MASTER OF PHILOSOPHY

The Project Community Framework

Towards Bridging the Gap between User Centred Design and Academic Scientific Software Development

Loynton, Scott

Award date:
2015

Awarding institution:
School of Computing

The Project Community Framework: Towards Bridging the Gap between User Centred Design and Academic Scientific Software Development

Scott Loynton
University of Dundee
MPhil

Table of Contents

Table of Contents ii
List of Figures vii
List of Tables and Boxes ix
List of Abbreviations x
Acknowledgements xi
Declaration by the candidate xii
Declaration by the supervisor xiii
Abstract xiv
Associated Publications xvi

Chapter 1: Introduction 1
1.1 Motivation 1
1.2 Research aims 5
1.3 Research objectives 5
1.4 Methodology 5
1.5 Audiences 6
1.6 Structure of thesis 6

Chapter 2: Literature Review 7
2.1 Introduction 7
2.2 Background to software engineering and user centred design 7
2.2.1 Software engineering 8
2.2.2 The role of UCD 11
2.2.3 Human computer interaction 14
2.3 Usability engineering 17
2.3.1 Usability engineering methodologies 19
a) Usability engineering lifecycle 19
b) Contextual design 21
c) Usage-centered design 24
d) The scenario-based engineering process 26
e) Star lifecycle 27
f) Discount usability engineering 29
2.4 Where user centred design and software engineering meet 30
2.5 UCD and software engineering in a commercial context 36
2.6 Established integration challenges 37
1) Separation between user interface and system functionality 37

2) Cultural gap between UCD roles and engineers 38
3) UCD has to be adopted throughout the organization 39
4) Usability of UCD methods 40
5) Lack of support tools 41
6) A collection of best practices is missing 42
7) Education gap 42
2.7 SSD Environment 43
2.7.1 Scientific software developers 43
2.7.2 Funding and exploratory scientific research 48
2.7.3 Community dependencies 49
2.8 Integration of UCD and SSD 50
2.9 Summary 59

Chapter 3: The Teams and Challenges 62
3.1 Introduction 62
3.2 Background to the OME project 63
3.3 The Academic Environment for OME 64
3.4 Background to the OMERO Software Team 65
3.5 The Academic Research Software Development Environment in OMERO 68
3.6 Background to the Usable Image Project 73
3.7 The Methods of the Usable Image Team 76
3.8 Summary 80

Chapter 4: Methodology 81
4.1 Context of the fieldwork 81
4.2 My role within the Usable Image team 81
4.3 My role within the OMERO software development team 84
4.4 Strategy of inquiry 85
4.5 Ethnographic fieldwork and the construction of the methodology 87
4.5.1 Data collection 89
4.5.2 The process of analysis process 91
4.5.3 The coding and categorisation of the data 96
4.5.4 Memos 97
4.5.5 Validation of the ethnographic approach 99
4.6 Secondary Analysis 101
4.6.1 Challenges of secondary analysis 103
4.7 Summary 105

Chapter 5: Usable Image Fieldwork 106
5.1 Ethnographic analysis 106
5.1.1 Working life 106
5.1.2 Microscopy 109
5.1.3 Tools 112

5.1.4 Practices 115
5.1.5 Workflow 119
5.2 Existing scientific laboratory studies 123
5.3 Supplementary fieldwork 127
5.3.1 Co-location benefits 127
5.3.2 The contribution of UCD to the Usable Image project 135
5.4 Summary 138

Chapter 6: Scientific Software Fieldwork 140
6.1 Repositioned role of the research 140
6.2 The OMERO working week 142
6.3 Ethnographic analysis 146
6.3.1 UI development 146
6.3.2 New techniques 148
6.3.3 OMERO components 150
6.3.4 Community 152
6.3.5 Current working practice 155
6.3.6 Development tools 158
6.4 Additional activities: Community use 160
6.5 Summary of the OMERO fieldwork analysis 162
6.6 Discussion 164

Chapter 7: Forming a Framework 167
7.1 The move towards the Project Community Framework 167
7.2 How to build the Project Community Framework 172
7.3 Step I: The capture and characterisation of the Project Community 174
7.3.1 Defining the Project Team 175
7.3.2 Project Team Example 177
7.3.3 Project Team Variable Conditions 178
7.3.4 Project Team Outcomes 178
7.3.5 Defining the Project Community 179
7.3.6 The Project Community Example 180
7.3.7 The Project Community variable conditions 184
7.3.8 Project Community Mediating Role 184
7.3.9 The Project Team and Project Community Limitations 186
7.3.10 The Project Community Outcomes 186
7.3.11 Tools of the PCF 187
7.3.12 PCF Tools Variable Conditions 187
7.3.13 PCF Tools' Limitations 188
7.4 Step II: The storage of the Project Community information 188
7.4.1 Step II Example 189
7.4.2 Step II Variable Conditions 189
7.4.3 Step II Limitations 189

7.5 Step III: The process of UCD 190
7.5.1 UCD for SSD 190
7.5.2 Variable Conditions of UCD 191
7.5.3 Limitations of UCD 191
7.6 Step IV: The Project Community action and reflection 192
7.6.1 Phase One: Reflection-in-Action 193
7.6.2 Phase Two: Reflection-on-Action 193
7.6.3 Step IV Example 195
7.6.4 Step IV Variable Conditions 198
7.6.5 Step IV Limitations 198
7.7 Summary 199

Chapter 8: The Evaluation of the Project Community Framework
8.1 Evaluation strategy 201
8.2 Re-examination of studies 203
8.3 Review 1: High Content Screening (HCS) 205
8.3.1 Background 205
8.3.2 HCS fieldwork 205
8.3.3 The timeline of the HCS fieldwork 206
8.3.4 The HCS developer review discussion 212
8.3.5 Summary of HCS fieldwork 213
8.3.6 Summary of the HCS fieldwork against the Project Community Framework .. 214
8.4 Review 2: New Community Fluorescence Lifetime Imaging Microscopy 215
8.4.1 Background 215
8.4.2 Central elements 216
8.4.3 The timeline of the FLIM fieldwork 216
8.4.4 Clarification and communication within the Project Team 221
8.4.5 Exploration of an external FLIM based Institution 224
8.4.6 Information outcomes for the project visit 226
8.4.7 FLIM developer discussion 227
8.4.8 Summary of FLIM fieldwork 231
8.4.9 Summary of the FLIM review against the Project Community Framework 232
8.5 Discussion of the reviews 233
8.6 The Project Community Framework Manifesto 236
8.7 Summary 238

Chapter 9: Conclusion 242
9.1 Summary of the research 242
9.2 Summary of contributions 244
9.3 Further Testing 245
9.4 Limitations of the research 247
9.5 Future research work 249
9.6 Closing words 252

References.....251

Appendixes (on a separate disk)

Appendix 1
Appendix 2
Appendix 3
Appendix 4
Appendix 5
Appendix 6
Appendix 7
Appendix 8
Appendix 9
Appendix 10
Appendix 11
Appendix 12

List of Figures

Chapter 2: Literature Review

Figure 2.1 Usability Engineering Lifecycle (Mayhew, 1998) 20
Figure 2.2 The Contextual Design Process (Holtzblatt & Beyer, 2013) 22
Figure 2.3 Process of usage-centered design (Constantine & Lockwood, 1999) 25
Figure 2.4 The star lifecycle (Hix & Hartson, 1993) 28
Figure 2.5 Practice bridge (Seffah & Metzker, 2004) 31
Figure 2.6 Institute for Systems Biology informatics context (Killcoyne & Boyle, 2009) 46
Figure 2.7 A model of scientific end-user software development (Segal & Morris, 2011) 53

Chapter 3: The Teams and Challenges

Figure 3.1 The OMERO Architecture 64
Figure 3.2 OMERO Team Diagram over Time 66
Figure 3.3 The Location of the Projects 73

