so much junk out there, where you really want to identify what's good and what's high-quality. That would be the main thing." (Participant M082615)

The participant makes note that the field of technology in particular could benefit from adopting a scientific approach. This is contrasted to Participant S082210's response above to the fact that software engineering may be able to benefit from using a more rigorous and scientific based approach to development. This observation is further discussed in the following section.

The final practice presented is the use of peer support programs that are prevalent throughout mental health. One participant in particular noted the strength of these programs in the discipline of mental health, and believed that other disciplines could benefit from adopting such approaches,

> "I think Mental Health is really great at peer support, and I don't think that necessarily happens in other domains or other fields.... I think it's a really strong and a really great thing. Particularly when you look at addiction in particular. I think that's a really great thing that you could imagine being used in other fields." (Participant M091514)

While the participant spoke about the strengths of peer support, no specific recipient discipline was specified in the response. Along with providing possible areas for pushing out mental health practices, the participants also considered practices from external disciplines that may be adopted within mental health itself.

All mental health respondents felt that their discipline could benefit from adopting generalised, adapted practices that originate within other disciplines, however the most common responses revolved around a lack of awareness of what other practices were actually doing. The question asked regarding adoptions of external practices was "In your opinion, do you think that Mental Health Research could benefit from using generalised, adapted practices and techniques from other disciplines?" One respondent explains that they see interdisciplinary perspectives as effective, while the sentiment provided by another captures the attitude of all three respondents,

> "Well, yes I would say. I think it's always going to benefit to look at what other disciplines and areas are doing, absolutely." (Participant M091514)

> "I assume there's probably a bunch of stuff that could be learnt from other disciplines that I'm just not across. I'm just really at the limits of my knowledge." (Participant M082615)

Despite a general lack of awareness identified by each participant, one respondent did feel that interdisciplinary work had made an impact on their work. This participant had used technology in the mental health space, and felt that this had positively influenced the way they approached health promotion,

> "Personally in my own practice, looking at technology, I think technology has been a big influence in techniques that they use that can be incorporated in health promotion." (Participant M091514)

The participant spoke positively about technology's influence in their work, and like all mental health participants, demonstrated an open-minded attitude to research and possible adoption of interdisciplinary practices within their domain. These responses demonstrate that just like the software engineering experts, the mental health experts interviewed during this case study support the notion of bringing in practices and lessons from other disciplines. Furthermore, there were no outlying or deviant cases to the examples above for these questions, further strengthening support for interdisciplinary research to be performed within software engineering. A summary of these findings can be seen in Table 5.8.

This subsection concludes with a discussion based on the observations of the researcher. Each participant interviewed from both software engineering and mental health appeared to be open minded to the idea of interdisciplinary collaboration, particularly for pushing out practices from within their own domain. This attitude to interdisciplinary research was somewhat surprising, considering both sets of participants identified the silo-based nature of their discipline. Furthermore, software engineering has repeatedly been found to be insular across various research projects (Glass

Table 5.8: Identified areas for interdisciplinary collaboration within mental health

Push Out	Pull In
Health Services	Technology
Genuine Engagement	
Scientific Approaches	
Peer Support	

et al. 2002; Segal et al. 2005), so the fact that all participants supported the notion of both pulling in and pushing out practices was an unexpected outcome.

Secondly, the researcher notes the observation that while all participants supported interdisciplinary collaboration, it seemed considerably easier for participants to think of practices to push out *from* their disciplines when compared to bringing in new practices *to* their disciplines. This observation was made on the length of responses for each questions, the time it took each participant to answer each question, and the general body language displayed by the candidates while answering these questions. This may seem intuitively obvious, that participants are more comfortable discussing practices in which they are familiar, however this observation is still given without judgement from the researcher.

The findings presented in this section lead into the following section's discussion regarding the implications for interdisciplinary work within the discipline of software engineering.

5.3 Implications for Interdisciplinary work within Software Engineering

There are several implications for further interdisciplinary work that were identified based on the findings presented above. First and foremost, probably the most unexpected result of the study was the apparent openness to interdisciplinary practices displayed by all software engineering participants. This struck the researcher as an interesting phenomenon- if the literature is telling us that software engineering is inward facing (and indeed, the literature itself is insular in general), then how likely is it that three experts from within the discipline would all be so open to the prospect of interdisciplinary work? This leads one to wonder, is it possible that while interdisciplinary work is not being published within the literature, perhaps software engineers are more open minded than initially perceived? These questions are important for the future prospects of interdisciplinary collaboration within software engineering. This leads to the first implication for interdisciplinary work, investigating the attitudes to interdisciplinary collaboration within software engineering itself. Through this type of investigation, a baseline could be established from which future interdisciplinary work could move forward from. This investigation could also explore the historical tendency for software engineering to be inward-facing and silo based, and look for root causes for how the discipline came to be this way. This investigation forms the

first, and arguably most important, implication for the findings presented within this study. Along with this implication, there are also further implications regarding the findings for the agile and CBPR comparison.

It was demonstrated through the findings of this case study that both agile and CBPR practitioners can suffer from a lack of understanding regarding their respective practices. These were seen to be as a detriment to undertaking these sets of practices, and as such present an opportunity for further research into firstly investigating how this has been allowed to develop within both disciplines, and secondly ways in which these disciplines may be able to mitigate these issues. While a study of this nature does not necessarily have to be interdisciplinary (that is, each discipline could investigate separately), by utilising some of the techniques demonstrated in this research project, it may be possible to investigate this phenomenon within the context of both disciplines. This has the two-fold benefit of being able to combine resources in order to investigate this issue, but may also further each discipline's exposure to and understanding of interdisciplinary practices. Along with practitioners suffering a lack of understanding, issues surrounding time constraints when undertaking agile and CBPR were also raised.

An interesting outcome of the case study was the prevalence of 'time constraints' amongst the list of major issues seen in CBPR, when it wasn't raised as an issue by the software engineering participants (indeed, in some instances fixed iterations were spoken about as a necessity in certain agile practices). Further interdisciplinary research into this phenomenon could include the investigation of how agile practices deal with the time constraints put onto them, and seeing if these can be translated into a CBPR environment. To date, no work has been published in this area, and thus presents a new research opportunity based off these findings.

Areas for potential knowledge transfer were also identified throughout the case study. It was noted by one of the software engineering participants that they believed software engineering could benefit from adopting a more scientific approach to problem development and evaluation. This is in contrast to the belief of one of the mental health participants, who believed part of the strength from their discipline came from the scientific research backing for development and evaluation. This represents an opportunity for further research, regarding how software engineering could potentially learn from mental health research in this scientifically focused area.

Similarly, a participant from the discipline of mental health discussed the issues experienced regarding communication with their users, and the difficulties in trying to use a common language. Two participants noted the communication tools used within software engineering as a potential candidate for pushing out to other disciplines. In particular, the use of modelling was emphasised, where the strength in simplicity of these techniques were highlighted. This represents an opportunity for further research, regarding how software engineering may be able to provide benefit to mental health research through the translation of modelling techniques.

The findings in this interdisciplinary case study uncovered some unexpected and interesting themes. While the focus was on a collaborative approach to software engineering and mental health research, the methods used (such as the approach to sys-

tems modelling discussed in Chapter 3 and the multidisciplinary case study) are not restricted to these disciplines. Indeed, these findings have shown that both the software engineer participants and mental health participants were open minded when it came to interdisciplinary work, yet identified as generally operating in a silo-based nature. These findings may well translate into other disciplines as well, and if so, present an opportunity to discover the attitudes toward interdisciplinary work in other domains. Therefore the methods used may prove useful to disciplines outside of software engineering and mental health.

Along with this discussion regarding the implications for future interdisciplinary work stemming from the findings of this study, a discussion regarding how this research fits amongst current software engineering literature is provided below.

5.3.1 How this work fits into existing research

The work presented in this thesis sits loosely amongst the works of McGrath & Uden (2000) and Conradi et al. (2000) due to the interdisciplinary approach to software engineering practices. Aside from a few notable mentions in Chapter 2 of this thesis, there has been very little interdisciplinary work performed to date within software engineering. The following model presents an updated version of the snapshot of work in this area (with annotated version in Appendix B), with one key alteration. The work of this thesis has been placed amongst the existing research, to show the reader exactly how this project fits in with the wider context of software engineering research. The two-way flow of knowledge that has been explored through this project is noted by the double ended arrow in the model.

Where this work stands out amongst existing research is in exploring the feasibility of transferring knowledge in order to improve practices within software engineering. In this way, this research project stands in isolation, however it is the hope of this researcher that further interdisciplinary work within software engineering can be performed based on the foundation that has been presented in this thesis.

5.4 Strengths and Limitations of Research

The strengths and limitations of this project's approach to research is presented in order to demonstrate transparency in the presentation of results. This section begins by identifying the strengths of the approaches adopted, along with their implementation within this project. Following the strengths, the limitations of the project are presented in order to paint a fair picture of the project for the reader. This section concludes with a discussion regarding the influence that the researcher may have had over the case study data.

The strengths of this project lie in the qualitative nature of the data gathering process. The interviews that were held throughout the case study were flexible, and allowing for unexpected themes to emerge and be discussed. As a result, the data collected throughout this project was rich with information, and reflected the themes that the participants felt were relevant for each discussed topic. The result of holding

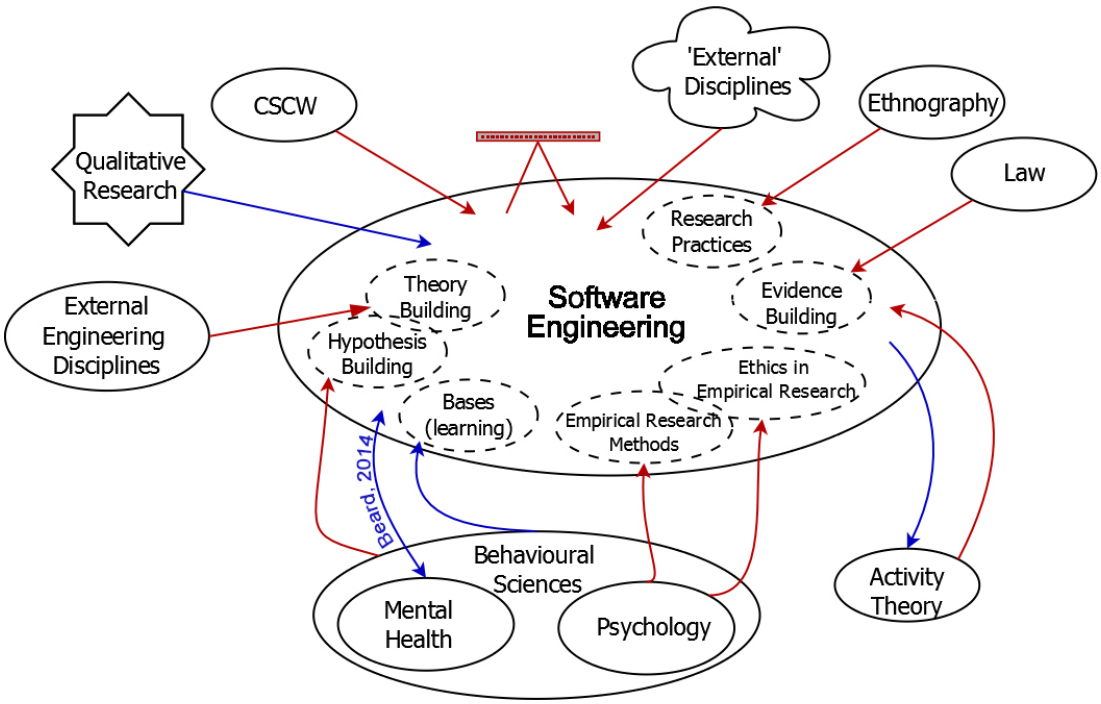


Figure 5.5: Where this work fits in with existing research

these sorts of interviews meant that topics (such as the constraints that time impose onto CBPR projects) are able to be fully explored, when they weren't a part of the interview protocol to begin with.