Chapter 4: Methodology

Figure 4.1 My Roles during the Research 81
Figure 4.2 Scope of UCD Led Techniques used by the Usable Image team 82
Figure 4.3 Weekly Evaluation Cycle (Adopted from Macaulay *et al.,* 2009) 83
Figure 4.4 System perspectives (Seffah *et al.*, 2005a) 84
Figure 4.5 The data analysis process (Seidel, 1998) 93
Figure 4.6 Analysing qualitative data (Seidel & Friese, 1994 cited in Seidel, 1998) 95

Chapter 5: Usable Image Fieldwork

Figure 5.1 ImageJ Main Menu 114
Figure 5.2a Worflow theme 119
Figure 5.2b Revised worflow theme 123
Figure 5.3a Cycle of Credibility (adapted from Latour, 1979) 125
Figure 5.3b Cycle of Credibility with OMERO workflow (adapted from Latour, 1979) 126
Figure 5.4 Timeline of Co-localisation through OMERO 129

Chapter 6: Scientific Software Fieldwork

Figure 6.1 The move to observe the OMERO team 140
Figure 6.2 Perspective through the Projects 141
Figure 6.3 Example screenshot of a Gantt chart (Burel, unpublished) 143
Figure 6.4 Communities built around common interests 161
Figure 6.5 The culmination of perspectives gained in the research 165

Chapter 7: Forming a Framework

Figure 7.1 The three systems perspective of SSD 171

Figure 7.2 The scope of domains of a Project Team 175
Figure 7.3 The scope of a role in a Project Team 176
Figure 7.4 Role Duties 177
Figure 7.5 Separate communities (Adapted from Fong *et al.,* 2007) 181
Figure 7.6 The Onion Model of the Project Community 182
Figure 7.7 New community growth (Schön, 1973) 183
Figure 7.8 The Role of the Project Community Mediator 185
Figure 7.9 The Reflection-in-Action cycle within the Project Team 193
Figure 7.10 The Reflection-on-Action cycle within the Project Team 194
Figure 7.11 The Reflection-in-Action as a continued learning process 196
Figure 7.12 The Reflection-in-Action as a Balanced Process 197
Figure 7.13 The Reflection-in-Action as a Conversing Process 197

Chapter 8: The Evaluation of the Project Community Framework

Figure 8.1 The approach for the PCF evaluation 200
Figure 8.2 Key timeline of development events for HCS 206
Figure 8.3 The infrastructure and functionality development conflict 211
Figure 8.4 OMERO scientific software development view of HCS 212
Figure 8.5 Key timeline of development events for FLIM 216
Figure 8.6 Roles in the FLIM work 219
Figure 8.7 Overview of the Mull institution 225
Figure 8.8 A software development view of FLIM 228

List of Tables and Boxes

Chapter 2: Literature Review

Table 2.1 Categories of scientific software developers 44

Chapter 3: The Teams and Challenges

Table 3.1 Background of the full-time developers in the OMERO Team 68
Table 3.2 OMERO Team Meetings 70
Table 3.3 Comparison between Academic Software Research and Industrial Software Projects against the OMERO project (modified Liu *et al.,* 2008) 72
Table 3.4 Roles in the Usable Image and Background to the Usable Image Team 74
Table 3.5 Usable Image Methods 76

Chapter 4: Methodology

Table 4.1 Example code study one 97
Table 4.2 Memos extracted from study one 98
Table 4.3 Memos extracted from study two 99
Table 4.4 Types of secondary analysis (Heaton, 1998) 102

Chapter 5: Usable Image Fieldwork

Table 5.1 Summary of Use of Usable Image Methods 136

Chapter 6: Scientific Software Fieldwork

Table 6.1 Summary of the software tools used by the OMERO team 143

Chapter 7: Forming a Framework

Table 7.1 Early roles identified for the potential application of the Project Community 174
Table 7.2 Core tools 187
Table 7.3 Scope of the Techniques to Enable Reflection-on-Action 195

Box 7.1 Project Community Terminology 173
Box 7.2 Defining Reflection Practice 192

Chapter 8: The Evaluation of the Project Community Framework

Table 8.1 Infrastructure and functionality identified for HCS development 212
Table 8.2 Project Community Framework evaluation review 235
Table 8.3 Manifesto summary 237

Chapter 9: Conclusion

Table 9.1 Scope of Possible Variables for Further Testing ……246

List of abbreviations

ACM:	Association for Computer and Machinery
BBRSC:	Biotechnology and Biological Sciences Research Council
CLI:	Command Line Interface
CSCW:	Computer-Supported Cooperative Work
DB:	Database
DV:	DeltaVision
EPRSC:	Engineering and Physical Sciences Research Council
FLIM:	Fluorescence lifetime imaging microscopy
FRET	Fluorescence Resonance Energy Transfer
FS:	File System
GOMS:	Goals Operators Methods Selection
GRE:	Gene Regulation and Expression
HCD:	Human Centred Design
HCI:	Human Computer Interaction
HCS:	High Content Screening
IM:	Instant Messenger
IxDA:	Interaction Design Association
JISC:	Joint Information Systems Committee
MMI:	Man-Machine Interaction
MUSE:	Method for Usability Engineering
NDIM:	N-Dimensions
OME:	Open Microscopy Environment
OMERO:	Open Microscopy Environment Remote Objects
OMII-UK	Open Middleware Infrastructure Institute
OOSE:	Object-Ordinated Software Engineering
OS:	Open Source
PCF:	Project Community Framework
PD:	Participatory Design
PI:	Principal Investigator
RCUK:	Research Councils UK
RITE:	Rapid Iterative Testing and Evaluation
ROI:	Region Of Interest
SEP:	Scenario-based Engineering Process
SPW:	Screen Plate Well
SSD:	Scientific Software Development
TIRF:	Total Internal Reflection Fluorescence
UCD:	User Centred Design
UI:	Usable Image
UML:	Unified Modelling Language
UPA:	Usability Professionals' Association
WCS:	Worm Community System

Acknowledgements

The thesis itself really has been such a life changing experience and one, which has really has given me the itch for asking more and more questions. My first thanks is to my supervisor Dr Catriona Macaulay for her support, including giving me the chance to get hooked on so many different ideas, concepts and most importantly the opportunity to do this work.

A big thanks is for all those that I have approached with just a quick question during the research. When I really have known there really is no 'quick question'. Those particularly targeted; namely Alex Carmichael, Mark Rice, Wendy Moncur, David Sloan, Paul Gault, Andrew and anyone who I may have missed. Their insights and views have all had a profound effect impact on me through this work. A special thanks goes to Xinyi who also showed me the social insights to making this research work. A big thank you for all the conversations and explanations that make people so interesting.

A big thanks goes to Jason, and all the members of the OME team. They took me under their wing and really gave me some real world problems with OMERO. And a very gracious thanks goes out to the scientists who been directly and indirectly involved through the research and who have been subject to me at times learning on the job. A special thanks goes to all those involved at the Platform in Pasteur, especially Bernd. I must thank my Aunty Dink for ferrying me round when needed it has always been much appreciated. A real thank you and a very large IOU to my Dad whose support really has gotten me here. To every car journey here there and everywhere and time and who's influence has only been a positive one – well maybe. The thesis too is dedicated to both my Mom and Gran who both helped install into me that a bit of hard work never killed anyone – they were right. Their influence is everlasting on me.

Ευχαριστίες μου προς Έλληνα συνεργάτη Νταντί Γκίκας-Μαλάκια στο έγκλημα, του οποίου η λέξη της σοφίας, συζητήσεις και αστεία έχει πάρει με σαφήνειαμέσα από τα σκαμπανεβάσματα και διασκέδαση που είναι η έρευνα. Και μου έδωσε μια Γκίκας δια βίου φίλος.

Mes derniers remerciements ne peuvent pas être dans une autre langue, et peut-être que la thèse elle-même aurait dû être rédigée en français. Le temps que m'a consacré Jean-Marie, son soutien et ses conseils m'ont donné tant d'enthousiasme pour ce travail. Et merci à Alexia, ma scientifique métamorphosée en designer d'interaction, qui m'a aidé à traverser la délicate période de fin de thèse, et tout le reste. Mais sans son support de tous les jours, je n'aurais pas pu en atteindre le bout, donc cette dernière phrase est pour toi...

Declaration by the candidate

I declare that I am the author of this thesis; that, unless otherwise stated in the text, all references cited have been consulted by me; that, except for those parts of work which are declared in this thesis to be based upon joint research, the work which this thesis records is mine; and that it has not been previously presented or accepted for a higher degree.

Scott Loynton

May 10th 2013

Declaration by the supervisor

I declare that Scott Loynton has satisfied all the terms and conditions of the regulations made under Ordinances 12 and 39; and has completed the required 9 terms of research to qualify in submitting this thesis in application for the degree of Doctor of Philosophy.

Dr Catriona Macaulay

May 10th 2013

Abstract

Information technology has brought new ideas and ways to explore data in multiple fields of science such as biology, chemistry, and physics; they all have benefited through the opportunity new technologies afford for the development of new scientific techniques. This has brought major public investment in academic SSD projects, with £250 million invested in the UK e-science program between 2001 and 2006 alone. However, there is major underlying problem: scientific software and the process of academic Scientific Software Development (academic SSD) suffers with a lack of User Centred Design (UCD). Typically, their focus is largely on the scientific software development itself.

This research investigates the gap between UCD and SSD, so that UCD may be more widely applied to the scientific software development process. The research explores how UCD may become more closely integrated into the development of scientific software, so that scientific software may have a stronger UCD focus and improved user experience for the scientists that use the software. Addressing the challenge of bridging this gap, this thesis presents insights into why this gap exists and how it might be bridged, based on a three-year field study undertaken within the context of two existing research projects – the OMERO Open Source academic SSD project (a software development team building tools for managing and analysing microscopy data in life sciences), and the Usable Image project, which was formed to investigate methods for introducing UCD into the OMERO software and to improve usability of this academic software.

The research was constructed in three phases. The first phase of the work attempted to understand the UCD-SSD gap from the point of view of UCD, via an embedded perspective developed within the Usable Image project. The second phase of the project was designed to understand the scientific software developer perspective on the gap and was undertaken from within the OMERO developer team itself. The analysis of the outcomes of these first two phases reveals the imbalance of academic SSD. The fieldwork emphasises working between the cultural gap between UCD and SSD. The third phase attempts to tackle that challenge by proposing a framework and manifesto for moving towards a more balanced approach to bring UCD and academic SSD together.

The insights arising from the deep observational fieldwork led to the development of a set of steps for SSD – the Project Community Framework. This Framework aims to support an awareness of an academic SSD's wider ecology – and it encourages teams to develop a

balanced and cognisant view of the SSD, UCD, and SSD project community, with the integration among them. The Project Community Framework was evaluated against two areas of functionality in the OMERO software from the perspective of the SSD project team, to question how the Project Community Framework as a concept may function against real SSD practice. The thesis supports the growing calls for more strongly UCD-oriented integration in SSD projects. In doing so, it proposes the Project Community Framework manifesto as a way to instil a new philosophy for academic SSD and to capture and integrate the core principles of the research of SSD, UCD, and the community of the project.

This research places the foundations for a new, pragmatic, approach to helping academic SSD teams to bridge the gap – the Project Community Framework Manifesto. It also advocates that future work explores and develops the Project Community Framework within other complex software development environments where UCD project management is critical, such as the medical context.

Associated Publications

Scott Loynton, David Sloan, Jean-Marie Burel, Catriona Macaulay. (2009). *Towards a Project Community Approach to Academic Scientific Software Development e-Science*, Users & Usability Workshop 2009.

Scott Loynton, Jean Marie Burel, David Sloan, Catriona Macaulay. (2009). *The Project Community Approach to Academic Scientific Software Development*, UK e-Science All Hands Meeting 2009.

Scott Loynton. (2009). *Towards a Usable Image: Methodology for the Design of Image Informatics Software*, Doctoral Colloquium, 2009 European Conference on Computer Supported Collaborative Work (ECSCW'09), Vienna, Sep 7-11.

Catriona Macaulay, David Sloan, Xinyi Jiang, Paula Forbes, **Scott Loynton**, Jason R. Swedlow and Peter Gregor. (2009). *Usability and user-centered design in scientific software development*, IEEE Software, 26(1), IEEE, pp.96-102.

David Sloan, Catriona Macaulay, Paula Forbes, and **Scott Loynton**. (2009). *User research in a scientific software development project*, In Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology (BCS-HCI '09), British Computer Society, Swinton, UK, UK, 423-429.

Chris Allan, Jean-Marie Burel, Josh Moore, Colin Blackburn, Melissa Linkert, **Scott Loynton**, Donald MacDonald, William J. Moore, Carlos Neves, Andrew Patterson, Michael Porter, Aleksandra Tarkowska, Brian Loranger, Jerome Avondo, Ingvar Lagerstedt, Luca Lianas, Simone Leo, Katherine Hands, Ron T. Hay, Ardan Patwardhan, Christoph Best, Gerard J. Kleywegt, Gianluigi Zanetti, and Jason R. Swedlow. (2012). *OME Remote Objects (OMERO): a flexible, model-driven data management system for experimental biology*, Nature Methods, 9, 245–253.

Chapter 1: Introduction

1.1 Motivation

This research examines the gap of User Centred Design (UCD) and academic Scientific Software Development (SSD[1]). This gap is a major problem for the e-Science[2] community – the computationally intensive scientific community that builds and uses highly distributed network environments and applications – and the communities of scientists who use them. The range of uses of scientific software is of course diverse, including, for example, the analysis of genomic data, viewing chemical data, and image processing. What unites them is the richness of the data and the complexity of the scientific work practices that the data is generated through and used for. For scientific communities, scientific software has become integrated into their everyday work practice, increasing the need for such software to be usable. Furthermore, SSD must be able to account for the needs of its future user communities – whose demands for ever-challenging data management, analysis, and computation will only grow exponentially. The need for SSD is evidenced by the huge investment being made in it. For example, the UK Governments, invested £250 million in it's UK e-Science programme between 2001 and 2006, while the National Science Foundation in the USA has made $5 million available for the cyberinfrastructure training, education, advancement, and mentoring for its 21st century workforce programme (NSF 2011).

Software design in general has to meet a number of accepted challenges in relation to highly complex SSD technology-driven environments. Norman (2010) discusses complexity can provide multiple experiences and opportunities for engagement for users. The challenge for software developers is to provide users with an experience where complexity can be exposed and managed. Norman (2010) recognises that to meet this challenge, developers must take the time and make the effort to learn the structure and the power of the design process, to reflect more deeply on how they develop software.

---

[1] Throughout this work, the term "SSD" means "academic SSD" unless otherwise stated.

[2] e-Science is computationally intensive science that is carried out in highly distributed network environments, or science that uses immense data sets that require grid computing. In 1999 John Taylor, the Director General of the United Kingdom's Office of Science and Technology, first used the term. Jim Gray say's that "*e-Science is where IT meets scientists*" (Bell *et al.,* 2009). For more information, see http://www.e-science.stfc.ac.uk/.

The concept of UCD in the development of usable software has been widely acknowledged for a long time. UCD emerged during the 70s and 80s. Grudin (1991) highlights how during this time participatory design emerged throughout Europe, which involves users to collaborate in the design process as full development team members. The term UCD was coined in the work by Norman and Draper (1986) in their work on user centred system design. Since then it has been through the progression and adoption of technology in the workplace, in the home and with ubiquitous mobile devices that an understanding of the user environment has presented much greater challenges in user interaction and the need for user involvement in software design has become a necessity.