Additional strengths for this project include the fact that multiple data sources were used in each case, allowing for data source triangulation. Furthermore, each interview was audio recorded, transcribed (or partially transcribed) and returned to each participant for verification. This process further strengthened the validity of the data, as the text that was coded and analysed was based off documents that were approved by each participant. However, while this project had notable strengths, there were also limitations to the case study, as discussed below.

The first notable limitation of the project was the constraint on the number of participants that could be interviewed due to the time limit of the project. Ideally, at least one more round of interviews would have been performed in order to further investigate some of the phenomena identified above, however in the time frame that the project was operating within meant that this was infeasible. The researcher acknowledges that 3 people for each case is a small number, and had time permitted, would have made this sample pool larger. This limitation also refers to the fact that participants were not able to review the data analysis prior to submission, which would have increased the validity of the findings.

The second notable limitation of the project is that the quality of the data analysis was dependent on the skill of the researcher. While all effort possible was taken to reduce the bias introduced by the researcher, the results were still dependent on the

analytical research ability of the researcher. This is not to say that the researcher is not competent, but rather that this is simply a limitation of undertaking this type of qualitative research project.

Another limitation comes from the selection of participants for the study. Participants were sought by using contacts from the advisors of this project. While each participant was an expert in their area and generally had a different background to other participants from the same discipline, possible bias may have been introduced due to the fact that participants were not randomly selected from all experts, but rather from a subset that were connected to people involved with the project. This may have made participants predisposed to try and give the answers they felt the researcher wanted to hear. In order to address this, the researcher made all efforts possible to make participants feel comfortable, and asked them to give their personal opinions, regardless of what it was, to all questions.

There was a small chance that bias may have been introduced into the study because the selection of participants were all experts (that is, they may not have fully reflected the responses of all people in the relevant disciplines). However this bias was balanced with the fact that due to the broad range of experts interviewed, it was felt that they would be more likely to understand the current phenomena occurring in their relevant fields. Nonetheless, there is a small chance this could have introduced elite bias (that is, the risk of giving greater weight to high status or more articulate participants (Kohn, 1997)) into the project.

Finally, the order in which questions were asked also may have skewed the data when it came to discussing interdisciplinary practices. When asked if there were any practices that participants felt could be generalised and transferred into other disciplines, many began their responses referring specifically to agile software development or CBPR (whichever was relevant for their field). While it is possible that they may have jumped to these topics of their own accord, there is a considerable likelihood that the fact these practices had just been discussed through the interview may have skewed their answers. In order to combat this, the researcher tried to encourage participants to think of other areas in which practices may be adopted from within their discipline.

Along with the strengths and weaknesses of this project presented here, the influence that the researcher may have had on the data is also considered, as discussed below.

5.4.1 Influence of the researcher on the data

Along with the strengths and weaknesses of the project presented above, consideration of how the researcher may have introduced bias into the data is also included for the sake of transparency and completion. There are three areas addressed in which the researcher may have had influence over the data.

Firstly, due to the fact that the researcher was present during the interviews, the presence of the researcher may have actually affected the responses given by participants (similar to the concern above regarding selection of candidates). As mentioned

above the researcher made all efforts possible to make participants feel comfortable, and asked them to give their personal opinions, regardless of whether or not it was what they thought the researcher wanted to hear.

The second identified area that bias may have been introduced is the researcher's personal bias during data analysis (i.e. finding the phenomena that the researcher wanted to find in the first place). While all efforts were made to go into the data analysis phase with an open mind, by following the grounded theory approach to data analysis, the researcher tried to remove as much personal influence over the data during the analysis phase as possible. The researcher followed the rigorous framework given by Runeson & Host (2008), along with supplementary instruction from Robson (2002), in order to mitigate this influence.

Finally, as Robson (2002) notes, first impressions of the data can leave a lasting impression on the researcher. Again, the potential bias that may have formed during the interview phase was actively addressed by following the rigorous grounded theory approach to analysis described in this chapter.

5.4.2 Concluding remarks

While several strengths and weaknesses about this project have been addressed, the researcher leaves judgement regarding the validity of the study to the reader. As much effort as possible was taken when completing this research, however as with any human-based activity, can never be completely void of bias and personal influence. While some aspects of this study may have been able to be quantified, the researcher maintains the position that a qualitative approach to data gathering and analysis was the most appropriate choice for this type of feasibility establishing investigation.

Chapter 6

Conclusion and Future Work

The primary objective of this project was to address the historically inward-facing nature of software engineering and to identify the feasibility of sharing knowledge between software engineering and other disciplines. The discipline of mental health research was identified as a potential candidate from which to learn, in particular the practice of CBPR, which was recognised as being similar in nature to agile software development. A general systems model that represented both agile and CBPR was developed and used in an exploratory case study. The purpose of this case study was to identify differences in implementation across these practices, generate hypotheses regarding the implementation of these practices, and identify possible opportunities for knowledge sharing between the disciplines. This case study involved interviews with experts from software engineering and mental health research, and the data from these interviews were analysed using a grounded theory approach to qualitative analysis. Hypothetical models of agile and CBPR were created based on the analysis of the case study data. Notable findings from this analysis are presented below.

The primary contributions for this project are made through a comparative investigation of agile and CBPR practices; the generation of an informal systems model through an original modelling approach; the implementation of a case study across software engineering and mental health research; analysis of the collected case study data; and hypotheses generated from these case studies. This study represents the first of its kind with respect to the investigation of interdisciplinary practices towards software engineering development techniques. It sits amongst the works of Sim et al. (2001) and McGrath & Uden (2000) with regards to adopting an interdisciplinary perspective within software engineering, however distinguishes itself by focusing on the practices surrounding the development and maintenance of software rather than on research techniques within software engineering.

6.1 Project Findings

The major findings for this project have implications for future interdisciplinary work within software engineering. These findings highlight an open-mindedness to interdisciplinary work that is not currently reflected in within software engineering literature, and present opportunities for further research into this area. A summary of

notable findings are given below:

- All software engineering and mental health participants of the case study identified that they believed their discipline could benefit from adopting adapted practices from external disciplines, as well as provide benefit to other disciplines by sharing practices from within their own domain.
- The responses from participants indicated that the major issues for software engineers practicing agile techniques appear to stem from cultural issues, whereas the major issues for mental health participants appear to be more resource-oriented.
- A major issue felt by both sets of participants was a lack of understanding about the investigated practice (agile or CBPR) within their industry.
- Possible opportunities for software engineering and mental health to share knowledge were identified, particularly across the topics regarding approaches to development and evaluation, tools for communication with users and the way in which time constraints are handled.
- The software engineering participants felt that when undertaking agile, the most engagement for users occurred at the requirements and evaluation phases. This is in contrast to the mental health participants who believed that the most engagement for users when undertaking CBPR practices was at the planning, design and testing phases.

Along with the presentation of the case study findings, each research question for the case study is addressed, and results are presented where appropriate.

- What are the main challenges faced in practice when undertaking user-focused, iterative methodologies?

Common issues that occurred in both agile and CBPR included a lack of understanding by practitioners, as well as the need for cultural and management support. Major issues identified by software engineering participants regarding agile software development projects included the attitude of the development team, teams producing a lack of documentation and the extent to which users could be engaged during the development process. Major issues identified by mental health participants regarding CBPR included time constraints, access to stakeholders and managing conflicting requirements.

- Can an effective, general model that represents Agile and CBPR methodologies be created and used to communicate knowledge?

The model that was developed throughout this project was deemed to be effective for this purpose, however recommendations were made on areas for improvement for such a model.

- Are there relationships between user engagement at particular lifecycle stages and issues experienced when implementing user-based, iterative practices?

Results were not conclusive regarding the relationship between user engagements at certain stages and identified issues by participants, however there did not appear to be a link between the two areas.

- What is the general attitude towards interdisciplinary knowledge sharing in the software engineering and Mental Health Research domains?

The attitudes displayed by participants in both disciplines were that they seemed open minded to the prospect of interdisciplinary collaboration. However participants from both disciplines identified the silo-based nature of their disciplines, and that they felt that their industries were generally inward facing.

- Which practices may be able to be transferred between mental health research and software engineering?

Identified practices that are currently present within software engineering that may be of benefit to mental health research include modelling techniques for communication purposes and practices for handling time constraints. Identified practices that are currently present within mental health research that may be of benefit to software engineering research include scientific approaches to development and evaluation.

6.2 Future Research Directions

There are several avenues for future work that can be derived from an exploratory study such as this. Firstly the existing attitudes towards interdisciplinary collaboration within software engineering must be investigated in order to understand why the discipline has been so historically inward focussed. This information can lead to an answer about what action can be taken in order to promote a more interdisciplinary and collaborative perspective within the software engineering discipline.

Secondly, several areas for further interdisciplinary research between software engineering and mental health have been identified through the findings of the case study. Future studies could include the feasibility of software engineering adopting scientific approaches to development and evaluation, as led by mental health research. Conversely, research could be undertaken investigating the adoption of modelling techniques as well as practices for handling time constraints from software engineering within the discipline of mental health research. These were areas identified by participants of the case study, however as seen in Chapter 5, several other areas for interdisciplinary collaboration were also identified by participants, and these too present potential research opportunities.

Thirdly, the hypothetical models created throughout the case study could be verified or refuted based on further investigation. If found to be an accurate representation of the way agile or CBPR are currently operating, these could help form part of a theory surrounding these practices.

Fourthly, there exists an opportunity to further refine the general systems model created in Chapter 3 of this thesis. Notable improvements that were recommended by case study participants included the addition of back links and inter-process loops, as well as adding time and organisational support as constraints on the system.

Finally, there exists an opportunity to further explore interdisciplinary approaches to research within software engineering, mental health research as well as outside of these disciplines. The research in this thesis has demonstrated that the themes of the literature do not always reflect the attitudes of the research community. Many comments were given by participants about the silo-based nature of their research area, and thus an opportunity to try to break down some of these barriers follows the work presented in this thesis.

6.3 Conclusions

This study has addressed the historically inward facing nature of software engineering and demonstrated not only the feasibility of interdisciplinary collaboration, but also several interdisciplinary areas from which software engineering may be able to learn. It has been demonstrated that contrary to the trends of software engineering literature, software engineers are not necessarily averse to adopting an interdisciplinary perspective. Through the use of a case study, it has been demonstrated that the practices within software engineering and mental health research are more similar than initially thought, and from this research, several areas for potential knowledge sharing have been identified. This work demonstrates the strength of adopting an interdisciplinary, collaborative attitude to research.

The work presented in this project has implications for the future directions of software engineering, particularly for software engineering research. It has been demonstrated that not only is interdisciplinary learning for the purposes of improving practices feasible, but that it is an area that is virtually untapped. By adopting an open mindset to interdisciplinary collaboration, software engineers can begin to look past the self-contained walls that the discipline has established for itself, and instead learn from the rich set of knowledge that lies beyond our disciplinary boundaries.