Despite the role of UCD and its contribution to the usability of software, its integration into the software development process still presents many challenges. Existing research has specifically investigated the integration gap between software engineering and UCD (Seffah *et al.,* 2005; Seffah *et al.,* 2009). These collections of approaches to UCD and software engineering integration differ greatly. Techniques vary from models of evaluation (Hvannberg, 2009; Tarpin-Bernard *et al.,* 2009) alternative approaches have taken the step to cross over UCD into the established software engineering lifecycle (Hix & Hartson, 1993). Other example proposals for integration have offered more *ad-hoc* styles (Anderson *et al*., 2001; Radle & Young, 2001). Throughout this research there is no single best approach for UCD identified for the variety of software development projects.

Aikio (2006) also shows how finding a generic solution for the integration between UCD and software engineering is difficult because of the variable factors for any given integration case in an organisation (Aikio, 2006). What she acknowledges from her work is the need for further background research and empirical evidence about the issue of UCD integration.

Academic scientific software is situated where accepting the “doing” of science is a growing more costly. The objectives of national UK scientific organisations, such as the Wellcome Trust, the research councils, and charities like Cancer Research UK, are important to society – they address complex research questions about how the brain works, they combat infectious diseases, they investigate ageing and chronic diseases, and they help to understand how cancer starts and develops. The Wellcome Trust alone in 2010 funded £530 million in the combined areas of science (£436 million),

technology transfer (£59 million), and medical humanities and engagement (£39 million) (Wellcome Trust Annual Report and Financial Statements 2010). The cost allocated by the Wellcome Trust to technology transfer is an indication and recognition by the Wellcome Trust of the increasingly vital role technology plays in science. Similarly, the EPSRC has made significant investment in the UK e-Science programme, funding 100 projects worth more than £250 million since the programmes inception in 2001 (EPSRC 2011).

The role technology plays in everyday scientific investigation and practice is critical for scientific research ranging from data analysis, to managing workflows (Howison & Herbsleb, 2011). Poorly designed technology can and will frustrate these ambitions. The scientific environment today is a fast-moving cutting-edge world where the technology must look to constantly match the same pace. SSD is, in the proverbial phase, a moving target. It must evolve continually to respond sufficiently to the questions science asks. The scientific communities expect and require SSD to meet these needs. This contrasts starkly with other complex software such as that used in back-office work, where despite the underlying complexity of the operational practices, it rarely evolves too quickly, and when upgrades arrive they are integrated into existing processes.

Usability and the associated challenges facing the scientific software community are gaining an awareness. Such work by the UK e-Science Usability Task Force (e-Science Usability Task Force 2011) has been involved in investigating the development and effective deployment of e-Science systems. The challenges they have identified cover four areas:

- **Global communities.** How do we maximise the use of e-Science technologies and applications to support new forms of scientific community?

- **Trust and ethics.** How do we handle the ethical and policy issues to emerge from the e-Science infrastructure?

- **Knowledge production.** How do we exploit an e-Science infrastructure and e-Science techniques to support scientists' expertise, new research methods and new forms of knowledge production?

- **Design, assessment and management.** How can we best assess e-Science technologies and applications, and use this assessment to guide the design and management of these Systems?

(e-Science Usability Task Force 2011)

I would argued that many of these questions raised by the e-Science Usability Task Force (2011) can be aided by providing more usable scientific software. Various researches have investigated the role of UCD in scientific projects (Letondal, 2005; Macaulay *et al.,* 2009; Thew *et al.,* 2009; De Roure & Goble, 2009). Existing work has also acknowledged that UCD can contribute to the design and development of scientific software (Javahery *et al.,* 2004; Schraefel *et al.,* 2004). Yet, despite this, usability issues are still often overlooked in SSD (Letondal 2005; Macaulay *et al.,* 2009; Thew *et al.,* 2009; De Roure & Goble, 2009).

Consequently, the requirement for understanding the UCD and software-development integration process in this context is just as significant. The existing UCD research in the SSD context, the question of integration of UCD in SSD has largely gone unaddressed. This may be due to the varied nature of the development of scientific software projects. The work by Segal (2008) identifies three levels of scientific software development contexts: scientists developing software for their own laboratory, software engineers developing software in partnership with scientists, or more experienced scientists developing software but without the expertise and understanding of software engineering practices.

Subsequently, this underlines the current permutations of SSD and furthers still the potential for variation for the integration between UCD and the scientific software development process. The requirement for scientific software is the ability to adapt to a rapidly changing world that includes increasingly complex datasets and analysis. This research therefore places the need, for UCD to be examined from the perspective of understanding the issues of integrating UCD into SSD.

With all this in mind, this research raises the question: how can we migrate the existing and well-established practices of UCD into the development of scientific software in order to help support the challenges of SSD? This research aims to understand the

landscape of scientific software and understand the UCD process from operating within a UCD project for SSD. It is therefore this issue that underpins the goal of this thesis.

1.2 Research aims

This research aims to explore and address the gap between UCD and academic SSD. Building on this understanding, it proposes a framework for integrating UCD more closely into the development of scientific software. The research questions are as follows:

1. *Why is so much of academic SSD still unusable and/or poorly accepted by scientists?*

2. *How is SSD undertaken in academic contexts (investigating SSD in commercial contexts is deemed outside the scope of this work)?*

3. *How can the uptake of User Centred Design philosophies, methods and thinking in the application of academic SSD be improved?*

1.3 Research objectives

The objectives of this work are to develop a clear understanding of how academic SSD happens, to illustrate where UCD does and does not drive academic SSD, and to operationalise my findings in the form of tools for bridging the gap between UCD and academic SSD.

1.4 Methodology

The researcher adopts an ethnographic approach that comes from being situated within both the Usable Image project and the Open Microscopy Environment Remote Objects (OMERO) software project in the University of Dundee. This approach has been chosen to allow for flexibility for the research work between the two projects. The research work has used ethnography for conducting a primary and secondary qualitative analysis of the fieldwork from within the Usable Image project and the OMERO project. The

methodology of the research is detailed in chapter 4 and the analysis of the research data is documented in chapters 5 and 6.

1.5 Audiences

The primary audience for this research is the academic community involved in SSD – developers, designers, UCD specialists. However, it is hoped that the insights and ideas discussed and developed here will be of interest to colleagues involved indirectly in SSD, the scientists themselves and the bodies that traditionally fund such research.

1.6 Structure of thesis

The motivation, overview and questions for the thesis have been presented here in Chapter 1. Chapter 2 investigates extensively the background of the thesis in the form of a literature review. This literature review covers previous work on software engineering and its relationship with UCD. The established integration challenges are then examined to understand about the UCD and software engineering gap, the subsequent context of SSD and challenge this brings for integrating UCD and SSD are finally explored. Chapter 3 provides the environment and context in which the research is being conducted, with the background to the two projects that this research was embedded in – the OMERO scientific software development project and the Usable Image project (a project to introduce UCD into OMERO). Chapter 4 presents the ethnographic methodology of the thesis and the steps of the analysis for the fieldwork carried out in Chapters 5 and 6. Chapter 5 provides the analysis and initial findings of the Usable Image fieldwork, while Chapter 6 presents the analysis and subsequent findings carried out in the OMERO fieldwork and then concludes with the findings and implications from both perspectives of the fieldwork. Chapter 7 takes the findings and proposes a new way of approaching SSD with the view of UCD; the Project Community Framework. Chapter 8 explores applying the concept of the Project Community Framework and the development of a manifesto for academic SSD. Finally, Chapter 9 reflects on the contributions of the thesis to the research community and the implications of these for future work on bridging the gap between academic SSD and UCD.

Chapter 2: Literature Review

2.1 Introduction

This literature review will examine the broader context and background of the gap between UCD and software engineering. The review considers both of the fields and the challenges of integrating them. The existing literature is examined in relation to the established challenges of working integrating UCD and software engineering. The range of techniques and approaches used to bridge the gap between UCD and software engineering are reviewed to underline the variety of approaches from usability engineering from the usability engineering cycle (Mayhew, 1996) to discount usability engineering (Nielsen, 1993b). A discussion then presents the more specific challenges to UCD and software engineering in the context of the SSD environment. The review concludes that UCD has been acknowledged in SSD. However, the application of UCD to SSD does come with the challenges that exist between UCD and software engineering integration, and in addition ones from working within a complex scientific domain. The literature review concludes by discussing the increasing acknowledgement of the gap between UCD and SSD. This literature review will be restricted to the position of integration of UCD and software engineering and the focus on the gap between UCD and SSD. Consequently, the use and purpose of UCD will be presented within this context.

2.2 Background to software engineering and user centred design

Software engineering and UCD stem from very different foundations and consequently form different perspectives of the software development process. Historically, software engineering as a field emerged first and precedes any UCD work by several years. This has allowed software engineering to mature and become much better practiced, whereas UCD has not yet reached this level of saturation. Subsequently, software engineering is relatively well established, whereas UCD is still working towards a similar level. The first section of this review gives an overview to the history of both these fields; in this, the role of Human Computer Interaction (HCI) and usability engineering are both explored because of their relationship to the integration of UCD and software engineering.

2.2.1 Software engineering

The term ‘software engineering’ was introduced in 1968/69 as a model for the field of software development (Naur & Randell, 1969). The term was selected for the reason *“to imply the need for software manufacture to be based on the types of theoretical foundations and practical disciplines that are traditional in the established branches of engineering”* (Naur & Randell, 1969, p. 8). The original document that documents the formation of software engineering, presents many interesting insights into the early challenges software engineers faced. The following extract has been taken from the Software Engineering report (Naur & Randell, 1969) to highlight the difficulties in the preliminary software development process. The extract has been chosen because it stresses the investment being made in software development and the repeated mistakes that software engineers were making once a software project has been completed, because nothing is learned. It is out of this that a call for more careful consideration for software development is made:

> ***Graham**: Today we tend to go on for years, with tremendous investments to find that the system, which was not well understood to start with, does not work as anticipated. We build systems like the Wright brothers built airplanes.*
>
> 1) *Build the whole thing, push it off the cliff, let it crash, and start over again.*
> 2) *Of course, any new field has its growing pains:*
>
> ***Gillette**: We are in many ways in an analogous position to the aircraft industry, which also has problems producing systems on schedule and to specification. We perhaps have more examples of bad large systems than good, but we are a young industry and are learning how to do better. Many people agreed that one of the main problems was the pressure to produce even bigger and more sophisticated systems.*
>
> ***Gill**: It is of the utmost importance that all those responsible for large projects involving computers should take care to avoid making demands on software that go far beyond the present state of technology unless the very considerable risks involved can be tolerated.*
>
> (Naur & Randell, 1969)

In the early period of software development, there was very little division of labour or specialisation in the software development industry (Mayhew, 2008). A programmer could cover all aspects of software development from functional analysis and project management to software testing, user interface design, and user support. Since the early days of software engineering, the software industry has evolved in many ways. Reed (2005) highlights how software is now part of the process when we fly or drive, so in many ways our lives rest on safe and reliable software. But for many contemporary projects, parts of these original statements of the challenges faced in 1968 still hold true.

> "*To paraphrase the rag trade joke, we may have gone from cottage industry to cottage industry in three generations!"*
>
> (Reed, 2005)

The acclaimed work by Brooks (1995) comments on software development in that there is no single development, either in technology or in the management that on its own can provide any magnitude of improvement in productivity, in reliability, in simplicity. Brooks (1995) identifies four irreducible elements of software systems: complexity, conformity, changeability, and invisibility. Because of these factors, he implies that developing software will always be a major challenge.

The traditional software development methodologies that have emerged for managing these complexities fall into two distinct categories: either more modern lightweight approaches or more planned heavyweight approaches. The following methodologies are considered to be more heavyweight methodologies, following a more conventional hierarchical approach where the challenges are defined up front and problems can be further refined and addressed in succession. These methodologies range from Waterfall (Royce, 1970), Spiral Model (Boehm, 1986), Unified Process (Bergstrom & Raberg, 2004), Cleanroom (Prowell *et al.,* 1999), Rapid Application Development (Kerr & Hunter, 1993), V-Model (Pressman, 2005), and Test Driven Development (Beck, 2003).

Software development methodologies have evolved to take on more reactive approaches. These methodologies have few principles with the aim to provide more flexible techniques for software development. Such techniques include Rational Unified Process (Kruchten, 1998), Lean (Poppendieck & Poppendieck, 2003), Iterative (Larman & Basili, 2003), and Agile (Beck *et al.,* 2004). There are also more formal mathematical

software development models including B-Methods (ClearSy, 2011), Petri Nets (Desel & Juhás, 2001), and Finite State Machines (Wagner *et al.,* 2006). Formal methodologies are traditionally used in areas where the software is safety critical, such as DO178B (a standard on Software Considerations in Airborne Systems and Equipment Certification) that requires formal methods to ensure safety checks).

There has been some growth in SSD in the area of Agile software development techniques (Ackroyd *et al.,* 2008; Kane *et al.,* 2006; Wood & Kleb, 2003). This has reflected the growth in the adoption of Agile throughout the entire software development community. The questions of suitability of the Agile methodology for SSD has nevertheless been raised in work by Crabtree *et al.,* (2009). The work by Crabtree *et al.,* (2009) questions how to successfully understand and evaluate the various interpretations of the Agile methodology in scientific projects and recognise it as being a very difficult one. Crabtree *et al.,* (2009) also emphasise that SSD inherits the common problems of software development in terms of the project goals, size, and culture playing an important role in adopting certain methodologies and practices in SSD.

These general problems of the selection of a software development methodology can be traced back to the evolution of methodologies. The emergence of many methodologies has come from work within specific domains. Such an example is the Waterfall model, which emerged through large-scale systems development, and the Spiral model, which emerged through the software defence industry (Boehm, 1988).

The work by Yourdon (2007) cites how Barry Boehm found that while working with Win Royce the Waterfall software methodology worked well within the domains of aerospace and military systems. The was due to how the user's requirements were clearly defined with military and aerospace systems and the requirements could be made up front. However, Barry Boehm acknowledges how the ability to have the requirements clearly defined has slowly diminished. The work by Vessey (1997) describes the problem of software engineering methodologies problem further and highlights how they were never characterised. Consequently, it affected both the academic and the software engineering field. Vessey (1997) adds that this computing philosophy occurred during the late 1960s when computer science and software engineering required the ability for domain-independent techniques, methods, and

paradigms to be taken seriously. This led to the demand for a broad application of computing and software techniques, methods, and paradigms.

Kelly (2007) discusses the effect of the range of distinctive development practices in software engineering that allows scientific software developers to take many different approaches in developing scientific software. She also underlines the growing need for more adaptable techniques for software development in the scientific context.

The problem of a lack of specification from the emergence of software engineering methodologies has caused many different approaches to be taken instead of questioning what approach may work best. The context of SSD is a further example of this problem. By highlighting the history of software engineering, SSD serves to demonstrate the underlying challenges for software development and how these are applicable for SSD. In addition, the evolution of software development methodologies has allowed software developers to take multiple paths for software development.

2.2.2 The role of UCD

The first use of UCD was in the work "The Design of Everyday Things" by Norman (1986).

> *"User centred design emphasizes that the purpose of the system is to serve the user, not to use a specific technology, not to be an elegant piece of programming. The needs of the users should dominate the design of the interface, and the needs of the interface should dominate the design of the rest of the system."*
>
> (Norman, 1986)

In this aspect the UCD term by Norman (1986) aimed to expose that in any software development context the requirements of the interface are central to the design of the system. The work by Gould and Lewis (1985) is also cited as a central reference to the formation of UCD principles. These include the principles of early and continual contact with users, quantitative usability criteria, evaluations, and iterative design (Keinonen, 2008). Other work by Katz-Haas (1998) defines UCD both as a philosophy and a process. This point is also argued by Karat (1996): UCD is a philosophy that

underlines the commitment to the users. However, he further raises the concern that as an abstracted philosophy it provides a vast amount of tools and techniques and can lead to many misinterpretations. There are more recent principles of UCD outlined in the work by Gulliksen *et al.,* (2003) and Maguire (2001):

- An appropriate allocation of function between user and system
- Active involvement of users
- Iterations of design solutions
- Multidisciplinary design teams

The range of factors and the established principles around UCD are set out in the framework by Gulliksen *et al.,* (2003). This framework outlines a prototypical UCD approach, utilising the UCD methods presented.