Appendix A

Project Change of Direction

This Appendix serves to explain the change of direction the project experienced part way through the project period.

The initial plan for the project was to investigate the effectiveness of software engineering techniques in a non-software engineering context. The inherent complexity of software engineering was investigated, and it was decided that the project would find another discipline in which complexity was inherent in which to translate techniques from within this domain. Complexity would serve as the thread that tied these seemingly disconnected disciplines together.

An identified domain that was inherently complex was that of mental health. Several meetings were held with staff from the (former) Centre for Mental Health Research (CMHR, now National Institute of Mental Health Research, NIMHR) in order to discuss this topic. A project was designed around running experiments within a high school environment in order to compare how traditional methods for dealing with complexity faired against techniques used within software engineering. Techniques that were to be evaluated included the use of abstraction, modelling and an agile approach to development (such as the use of user stories and creating a walking skeleton, as used in SCRUM). The initial project proposal stated that the research for this question would be:

> *Can Software Engineering practices, methodologies and laws that address complexity be generalised and effectively applied to disciplines outside of Software Engineering?*

A draft ethics proposal was created for the project in consultation with the CMHR staff (however this draft proposal never made it to submission). The plan was that experiments would be run over the winter holiday period, and possibly into the beginning of the second semester.

During a meeting with the CMHR staff on the 9^{th} of May, a discussion regarding the processes and techniques used in agile software development was underway (for the purposes of identifying techniques that would be appropriate to be applied within schools), when one mental health researcher mentioned that agile was very similar to a practice currently being performed in mental health. A follow-up e-mail

exchange with Rebecca Randall of the CMHR identified that the practice being discussed was Community Based Participatory Design (Personal communication, 23rd May 2014). Investigation into this practice led to the overarching set of practices under the umbrella of Community Based Participatory Research (CBPR). Further investigation showed that the relationship between Community Based Participatory Design and CBPR was much like the relationship between SCRUM and agile software development.

Along with preparing for the upcoming experiments, further research was performed with regards to CBPR. An interesting phenomenon was occurring- the more investigation that was done on CBPR, the closer in design it seemed to be to agile software development (albeit, performed within a different context). However, despite extensive searches, there appeared to be no link between the two sets of practices in the literature.

Further discussions were had between the CMHR staff, and with the supervisors of this project, and there was no clear understanding or obvious reason for how this phenomenon came to be. This seemed very interesting, and raised questions regarding the fact that if we are doing our practices so similarly, was there potential for us to learn from one another (for example, sharing 'lessons learned', or mitigations for potential issues). There was no precedence for this multidisciplinary, multi-practice research within software engineering, however as this project was already established to cross interdisciplinary boundaries, it seemed like an appropriate candidate for researching these questions. The issue came in that there was not enough time in the project period to conduct both the original experiment, and investigate this interesting relationship between agile and CBPR.

The researcher consulted with the supervisors of this project, who advised that while it could put the project under time pressure, it was not uncommon for research projects to pursue interesting and emerging phenomena. After a meeting with the Research School of Computer Science's Honours Convenor in late July, the researcher decided to change direction for the project towards the emerging relationship between agile and CBPR.

The major changes for the project included:

- Major focus shift, where now attention would go to comparing agile and CBPR;
- Rather than simply translating software engineering practices into another context, the emphasis would be on determining whether a two-way flow of knowledge with another discipline was possible
- Emphasis was put on the practices themselves rather than the abstract idea of 'complex systems', although those notions were not abandoned entirely
- Emphasis moved from validation of software engineering techniques to opportunities for interdisciplinary learning

Given the shortened time frame for the project, this meant that any case study that was to be performed had to be scaled down compared to the original plan. A new

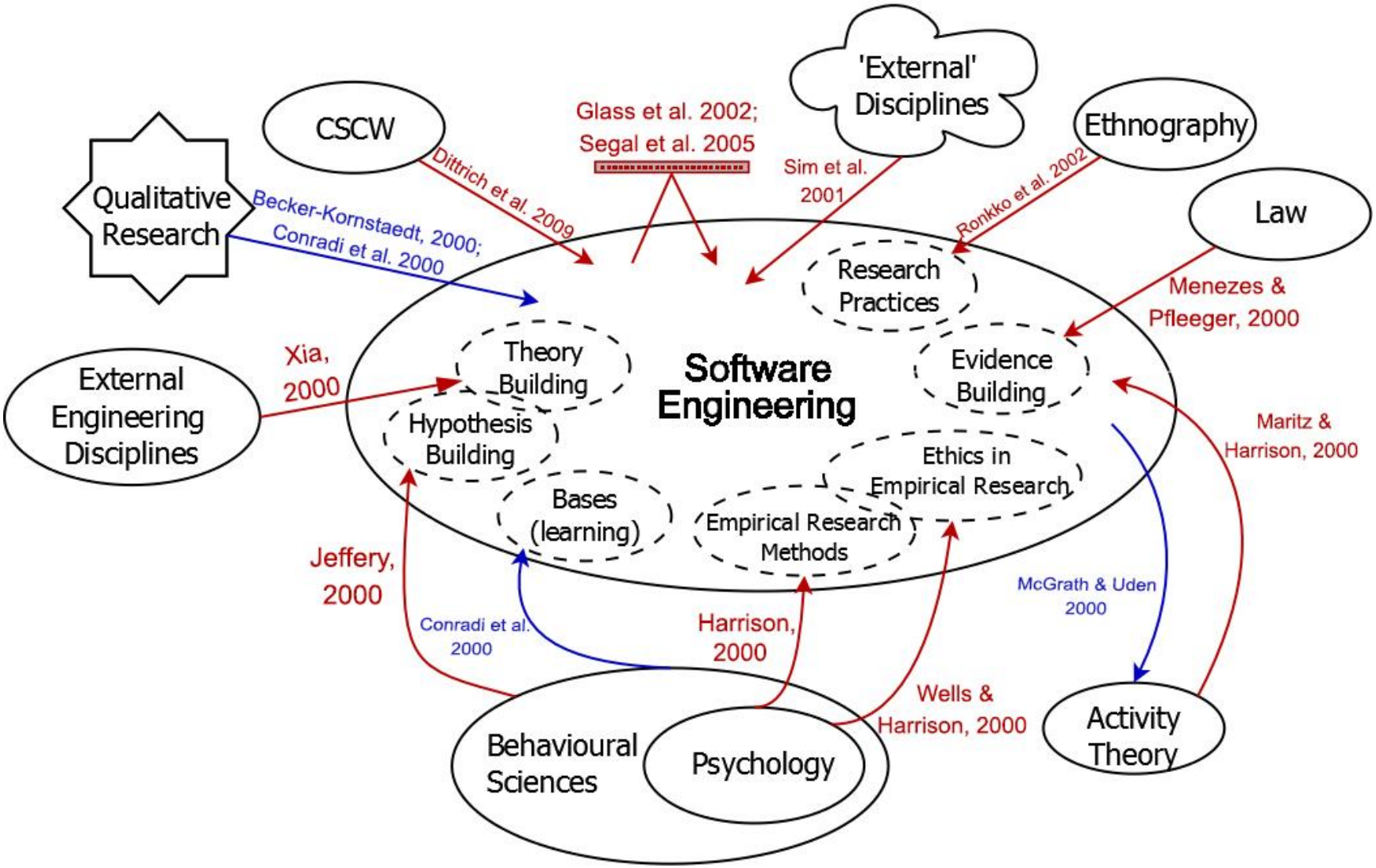
project outline was drafted, and a new ethics protocol was created. A large part of the literature review also had to be re-written to fit in with the new project direction.

The researcher maintains the opinion that this change of direction strengthened the research component of the project, and as seen throughout the thesis, has provided the foundation for future interdisciplinary work to be performed within the discipline of software engineering.

Appendix B

Interdisciplinary Research within Software Engineering

Please see over for an annotated model of interdisciplinary work undertaken within Software Engineering to date.


'External' Disciplines
CSCW
Glass et al. 2002; Segal et al. 2005
Ethnography
Qualitative Research
Dittrich et al. 2009
Sim et al. 2001
Ronkko et al. 2002
Law
Becker-Kornstaedt, 2000; Conradi et al. 2000
Research Practices
Menezes & Pfleeger, 2000
Xia, 2000
Theory Building
Software Engineering
Evidence Building
External Engineering Disciplines
Hypothesis Building
Maritz & Harrison, 2000
Ethics in Empirical Research
Bases (learning)
Empirical Research Methods
Jeffery, 2000
McGrath & Uden 2000
Conradi et al. 2000
Harrison, 2000
Wells & Harrison, 2000
Activity Theory
Behavioural Sciences
Psychology

Appendix C

Case Study Information Sheet

Participant Information Sheet

Researcher:
My name is Damien Beard. I am a final year Software Engineering student in the Research School of Computer Science at the Australian National University undertaking an individual research project resulting in a thesis. My principal supervisors are Dr Shayne Flint and Dr Ramesh Sankaranarayana.

Project Title:
Investigating complexity using Software Engineering Methods, with application to Mental Health Research

General Outline of the Project:
This project will be looking at the way Software Engineering practices deal with complex problems. The project will investigate whether these practices can be generalised and applied to an external domain to see if the practices within Software Engineering are applicable to other complex areas, or if they are constrained to being effective only within the domain of Software Engineering. This project is also investigating techniques for creating two-way flows of knowledge across disciplines and domains, including between Software Engineering and Mental Health Research. Data will be collected from interviews with Mental Health researchers and experts in Software Engineering and Psychology.

The data collected during this research will be presented in an Honours thesis with a chance of publication in a Software Engineering journal.

Participant Involvement:
- Participation in this study is entirely voluntary and participants may without any penalty, decline to take part or withdraw from the research at any time up until the submission of the Honours thesis without providing an explanation, or refuse to answer a question. Non-participation will not be held against any potential candidate for this study. If you do chose to withdraw from this study, your data will be destroyed.
- Participants will be asked questions as part of an interview based on current methods in their field of practice or research. The responses to these questions will be based on the participants' personal opinions. If appropriate, participants may be welcomed to participate in a follow up interview, however there is no requirement for them to do so.
- Notes regarding the results of the interviews will collected by the primary investigator. Audio recording may also be used to record interviews, however such technology will be drawn to participants' attention prior to the commencement of any interview, and will only be used with the participant's consent.
- The research will take place in a classroom or office setting, either on or off the Australian National University campus. Interviews are expected to take between 45 and 60 minutes.
- The activities undertaken by participants may involve sensitive topics surrounding Mental Health and Mental Health Research. Participation is voluntary, and if participants feel discomfort or stress as a result of these activities they are encouraged to contact their local counselling service, or Lifeline Australia (whose details can be found under Queries and Concerns).

Confidentiality:

- Only my project supervisors and I will have access to the collected data.
- With regard to publication of results, participant information will be attributed by **occupation only** (e.g. 'High School Teacher'), with participants being referenced by an unidentifiable pseudonym where necessary. No references will be made to participants' names.
- Participants have the option to choose their desired level of attribution on the participant consent form.
- Confidentiality will be protected as far as the law allows.

Data Storage:

- Interview recordings will be stored at the Research School of Computer Science at the Australian National University for one year following any publications arising from this thesis in the office of my primary supervisor. After this time they will be destroyed.
- Interview transcripts and field notes from interviews will be stored on a password protected laptop for one year following any publications arising from this thesis. After this time they will be erased from the computer.

Queries and Concerns:

- If you require further information, please contact myself at u4847291@anu.edu.au or my primary supervisors Dr Shayne Flint and Dr Ramesh Sankaranarayana at Shayne.flint@anu.edu.au and ramesh@cs.anu.edu.au respectively.

Ethics Committee Clearance:

The ethical aspects of this research have been approved by the ANU Human Research Ethics Committee. If you have any concerns or complaints about how this research has been conducted, please contact:

Ethics Manager
The ANU Human Research Ethics Committee
The Australian National University
Telephone: +61 (0) 2 6125 3427
Email: Human.Ethics.Officer@anu.edu.au

Appendix D

Case Study Invitation E-mail and Consent Form

INVITATION EMAIL

Good morning/afternoon,

My name is Damien Beard, and I am a final year Software Engineering Student in the Research School of Computer Science at the Australian National University. I am currently undertaking an individual research project which will result in a thesis.

The focus of my project has been investigating the ways in which Software Engineering addresses complexity, including if and how these practices can be generalised in order to have cross-discipline application. This study includes investigating current practices in Software Engineering as well as other domains which address complex systems and problems. The focus has largely been on User-Driven, iterative processes.

As an expert in X, I was hoping that I may be able to interview you in order to gain some insight into your area, in the hope that we may be able to broaden our understanding of User-Driven, iterative processes across disciplines. It is our hope that we may be able to address what the key differences are across disciplines, as well as which practices seem particularly effective across different disciplines. We are hoping that by conducting this research, it will be possible to facilitate a two-way flow of knowledge across different disciplines and domains in order to improve the way complex problems are addressed.

If you are interested in this research and would like to learn more before considering participation, I will send you a Participant Information Sheet which includes details about my project, as well as a consent form which I will collect at the time of the interview.

Thank you for taking the time to read this e-mail, I truly appreciate it. If you are willing to be a participant in this study, or would like further information about this project, please feel free to e-mail me directly at u4847291@anu.edu.au.

Kind Regards,

Damien Beard

Australian National University

WRITTEN CONSENT for Participants for the study known as Investigating complexity using Software Engineering Methods, with application to Mental Health Research

Ethics protocol number: 2014/368

I have read and understood the Information sheet you have given me about the research project and I agree to participate in the project.

Signature:…… Date……

YES ☐ NO ☐ I agree to this interview being audio taped

I agree to be identified in the following way

YES ☐ NO ☐ Occupation

YES ☐ NO ☐ Pseudonym

YES ☐ NO ☐ Complete confidentiality

Signature:…… Date……

Appendix E

Software Engineering Interview Questions

Date_____ Time_____

Interviewer _____

Consent Form Received ☐

- How much experience have you got in the discipline of Software Engineering?
- How much experience do you have with iterative development techniques such as Agile Programming?
- What is your understanding of how Agile was first conceptualised and developed? To the best of your knowledge, was there any influence from other disciplines in the creation of these techniques?
- What are the pain points in agile development? In your opinion what are the main reasons projects fail when using agile?
- What parts of Agile are directly based off existing Software Engineering research? How and why were these effective? In your opinion, do you think these practices could be generalised and translated into another discipline?
- Have a look at the simple Agile Diagram. From your experience, is there anything you would add (can be in your own notation)?
- Which parts of agile emphasise user engagement?
- Do you think that Agile is an effective technique for dealing with complexity? Why or why not?
- In your opinion, do you think other disciplines could benefit from using generalised, adapted practices and techniques from the Software Engineering domain?
- In your opinion, do you think that Software Engineering could benefit from using generalised, adapted practices and techniques from other disciplines?

Appendix F

Mental Health Interview Questions

Date_____ Time_____

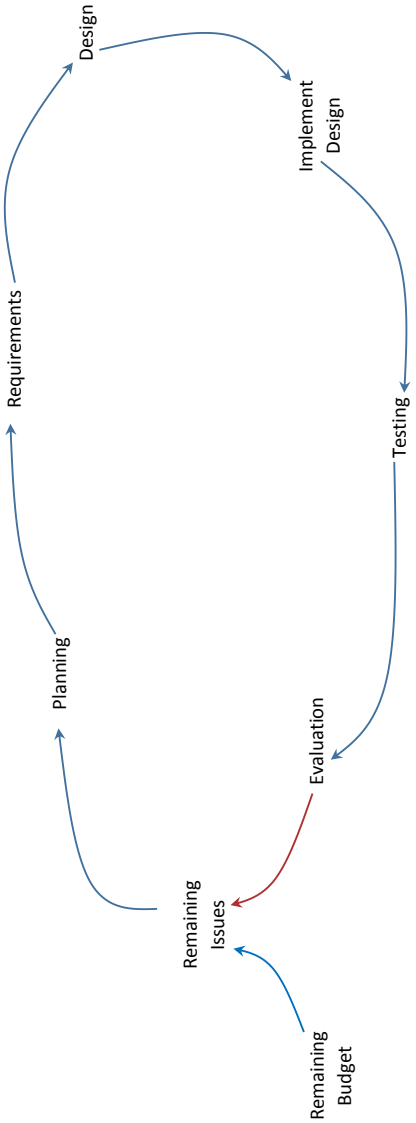
Interviewer _____

Consent Form Received ☐

- How much experience have you got with Mental Health Research?
- How much experience do you have with user-based, iterative research techniques such as Community Based Participatory Research?
- What is your understanding of how CBPR was first conceptualised and developed? To the best of your knowledge, was there any influence from other disciplines in the creation of these techniques?
- What are the pain points in CBPR? In your opinion what are the main reasons projects may not be successful when using CBPR?
- What parts of CBPR are directly based off existing Mental Health research? How and why were these effective? In your opinion, do you think these practices could be generalised and translated into another discipline?
- Have a look at the simple CBPR Diagram, generalised off one produced by the Young and Well CRC (2012). From your experience, is there anything you would add (can be in your own notation)?
- Which parts of CBPR emphasise user engagement the most, or in your experience is it pretty uniform throughout the whole process?
- Do you think that CBPR is an effective technique for dealing with complexity? Why or why not?
- In your opinion, do you think other disciplines could benefit from using generalised, adapted practices and techniques from the Mental Health Research domain?
- In your opinion, do you think that Mental Health Research could benefit from using generalised, adapted practices and techniques from other disciplines?

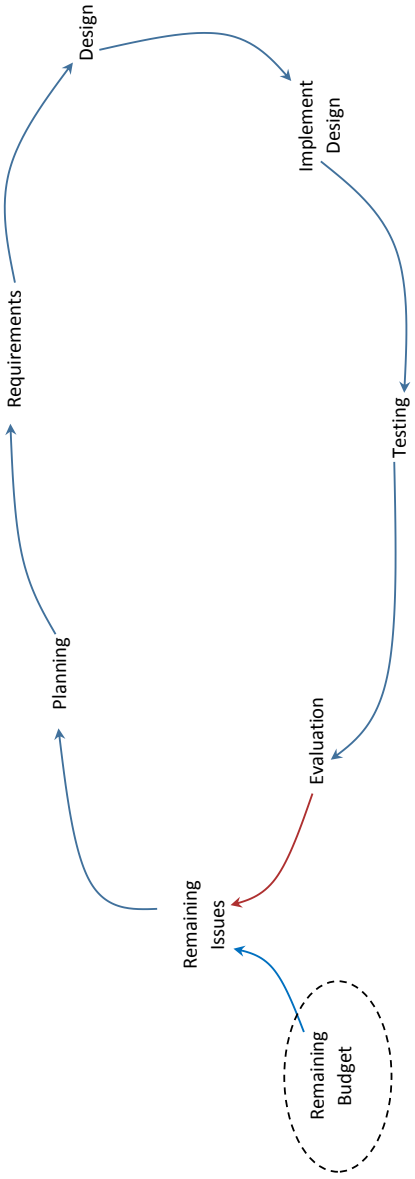
Appendix G

Systems model for Software Engineering Interviews



Appendix H

Systems model for Mental Health Interviews



Appendix I

Initial Coding Schema from Open Coding Activities

The set of codes presented in this Appendix are the result of open coding across the case study data set. These are the raw codes that were iteratively merged throughout the axial coding process.

I.1 Software Engineering Open Codes

- Agile
  - Iterative has always been done
  - Requirements volatility
  - Communication
  - Communication Barriers
  - Dealing with Complexity
  - Issues
    - Lack of Understanding
    - Lack of Documentation
    - Adherence to strict timeframes
    - Attitude of Development Team
    - Danger of unengaged client
    - Culture & management support
    - People averse to change
  - Loops within Loops
  - Chaos
  - Prototyping
  - User Engagement
  - Customer Involvement

  - Model Development
  - Requirements
  - Management of documentation
  - Mechanism for quality
  - Catering for code-hackers
  - Agile Research
  - Software Engineering Research
  - Budget
  - Benefits
  - Rework
  - Project Scope
  - Planning
  - Project Failure
  - Customer Engagement
  - Stakeholder identification
  - Effective for non-functional requirements
  - Effectiveness dependent on experience of practitioners
  - Importance of attitude and culture
  - Effective for co-location
  - Planning
  - Evaluation
  - Complexity
    * Effective at dealing with Complexity
    * Not effective at dealing with complexity
  - Benefits of automation
  - Good for learning
- Interdisciplinary approaches
  - Push-out
    * Agile to education
    * Software engineering in education
    * Problem representation
    * Modelling
    * Law enforcement
    * Agile will only work in environments with open minded people
    * Electronics

    - Large scale infrastructure
  - Pull-in
    - Automation from manufacturing
    - Adopting Scientific approaches
    - Recognition of certification
    - Emphasis on training programs
    - Currently using Project Management techniques from outside of SE
    - Standardisation and model driven development from architecture
  - Software Engineering silo based