1. User focus (Gould *et al.,* 1997; ISO 13407 1999)
2. Active user involvement (Nielsen, 1993; Gould *et al.,* 1997; ISO 13407 1999)
3. Evolutionary systems development - the systems development should be both iterative and incremental (Boehm, 1988; Gould *et al.,* 1997)
4. Simple design representations (Kyng, 1995)
5. Prototyping (Nielsen, 1993; Gould *et al.,* 1997)
6. Evaluate use in context (Nielsen, 1993; Gould *et al.,* 1997)
7. Explicit and conscious design activities (Cooper, 1999)
8. A professional attitude (ISO 13407 1999)
9. Usability champion - usability experts should be involved early and continuously throughout the development lifecycle (Kapor, 1996)
10. Holistic design - all aspects that influence the future use situation should be developed in parallel (Gould *et al.,* 1997)
11. Processes customisation - the UCSD process must be specified, adapted, and/or implemented locally in each organisation
12. A user-centred attitude should always be established

(Gulliksen *et al.,* 2003)

The introduction of the term ‘UCD’ has benefited the design of information technology and interactive systems, helping to introduce a user awareness and consideration to the design process. As a practice, it has gained UCD industry acceptance and uptake (Mao & Vredenburgh, 2005). The adoption of UCD in industry has in turn created several

professional usability organisations - e.g. The Usability Professionals' Association (UPA) and Interaction Design Association (IxDA) as industry has further developed the term UCD. The UPA defines UCD as "*an approach to design that grounds the process in information about the people who will use the product*" (UPA 2010).

Despite the broad scope of UCD definitions, some similarities exist for the term. This was a problem also reflected in the field of HCI and is discussed in the following section (2.2.3). However, despite this the work by Karat (1997, p.38) summarises a key point for moving forward with the term UCD:

> "*I suggest we consider UCD an adequate label under which to continue to gather our knowledge of how to develop usable systems. It captures a commitment the usability community supports - that you must involve users in system design - while leaving fairly open how this is accomplished*."

Iivari and Iivari (2006) further state how the concept of UCD is still unclear. They note that UCD has picked up ideas from many different sources such as prototyping, evolutionary delivery, socio-technical design, user participation, participatory design, and usability engineering. From this, UCD has taken on more meaning. This broader interpretation of the term UCD has led to some criticism, as this general and non-specific definition, in practice, ends up being a concept with no real meaning (Kujala, 2003; Gulliksen *et al.,* 2003).

The term UCD is used throughout this thesis and acknowledges that UCD shall continue to evolve. This recognizes a similar position to Karat (1997), who accepts that you must involve users in the systems design process but yet be open to how this may be accomplished.

For this thesis, UCD shall be fundamentally regarded as a philosophy throughout the entire systems development process. The term from this perspective represents the holistic view for the goals of this research. I have actively chosen UCD to represent a holistic view, knowing that it must operate and evolve in this context.

Given this interpretation, the following two sections focus and expand on the fields of HCI and usability engineering. These two fields significantly contribute to UCD and

software engineering integration, so several of the techniques and methods from the two fields and their relevance to UCD are explored.

2.2.3 Human computer interaction

Human Computer Interaction (HCI) is an interdisciplinary field that joins computer science, social, cognitive, and behavioural science, and human factors engineering. It is a field that serves to integrate the technical development with planning and assessing impact of the development process. The Association for Computer and Machinery (ACM) gives a definition for HCI "*Human Computer Interaction is a discipline concerned with the design, evaluation, and implementation of interactive computer systems for human use and with the major phenomena surrounding them*" (ACM 2009). In the definition by ACM it reveals three key components of design, evaluation, and implementation and covers how HCI extends to the context surrounding the systems. Dix *et al.,* (1993) defines HCI as "*the study of people, computer technology, and the ways these influence each other. We study HCI to determine how we can make this computer technology usable by people*." The definition by Dix *et al.,* (1993) captures the goal of HCI of making computer technology usable through the study of between people and computer technology.

The growing challenges of software engineering in the 1970s led to the acknowledgement that the way forward for computing was a better understanding of users. It was this that HCI rose to meet with its convergence of several disciplines. In providing the necessary support for the systems development process, HCI comprises methods, models and various usability evaluation approaches and tools. The principles set out from UCD are embodied in HCI and enabled through the methods and techniques that have subsequently been formed in the HCI field. Myers (1998) has reviewed the history of HCI from a technological perspective and he discusses how HCI began with research in direct manipulation in academia as early as 1960. He also points out that the research work in the commercial context for HCI did not begin until 1970, and the introduction of HCI in commercial products was not realised until 1980.

The work by Carroll (2009), a major contributor to the field of HCI, has documented the rapid evolution of HCI and how the field continues to diversify and outgrow all

boundaries. The field has expanded to encompass visualisation, information systems, collaborative systems, system development process, and many areas of design. As a result of the development of HCI, it is now less focused with respect to core concepts and methods, problem areas, and assumptions about infrastructures, applications, and types of users (Carroll, 2009). The expansion of HCI from its focus on individual behaviour and singular work applications to a much wider emphasis on social and emotional behaviour and ubiquitous context-aware settings, applications covering games, e-learning, and e-commerce. This is also a reflection of the growing interdisciplinary nature of HCI. While historically the domain was one involving computer scientists with one or two more fields such as human factors or psychology, there has been continued growth into new fields such as information science, art, and design (Grudin, 2006). These new disciplines that continue to drive the HCI agenda have opened up new research approaches. Further still with the research questions presented and scope of the variables involved, the older reductionist methods based in the laboratory provide only a limited insight into the problem (Shneiderman, 2008).

In a similar situation to UCD, HCI has suffered with a lack of consistent development (Diaper & Sanger, 2006). Diaper and Sanger (2006) say this is because of a general lack of agreement as to 1) what HCI should be, 2) what HCI can do, 3) how HCI can do it, and 4) how HCI can be allowed to do it. They argue that the first of these four points has almost been completely ignored. Consequently, HCI has developed by concentrating on the last three points. This has led to HCI being developed away from real problems or new technologies. They also consider how significant design decisions are made before the HCI issues. They point out that the division between systems development and HCI derives from the historical issue of software engineering and HCI developing separately.

The HCI methods and techniques can be compared against the principles of UCD. These principles were outlined by Gould and Lewis (1985) as mentioned in the previous UCD section (see section 2.2.2). An early method developed was task analysis (Diaper, 1989; Kirwan & Ainsworth, 1992; Preece, 1994). Task analysis observes and understands the work to be done by the users in order to inform the design process and to help provide an early focus on the users and gain their involvement. Other early methods attempted to provide interface guidelines. An example was the research by Shneiderman (1987), who formed the ‘Eight Golden Rules of Interface Design’, and

later the work by Molich and Nielsen (1990), who formed the heuristic evaluation approach. The specific work by Nielsen (1993) formed 10 rules called the usability heuristics that simplified and explained common problems in interface design. Nielsen (1993) recommends that interface designers should respect the following set of usability heuristics:

1. Simple and natural dialogue
2. Speak the user's language
3. Minimise the user's memory load
4. Consistency
5. Feedback
6. Clearly marked exits
7. Shortcuts
8. Good error messages
9. Prevent errors
10. Help and documentation

(Nielsen, 1993)

These early methods – task analysis and a heuristic checklist – were formed in HCI were due to its early roots in cognitive science. Work by Norman (1981) and Rasmussen (1986) produced early theories that were based on understanding and categorising human error.