I.2 Mental Health Open Codes

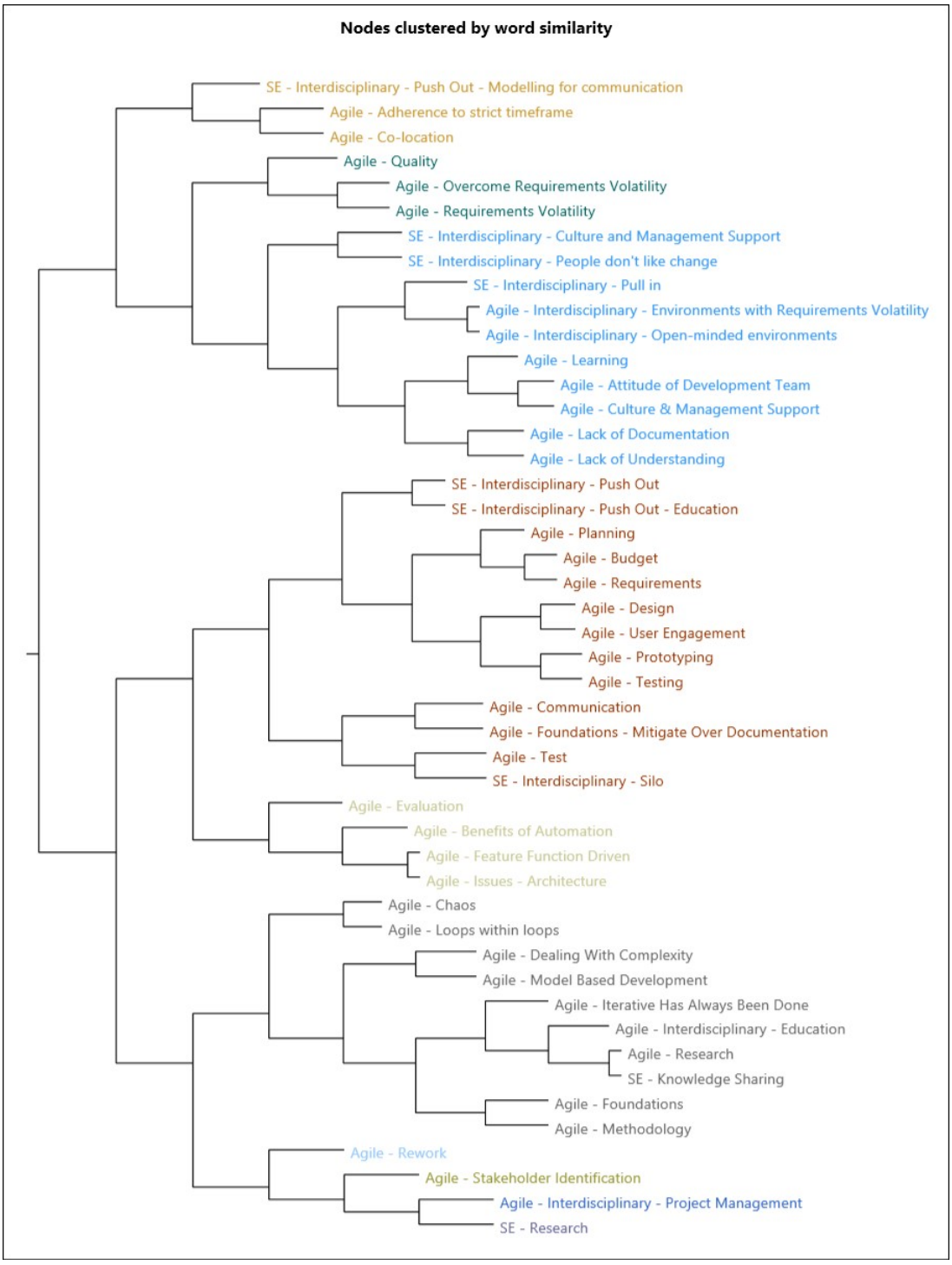
- CBPR
  - Importance of culture
  - Time constraints
  - Budget constraints
  - Helps understand requirements
  - Complexity
  - Organically grown
  - User Engagement
    - Testing
    - Design
    - Planning
    - Remaining Issues
    - Requirements
    - Implementation
    - Evaluation
  - User Participation
  - Loops within Loops
  - CBPR embraces complexity
  - Generates creativity amongst users
  - Provides users with sense of ownership
  - Simplification of design process
  - Diversity of stakeholders
  - Effective stakeholder identification
  - Strong consumer focus

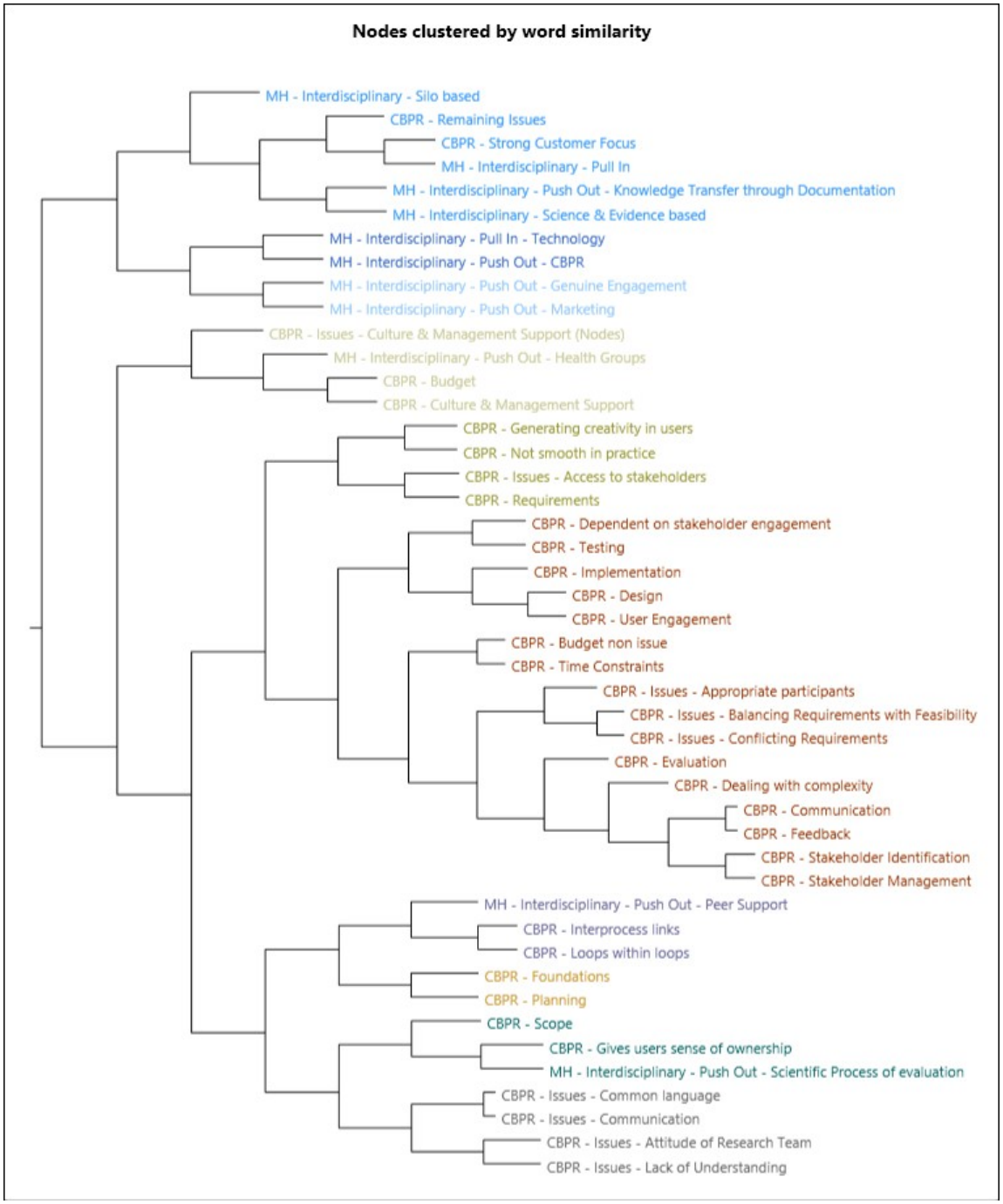
  - Dependent on stakeholder engagement
  - Execution not always as per general CBPR model in practice
  - Issues
    - Culture Challenges
    - Lack of Understanding
    - Lack of education
    - Stakeholder management
    - Understanding Requirements
    - Management Support
    - Time constraints within iterations (over budget)
    - Communication with users
    - Providing feedback
    - User engagement
    - Access to stakeholders
    - Balancing Requirements with feasibility
    - Conflicting Requirements
  - Focus is on delivery of high quality services
- Interdisciplinary approaches
  - Push out
    - Other health disciplines
    - Peer support programs
    - Market Researchers
    - Science and evidence based approaches
    - Rigorous, scientific evaluations of projects
    - Knowledge sharing through documentation
  - Pull in
    - Technology for health promotion
    - Open minded but unaware
  - Silo based

Appendix J

Results of cluster analysis on open codes

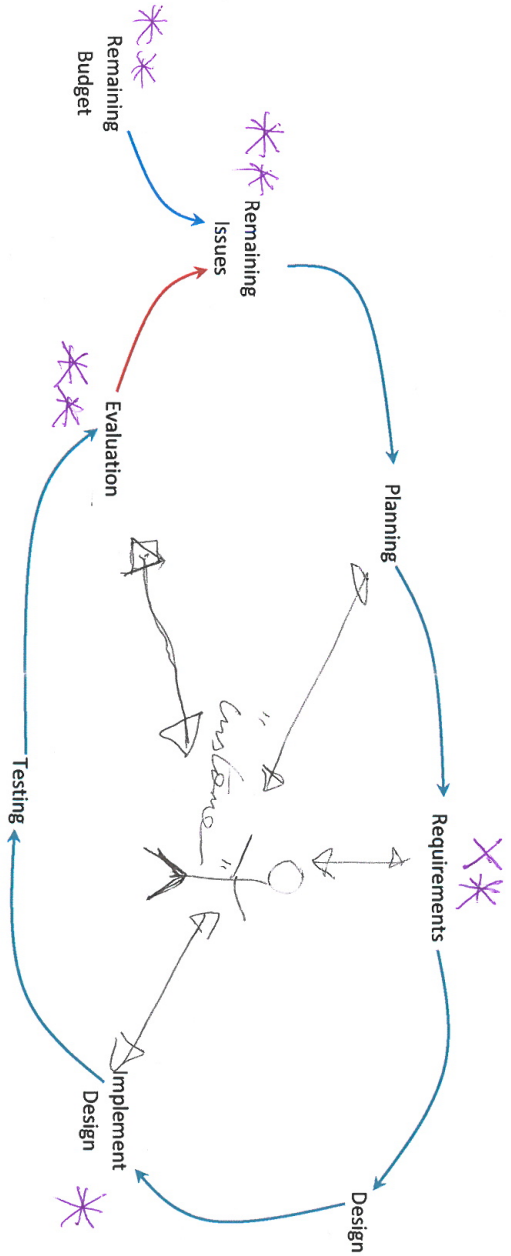
See over for the results of performing a cluster analysis across the software engineering open codes and mental health open codes respectively. These analyses were performed prior to axial coding, using the qualitative analysis software NVivo.