In the growth of HCI and the wider application of technology to the workplace, HCI turned to other disciplines. The use of ethnography in the HCI design process is an example of the turn to sociology led methods to provide more detailed accounts of the user. The use of ethnography and its roots in anthropology help to provide HCI into understand complex human practices and contexts. A core belief in ethnography is that to gain the necessary understanding of the world you know little about, you must emerge within it firsthand (Blomberg *et al.,* 2007). Dourish (2006) cites that the adoption of ethnography within HCI is down to two trends: first, the emergence of Computer-Supported Cooperative Work (CSCW) as an area of inquiry. This placed an increasing emphasis on questioning the social organisation of activity, so it required methodological approaches that could understand social organisation. The second trend was the rise of the Participatory Design (PD) movement. Especially popular in Scandinavia where it has its roots, this movement has gone on to have a global

influence in HCI. Consequently, through CSCW and PD, the use of ethnographic methods became more familiar to the HCI field. Existing studies have shown the value of ethnography in various domains from homes (Crabtree & Rodden, 2004), workplaces (Newman & Landay, 2000), and education (Wyeth, 2006). The work by Suchman (1987) is influential in its example of the advantage of ethnography in HCI. Her work provides insight into the communication breakdown and task failures between a user's constructed model and the expert model of a photocopier machine.

The work by Anderson (1994) suggests that ethnography can challenge and reveal the taken-for-granted assumptions of a problem framework. Critically, what Anderson identifies as an ethnographic representation is that it "*seeks to tell a story which plays through these antinomies and in so doing, synthesises them*" (Anderson, 1994).

However, various research documents the problems for the system design process as it struggles to fully use such rich material that ethnographic work provides (Hughes *et al.,* 1995; Dourish, 2006). The work by Dourish (2006) highlights this debate and discusses that the implications for design are not always a necessity for the use of ethnography in HCI, despite the recognition that design practice has been both successful in design terms with such work by Hughes *et al.,* (1993) and Bentley *et al.,* (1992). Dourish (2006) says that if such an approach is taken, then it can restrict the requirement's capture. His argument for this is the work of the ethnographer is more than a collection and that the ethnographic process is an interpretive and analytic practice. In addition, ethnography's emphasis is on social facts so any implications for design can inappropriately emphasise technology over practice. Critically, the contribution of ethnography to HCI is as a method to provide further understanding of what it says happens, and ideas for thinking about social life. This attitude is embodied in the ethnographic outlook as argued by Randall *et al.,* (1995), who themselves note that ethnography provides a view of seeing the social world from the point of view of participants.

2.3 Usability engineering

Usability engineering emerged partly in response to the need to integrate UCD concepts and techniques with software engineering (Faulkner, 2000). This was a direct result of the lack of usability in software development. The work by Reiterer (2000) underlines

usability engineering as a philosophy that incorporates HCI into software engineering. The level of closer integration between the two fields of UCD and software engineering is also discussed by Karat and Dayton (1995) to the extent that they view usability engineering not as a field separate from the base engineering activity, but a "*special perspective on that activity*". Seffah and Metzker (2008) notes that a large proportion of development release costs occur because current software engineering methodologies lack attention to user needs and usability requirements, testing and requirements validation, design prototypes, and functional systems with end-users before, during and after development.

Seffah and Metzker (2009) highlight how the term usability engineering is inconsistently defined throughout the literature. They cite this to be down to the changeability and ambiguity of the terms 'usability engineering', 'user interface design and development', 'UCD', and 'user interaction design'. These terms are described here: Faulkner (2000) defines usability engineering as "*an approach to the development of software and systems which involves user participation from the outset and guarantees the efficacy of the product through the use of a usability specification and metrics*". Hix and Hartson (1993) define it as "*a process through which usability characteristics are specified, quantitatively and early in the development process, and measured throughout the process*". Nielsen (1993) emphasizes this aspect when defining usability engineering as a set of activities that take place throughout the lifecycle of the product. The significant step of defining activities for the software occur at early stages before the user interface has even been designed. Preece *et al.,* (1994, p. 722) describe it as "*an approach to system design in which levels of usability are specified and defined quantitatively in advance, and the system is engineered towards these measures, which are known as metrics*".

These collections of terms cover activities that can be emphasised at various points of a lifecycle and emphasize metrics in different ways. The changeability and ambiguity of the terms in usability engineering again underlines how it suffers in a similar way to both UCD and HCI in terms of the clarity of the term. This is significant for this research and the literature review as the information will be reviewed later on in section 2.4.

2.3.1 Usability engineering methodologies

The goals of usability engineering to support systems design provides a wide range of methods to support the integration of UCD with the software development process. Therefore, the following section examines the principles of usability engineering methodologies and reviews them. Again, like the inconsistency of the terminology for usability engineering, usability engineering methods and techniques are also cited as being inter-changeable (Seffah & Metzker, 2009). For this reason I have adopted the same approach taken by Seffah and Metzker (2009) and used the terminology set out by the IEEE standard glossary of software engineering (ISO Std.610.12, 1990).

1. An integrated set of policies, procedures, rules, standards, techniques, tools, languages, and other methodologies for analysing, designing, implementing, and testing software.
2. A set of rules for selecting the correct method and underlying process and tools.

Example usability engineering methodologies include the following: (a) usability engineering lifecycle (Mayhew, 1996); (b) contextual design (Beyer & Holtzblatt, 1998); (c) Usage-Centered Design (Constantine & Lockwood, 1999): (d) Scenario-Based Engineering process (SEP); (e) star lifecycle (Hix & Hartson, 1993); and (f) discount usability engineering (Nielsen, 1993). Other example methods not reviewed in this thesis are the Rapid Iterative Testing and Evaluation (RITE) method (Medlock *et al.,* 2005), the method for usability engineering (MUSE) (Lim & Long, 1994), and InterMod (Losada *et al.,* 2012). While a broad scope of methodologies and variation is acknowledged, reviewing them in detail goes beyond the scope of this thesis. I have chosen to review the aforementioned six methodologies ((a) to (f)). I will also question the approach for managing the integration of UCD and software engineering.

a) Usability engineering lifecycle

The usability engineering lifecycle forms a comprehensive guide to a set of tasks for a development process that integrates usability tasks into the software development (Mayhew, 1996). Her work identifies its position in relation to object-ordinated Software Engineering (OOSE), which was formed by Jacobson *et al.,* (1992). She recognises that although OOSE has a UCD-led philosophy, it has several shortcomings in not addressing well-defined usability goals and techniques. Based on this the usability engineering lifecycle identifies four phases that make up the lifecycle these

are: scoping, requirements definition and design, development, and installation. These four steps are shown in Figure 2.1. This figure 2.1 shows how the middle stage of design testing and development is the largest and involves many subtasks. Mayhew recognises that people will not require the entire structure and she suggests that steps can be omitted if they are not necessary. Figure 2.1 also shows the significant position of the style guide. The style guide is used to provide information on the standards that will be applied across the software. It is formed from the early goals in the requirements phase. The subsequent guide may then be used to support the further usability goals of the project (Preece *et al.,* 2002).