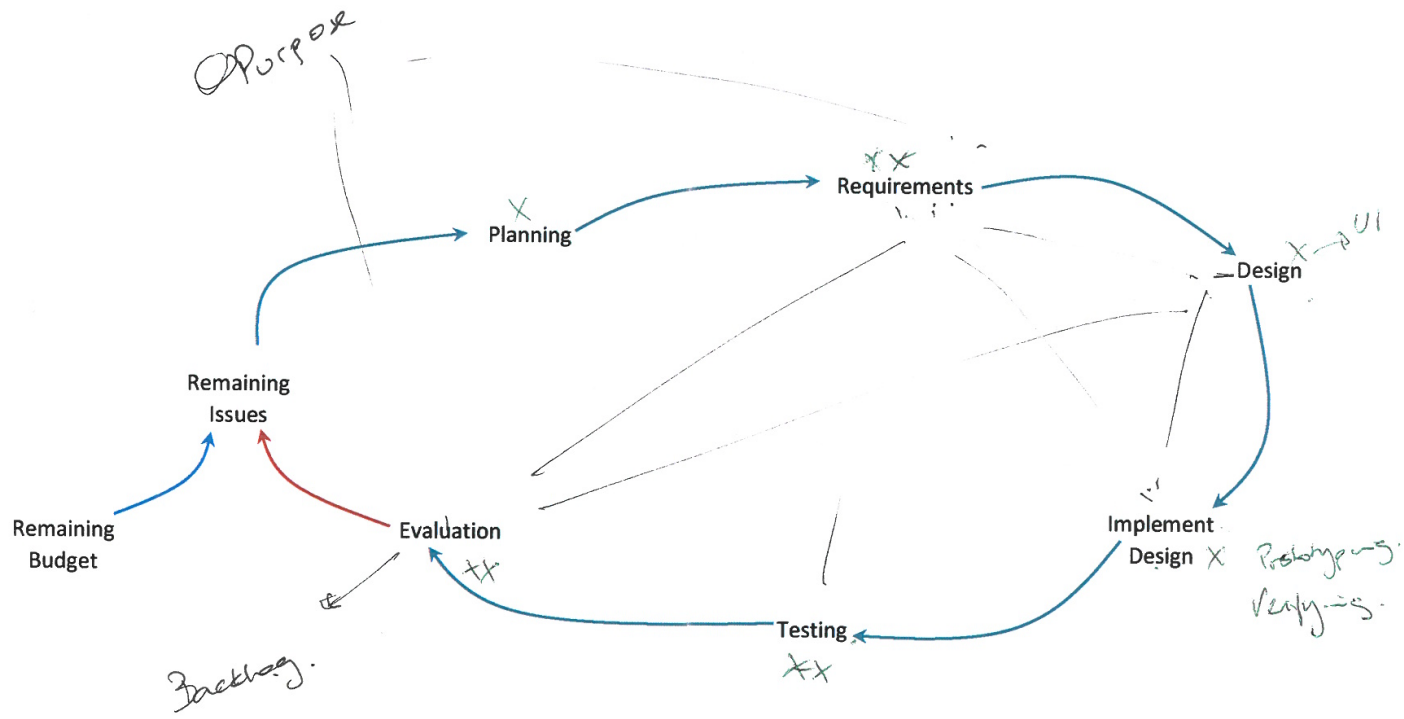

Nodes clustered by word similarity
SE - Interdisciplinary - Push Out - Modelling for communication
Agile - Adherence to strict timeframe
Agile - Co-location
Agile - Quality
Agile - Overcome Requirements Volatility
Agile - Requirements Volatility
SE - Interdisciplinary - Culture and Management Support
SE - Interdisciplinary - People don't like change
SE - Interdisciplinary - Pull in
Agile - Interdisciplinary - Environments with Requirements Volatility
Agile - Interdisciplinary - Open-minded environments
Agile - Learning
Agile - Attitude of Development Team
Agile - Culture & Management Support
Agile - Lack of Documentation
Agile - Lack of Understanding
SE - Interdisciplinary - Push Out
SE - Interdisciplinary - Push Out - Education
Agile - Planning
Agile - Budget
Agile - Requirements
Agile - Design
Agile - User Engagement
Agile - Prototyping
Agile - Testing
Agile - Communication
Agile - Foundations - Mitigate Over Documentation
Agile - Test
SE - Interdisciplinary - Silo
Agile - Evaluation
Agile - Benefits of Automation
Agile - Feature Function Driven
Agile - Issues - Architecture
Agile - Chaos
Agile - Loops within loops
Agile - Dealing With Complexity
Agile - Model Based Development
Agile - Iterative Has Always Been Done
Agile - Interdisciplinary - Education
Agile - Research
SE - Knowledge Sharing
Agile - Foundations
Agile - Methodology
Agile - Rework
Agile - Stakeholder Identification
Agile - Interdisciplinary - Project Management
SE - Research

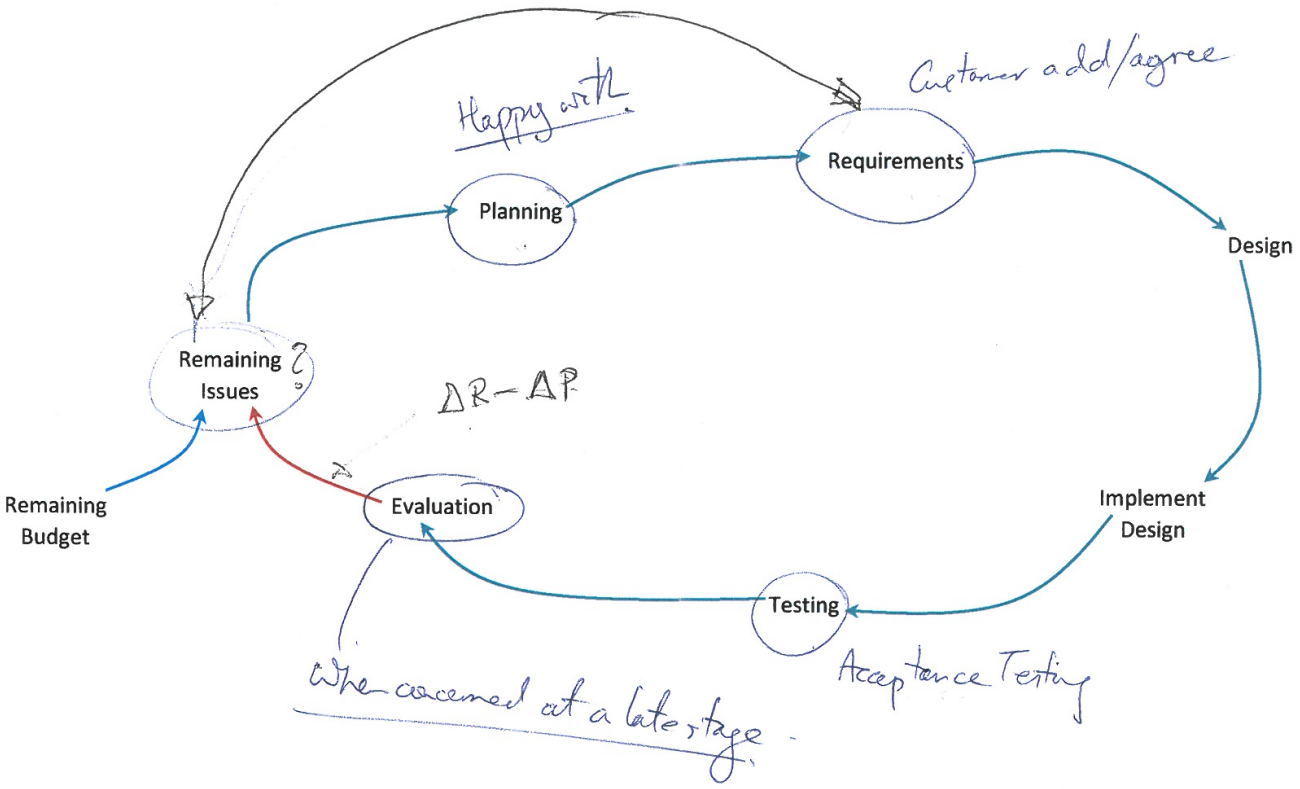

Nodes clustered by word similarity
MH - Interdisciplinary - Silo based
CBPR - Remaining Issues
CBPR - Strong Customer Focus
MH - Interdisciplinary - Pull In
MH - Interdisciplinary - Push Out - Knowledge Transfer through Documentation
MH - Interdisciplinary - Science & Evidence based
MH - Interdisciplinary - Pull In - Technology
MH - Interdisciplinary - Push Out - CBPR
MH - Interdisciplinary - Push Out - Genuine Engagement
MH - Interdisciplinary - Push Out - Marketing
CBPR - Issues - Culture & Management Support (Nodes)
MH - Interdisciplinary - Push Out - Health Groups
CBPR - Budget
CBPR - Culture & Management Support
CBPR - Generating creativity in users
CBPR - Not smooth in practice
CBPR - Issues - Access to stakeholders
CBPR - Requirements
CBPR - Dependent on stakeholder engagement
CBPR - Testing
CBPR - Implementation
CBPR - Design
CBPR - User Engagement
CBPR - Budget non issue
CBPR - Time Constraints
CBPR - Issues - Appropriate participants
CBPR - Issues - Balancing Requirements with Feasibility
CBPR - Issues - Conflicting Requirements
CBPR - Evaluation
CBPR - Dealing with complexity
CBPR - Communication
CBPR - Feedback
CBPR - Stakeholder Identification
CBPR - Stakeholder Management
MH - Interdisciplinary - Push Out - Peer Support
CBPR - Interprocess links
CBPR - Loops within loops
CBPR - Foundations
CBPR - Planning
CBPR - Scope
CBPR - Gives users sense of ownership
MH - Interdisciplinary - Push Out - Scientific Process of evaluation
CBPR - Issues - Common language
CBPR - Issues - Communication
CBPR - Issues - Attitude of Research Team
CBPR - Issues - Lack of Understanding

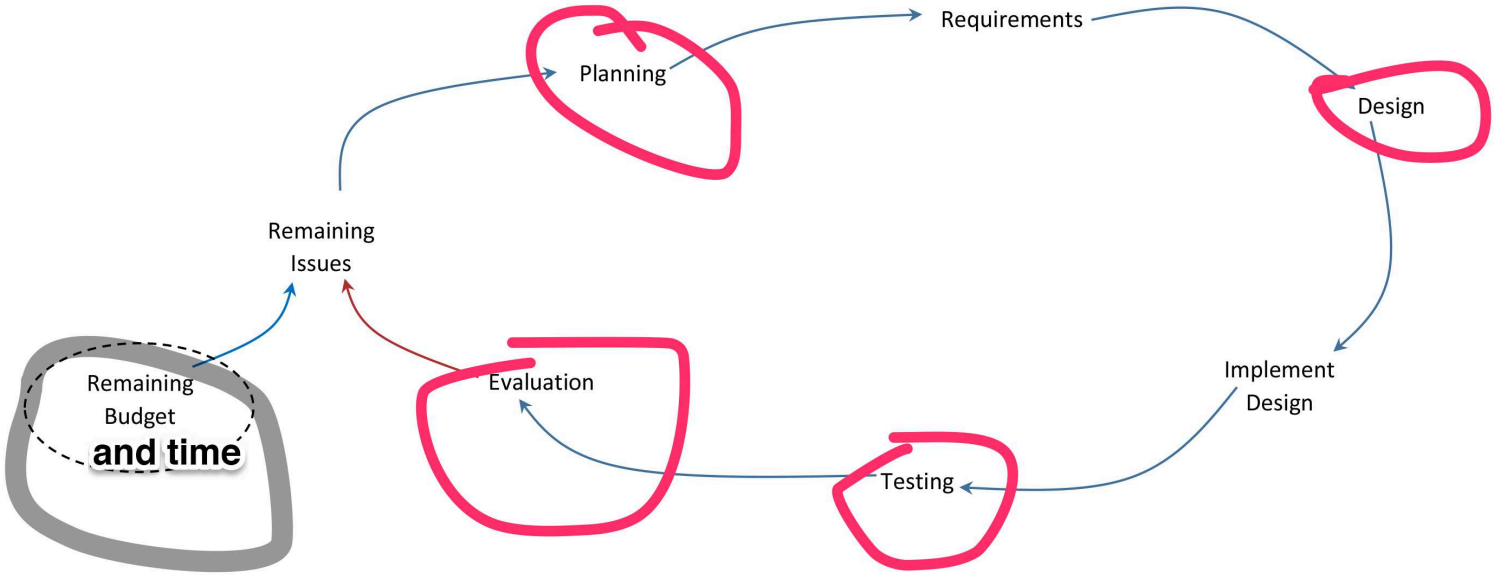
Appendix K

Annotated Systems Models

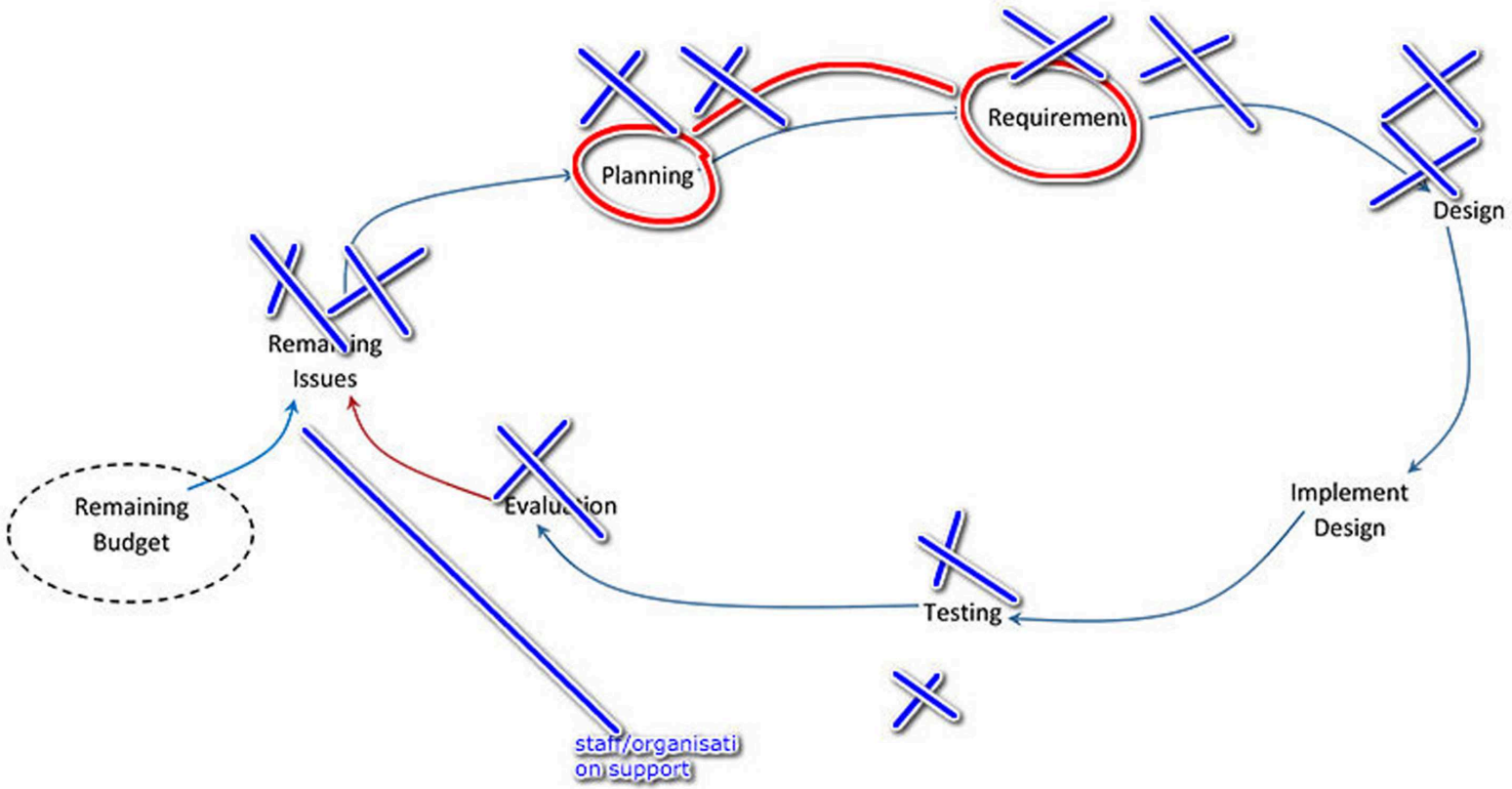


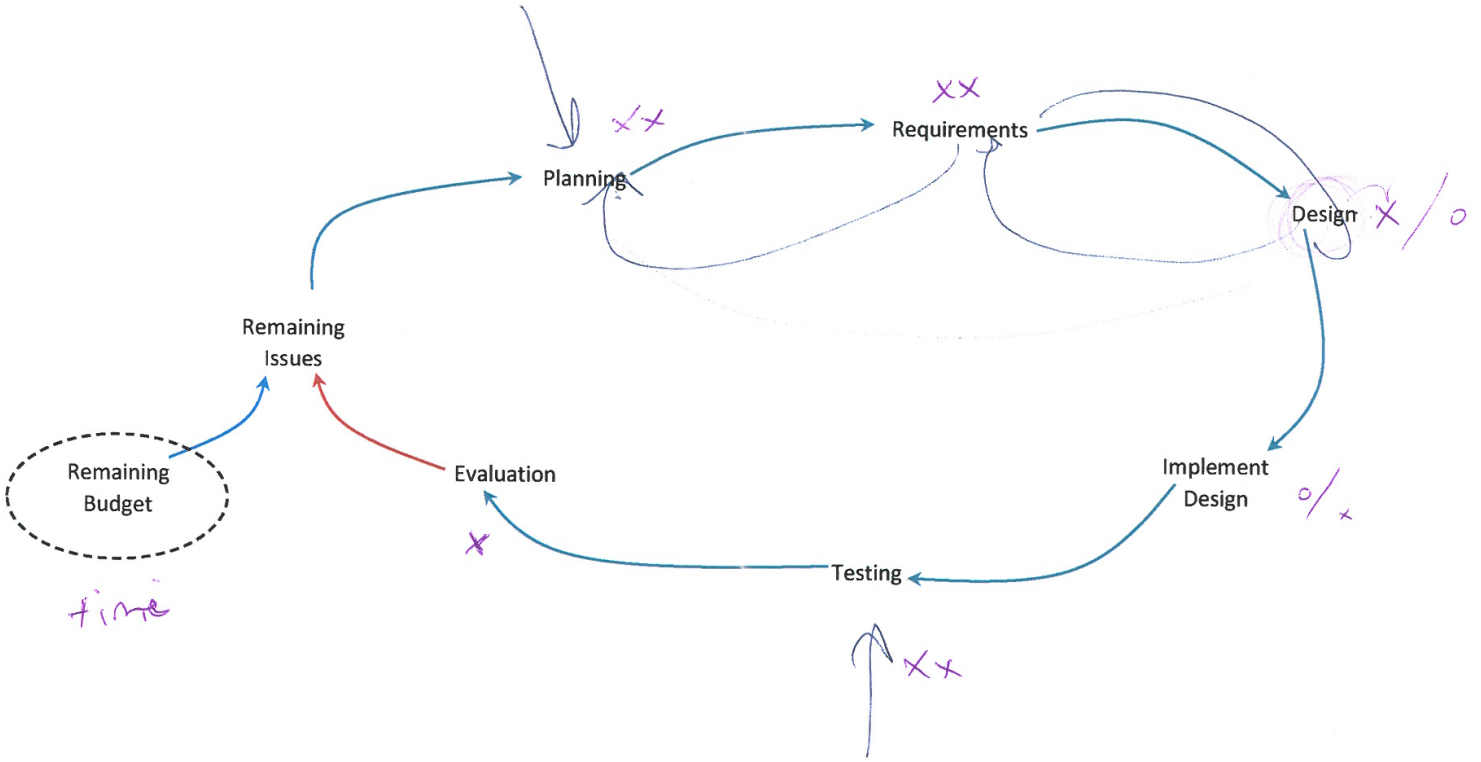

Purpose
Planning
Requirements
Design
UI
Remaining
Issues
Remaining
Budget
Evaluation
Implement
Design
Prototyping
Verifying
Testing
Backlog


Happy with
Customer add/agree
Requirements
Planning
Design
Remaining Issues
ΔR-ΔP
Remaining Budget
Evaluation
Implement Design
Testing
When arcemed at a late, tage
Acceptance Testing


Requirements
Planning
Design
Remaining
Issues
Implement
Design
Evaluation
Testing
Remaining
Budget
and time

Micro loops


Planning
Requirement
Design
Remaining
Issues
Remaining
Budget
Evaluation
Implement
Design
Testing
staff/organisati
on support


Planning
Requirements
Design
Implement Design
Testing
Evaluation
Remaining Issues
Remaining Budget

Bibliography

AMBLER, S. 2012. The Agile System Development Life Cycle (SDLC). http://www.ambysoft.com/essays/agileLifecycle.html.

AMERICAN PSYCHOLOGICAL ASSOCIATION. 2014. Health Care Reform: Integrated Health Care. http://www.apa.org/about/gr/issues/health-care/integrated.aspx.

ANDERSON, C. 2010. Presenting and evaluating qualitative research. *American journal of pharmaceutical education 74*, 8.

ANDREWS, A. A. AND PRADHAN, A. S. 2001. Ethical issues in empirical software engineering: The limits of policy. *Empirical Softw. Engg. 6*, 2 (June), 105–110.

ARONSON, D. 1996. Overview of systems thinking. *The thinking page*, 1 – 3.

AUERSWALD, E. H. 1968. Interdisciplinary versus ecological approach. *Family Process 7*, 2, 202–215.

BECKER-KORNSTAEDT, U. 2000. Knowledge Elicitation for Descriptive Software Process Modeling. In *ICSE'2000 Workshop on "Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research"* (2000), pp. 1 – 6.

BECKMAN, K., COULTER, N., KHAJENOORI, S., AND MEAD, N. R. 1997. Collaborations: Closing the industry-academia gap. *IEEE Softw. 14*, 6 (Nov.), 49–57.

BELLINGER, G. 2004. Systems Thinking - An Operational Perspective of the Universe. http://www.systems-thinking.org/systhink/systhink.htm.

BENNETT, E. M., CUMMING, G. S., AND PETERSON, G. D. 2005. A Systems Model Approach to Determining Resilience Surrogates for Case Studies. *Ecosystems 8*, 8 (Nov.), 945–957.

BENNETT, K., LAYZELL, P., BUDGEN, D., BRERETON, P., MACAULAY, L., AND MUNRO, M. 2000. Service-based software: The future for flexible software. In *Proceedings of the 7th Asia-Pacific Software Engineering Conference, (Singapore*.

BJØRNSON, F. O. AND DINGSØYR, T. 2008. Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used. *Inf. Softw. Technol. 50*, 11 (Oct.), 1055–1068.

BOEHM, B. 1986. A spiral model of software development and enhancement. *SIGSOFT Softw. Eng. Notes 11*, 4 (Aug.), 14–24.

BOEHM, B. W. 2002. Software pioneers. Chapter Software Engineering Economics, pp. 641–686. New York, NY, USA: Springer-Verlag New York, Inc.

BOOCH, G. 1994. *Object-Oriented Analysis and Design with Applications, Addison-Wesley*.

BOOCH, G., RUMBAUGH, J., AND JACOBSON, I. 1997. *Unified Modeling Language*. Version 1.

BOURQUE, P. AND FAIRLEY, R. E. 2013. *Guide to the Software Engineering Body of Knowledge SWEBOK V3*.

BROOKS, F. P., JR. 1987. No silver bullet essence and accidents of software engineering. *Computer 20*, 4 (April), 10–19.

BROWN, A. W. AND MCDERMID, J. A. 2007. The art and science of software architecture. In *Proceedings of the First European Conference on Software Architecture*, ECSA'07 (Berlin, Heidelberg, 2007), pp. 237–256. Springer-Verlag.

BURNS, J. M. C. 2010. *Cross-Case synthesis and Analysis* (0 ed.)., pp. 265–268. SAGE Publications, Inc.

CHECKLAND, P. AND POULTER, J. 2006. *Learning for action: a short definitive account of soft systems methodology and its use for practitioner, teachers, and students*. Wiley.

CONRADI, R., LINDVALL, M., AND SEAMAN, C. 2000. Success Factors for Software Experience Bases : What We Need to Learn from Other Disciplines 2 . Success Factors for Software Experience Bases. In *ICSE'2000 Workshop on "Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research"* (2000), pp. 1–7.

CORNWALL, A. AND JEWKES, R. 1995. What is participatory research? *Social science & medicine* 12.

CRESWELL, J. 1998. *Qualitative inquiry and research design: choosing among five traditions*. Sage Publications series. Sage Publications.

DAHL, O. J., DIJKSTRA, E. W., AND HOARE, C. A. R. Eds. 1972. *Structured Programming*. Academic Press Ltd., London, UK, UK.

DITTRICH, Y., RANDALL, D. W., AND SINGER, J. 2009. Software engineering as cooperative work. *Comput. Supported Coop. Work 18*, 5-6 (Dec.), 393–399.

DYBÅ, T. AND DINGSØYR, T. 2008. Empirical studies of agile software development: A systematic review. *Inf. Softw. Technol. 50*, 9-10 (Aug.), 833–859.

EASTERBROOK, S., SINGER, J., ANNE STOREY, M., AND DAMIAN, D. Selecting empirical methods for software engineering research.

ENDRES, A. AND ROMBACH, D. 2003. *A Handbook of Software and Systems Engineering: Empirical Observations, Laws and Theories, Pearson Educations Limited*. Pearson Educations Limited, Great Britain.

FARIDI, Z., GRUNBAUM, J. A., GRAY, B. S., FRANKS, A., AND SIMOES, E. 2007. Community-based participatory research: necessary next steps.

FOWLER, M. AND HIGHSMITH, J. 2001. The agile manifesto. *Software Development* August.

GEBBIE, K., ROSENSTOCK, L., AND HERNANDEZ, L. M. 2003. *Who Will Keep the Public Healthy? Educating Public Health Professionals for the 21st Century*. Washington, DC, Institute of Medicine.

GLASS, R., VESSEY, I., AND RAMESH, V. 2002. Research in software engineering: an analysis of the literature. *Information and Software Technology 44*, 8, 491 – 506.

GLASS, R. L. 1969. An elementary discussion of compiler/interpreter writing. *ACM Comput. Surv. 1*, 1 (March), 55–77.

HAGEN, P., COLLIN, P., METCALF, A., NICHOLAS, M., K, R., AND N, S. 2012. Participatory Design of evidence-based online youth mental health promotion, intervention and treatment. *Young and Well*.

HARRISON, W. 2000. N= 1: an alternative for software engineering research. In *ICSE'2000 Workshop on "Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research"*, Volume 2 (2000).

HERBSLEB, J. D. AND MOCKUS, A. 2003. Formulation and preliminary test of an empirical theory of coordination in software engineering. *ACM SIGSOFT Software Engineering Notes 28*, 5, 138.

HEUSSER, M. Has Agile Software Development Gone Mainstream?

HIGHSMITH, J. AND COCKBURN, A. 2001. Agile software development: the business of innovation. *Computer 34*, 9 (Sep), 120–127.

HOLMSTRM, H., FITZGERALD, B., GERFALK, P. J., AND CONCHIR, E. . 2006. Agile practices reduce distance in global software development. *Information Systems Management 23*, 3, 7–18.

HOSIE, A., VOGL, G., HODDINOTT, J., J, C., AND Y, C. 2014. CROSSROADS : RETHINKING THE AUSTRALIAN MENTAL. *ReachOut.com by Inspire Foundation*.

ISRAEL, B. A., ENG, E., SCHULZ, A. J., AND PARKER, E. A. 2012. *Methods for Community-Based Participatory Research for Health*. John Wiley and Sons.

ISRAEL, B. A., SCHULZ, A. J., PARKER, E. A., BECKER, A. B., ALLEN, A., AND GUZMAN, J. R. 2008. Critical issues in developing and following CBPR Principles. pp. 47–66.

JEFFERY, R. 2000. Theory, models and methods in software engineering research. In *ICSE'2000 Workshop on "Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research"* (2000), pp. 2–7.

KANARACUS, C. Air Force scraps massive ERP project after racking up $1 billion in costs. `http://www.cio.com/article/2390341/cio-role/air-force-scraps\-massive-erp-project-after-racking-up--1-billion-in-costs.html`.

KHANDUJA, J. 2014. Quality Assurance and Project Management. http://itknowledgeexchange.techtarget.com/quality-assurance/agile-methodology-and-project-management/.

KOHN, L. T. 1997. *Methods in case study analysis*. Center for Studying Health System Change.

LARMAN, C. AND BASILI, V. R. 2003. Iterative and incremental development: A brief history. *Computer 36*, 6 (June), 47–56.

LAWRENCE, M. J. 1982. An examination of evolution dynamics. In *Proceedings of the 6th International Conference on Software Engineering*, ICSE '82 (Los Alamitos, CA, USA, 1982), pp. 188–196. IEEE Computer Society Press.

LEHMAN, M. M. 1996. Laws of software evolution revisited. In *Proceedings of the 5th European Workshop on Software Process Technology*, EWSPT '96 (London, UK, UK, 1996), pp. 108–124. Springer-Verlag.

LETHBRIDGE, T. C., SIM, S. E., AND SINGER, J. 2005. Studying software engineers: Data collection techniques for software field studies. *Empirical Softw. Engg. 10*, 3 (July), 311–341.

MARINERTEK. 2014. Cleveland .NET Programming & SQL Developers in northeast OH. http://www.marinertek.com/services/custom-programming-application-development.

MCGRATH, G. M. AND UDEN, L. 2000. Beg , Borrow or Steal : OK , but it ' s not all One-Way Traffic ! In *ICSE'2000 Workshop on "Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research"* (2000), pp. 1–6.

MENDOZA, J., BRESNAN, A., ROSENBERG, S., ELSON, A., GILBERT, Y., LONG, P., WILSON, K., AND HOPKINS, J. 2013. *Obsessive Hope Disorder Reflections on 30 Years of Mental Health Reform in Australia and Visions for the Future*. ConNetica.

MENEZES, W. AND LAWRENCE-PFLEEGER, S. 2000. Improving Software Technology Transfer: Making Decisions based on Evidence In. In *ICSE'2000 Workshop on "Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research"* (2000), pp. 1 – 5.

MINKLER, M. AND WALLERSTEIN, N. 2011. *Community-Based Participatory Research for Health: From Process to Outcomes*. Wiley Desktop Editions. Wiley.

NAYAB, N. 2011. The Extreme Programming Life Cycle. http://www.brighthubpm.com/methods-strategies/88996-the-extreme-programming-life-cycle/.

NERUR, S., MAHAPATRA, R., AND MANGALARAJ, G. 2005. Challenges of migrating to agile methodologies. *Commun. ACM 48*, 5 (May), 72–78.

NUANCE COMMUNICATIONS. 2014. Dragon - Dragon NaturallySpeaking. http://australia.nuance.com/dragon/index.htm.

ONE, V. 2012. The seventh annual "State of Agile Development" survey.

PARKER, E. A., ISRAEL, B. A., WILLIAMS, M., BRAKEFIELD-CALDWELL, W., LEWIS, T. C., ROBINS, T., RAMIREZ, E., ROWE, Z., AND KEELER, G. 2003. Community action against asthma. *Journal of General Internal Medicine 18*, 7, 558–567.

PARNAS, D. L. 1995. On icse's “most influential” papers. *SIGSOFT Softw. Eng. Notes 20*, 3 (July), 29–32.

PROJECT LIFECYCLE SERVICES. 2007. AGILE SCRUM Project Management. `http://www.projectlifecycleservicesltd.co.uk/project-management-services/scrum-project-management.php`.

QSR INTERNATIONAL. 2012. NVivo qualitative data analysis software, QSR International Pty Ltd. Version 10.

RICHARDSON, G. P. AND PUGH, A. L. 1981. *Introduction to System Dynamics Modeling with Dynamo*. MIT Press, Cambridge, MA, USA.

ROBSON, C. 2002. *Real World Research: A Resource for Social Scientists and Practitioner-Researchers*. Regional Surveys of the World Series. Wiley.

RÖNKKÖ, K., LINDEBERG, O., AND DITTRICH, Y. 2002. 'bad practice' or 'bad methods' " are software engineering and ethnographic discourses incompatible? In *Proceedings of the 2002 International Symposium on Empirical Software Engineering*, ISESE '02 (Washington, DC, USA, 2002), pp. 204–. IEEE Computer Society.

ROSENBERG, S. AND HICKIE, I. 2013. Managing madness: mental health and complexity in public policy.

RUNESON, P. AND HÖST, M. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Engg. 14*, 2 (April), 131–164.

RUNESON, P., HOST, M., RAINER, A., AND REGNELL, B. 2012. *Case Study Research in Software Engineering: Guidelines and Examples*. Wiley.

RUS, I. AND LINDVALL, M. 2002. Knowledge management in software engineering.

SEGAL, J., GRINYER, A., AND SHARP, H. 2005. The type of evidence produced by empirical software engineers. *SIGSOFT Softw. Eng. Notes 30*, 4 (May), 1–4.

SENGE, P. 2010. *The Fifth Discipline: The Art & Practice of The Learning Organization*. Crown Publishing Group.

SENGE, P. M. AND STERMAN, J. D. 1992. Systems thinking and organizational learning: Acting locally and thinking globally in the organization of the future. *European Journal of Operational Research 59*, 1 (May), 137–150.

SHNEIDERMAN, B. 2007. Creativity support tools: Accelerating discovery and innovation. *Commun. ACM 50*, 12 (Dec.), 20–32.

SIM, S. E., SINGER, J., AND STOREY, M. A. 2001. Beg, borrow, or steal: using multidisciplinary approaches in empirical software engineering research. *Empirical Software Engineering 6*, 1, 85–93.

SJOBERG, D. 2007. The future of empirical methods in software engineering research. *Future of Software ...*.

SJOBERG, D. I. K., DYBA, T., AND JORGENSEN, M. 2007. The future of empirical methods in software engineering research. In *2007 Future of Software Engineering*, FOSE '07 (Washington, DC, USA, 2007), pp. 358–378. IEEE Computer Society.

STANDISH GROUP INTERNATIONAL, I. 2001. *Extreme Chaos*. Standish Group International.

STEPANEK, G. 2005. *Software Project Secrets: Why Software Projects Fail, Apress*. Apress.

TAYLOR, R. N. 2010. Enabling innovation: A choice for software engineering. In *Proceedings of the FSE/SDP Workshop on Future of Software Engineering Research*, FoSER '10 (New York, NY, USA, 2010), pp. 375–378. ACM.

WELLS, M. AND HARRISON, R. 2000. Multidisciplinary Solutions for Multidisciplinary Problems. In *ICSE'2000 Workshop on "Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research"* (2000), pp. 1 – 6.

WILLIAMS, B. 2005. Soft Systems.

XIA, F. 2000. Borrow Fundamentals from Other Science and Engineering Disciplines. In *ICSE'2000 Workshop on "Beg, Borrow, or Steal: Using Multidisciplinary Approaches in Empirical Software Engineering Research"* (2000), pp. 1 – 6.

YIN, R. 2003. *Case Study Research: Design and Methods*. Applied Social Research Methods. SAGE Publications.