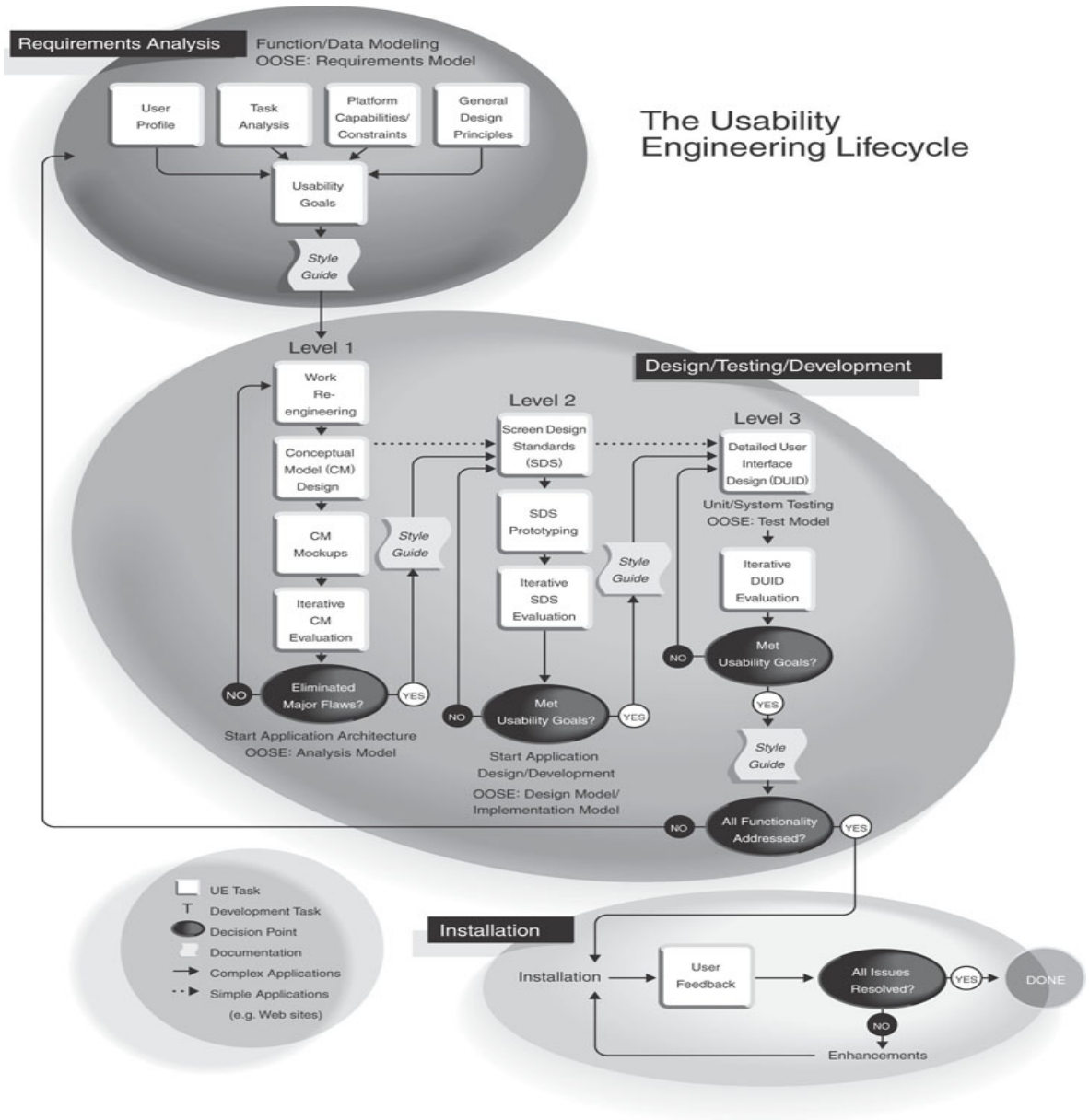

Figure 2.1:Usability Engineering Lifecycle (Mayhew 1998)

The methodology provides a detailed step-by-step approach for the integration of UCD and software engineering. In managing the integration of UCD and software engineering, the methodology serves to align the usability tasks against the traditional software development tasks. This methodology is significant in our context as the work by Mayhew identifies several successful projects where it has been applied. However, this is not without concern by Mayhew as re-designing the software development process around UCD issues can frequently pose problems to the organisation culture of software engineering organisations. Software development teams are frequently lacking the knowledge to conduct UCD activities, so this lack of UCD expertise further exacerbates the problem.

b) Contextual design

The contextual design process was formed by Beyer and Holtzblatt (1998). It is a structured design process that provides methods to collect data about users in the field, interpret and to consolidate that data in a structured way. It then uses the data to create prototypes and iteratively test and refine the design concepts with the users. Contextual design empathises the behavioural aspects of the system design process. The core principle behind contextual design states that any technology, product or system must support and extend its users' work practice. Holtzblatt gives a definition of contextual design: "*A set of techniques to be used in a customer centred design process with design teams. It is also a set of practices that help people engage in creative and productive design thinking with customer data and it helps them co-operate and design together."* (Preece *et al.,* 2002, pg, *313*)

The eight steps of contextual design are illustrated in Figure 2.2 and explained below with reference to the work by Holtzblatt and Beyer (2013).

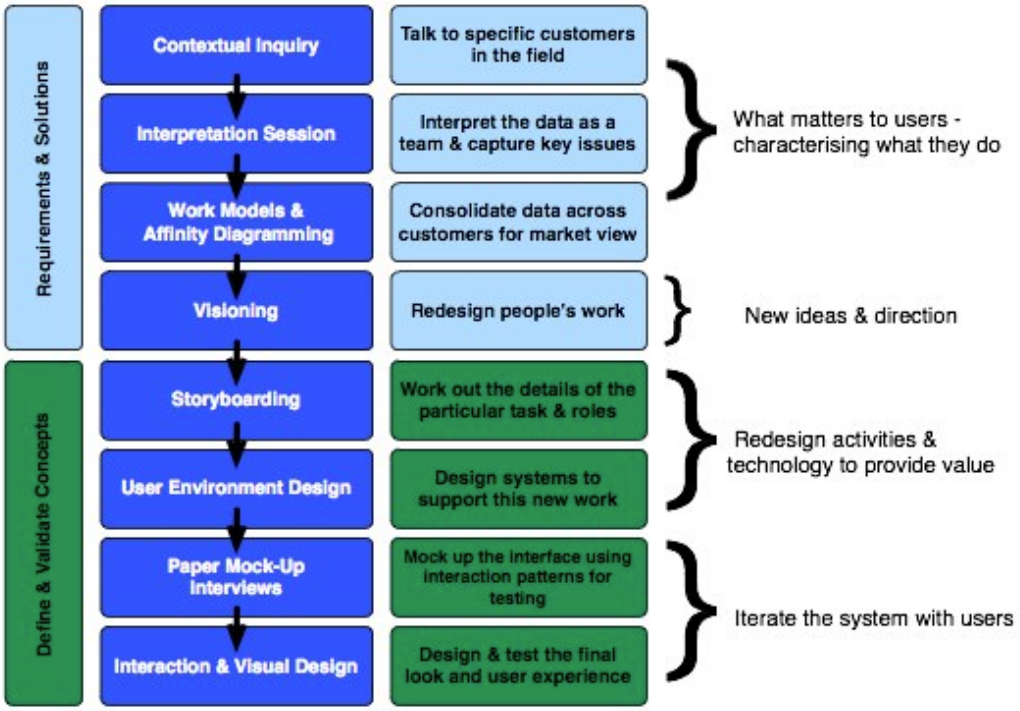

Figure 2.2: The Contextual Design Process (Holtzblatt & Beyer 2013)

The first step is **contextual inquiry**. This has the goal of understanding who the users really are. Contextual inquiry involves the designer carrying out interviews, but the interviews take place at the users' workplace to create a deeper understanding of their actions and motivations. The second step is **interpretation sessions**. This brings the design team together to listen to the interviews from the contextual inquiry so that they can capture insights from these and understand their relevance to the design problem. Through the discussion the entire team can learn about the data and bring their own perspective. The third step is **flow models and affinity diagramming.** The flow model is made up of a series of models to capture the work of individuals and organisations in diagrams. Five different diagrams provide the contextual designers with several different viewpoints on how the work is carried out. These diagrams can be summarised as follows:

1. The flow model – This model aims to communication and coordinate between people to accomplish their work.
2. The cultural model – This model captures the policies that restrict how work is done and how users form workarounds to these problems.
3. The sequence model – This model details the task required to complete the work. It can help uncover the different strategies users can take to complete a task.

4. The physical model – This model highlights how the physical environment supports or obstructs the work, as well as how the users organise their environment.
5. The artifact model – This model demonstrates the outcome of artifacts that are created from the work. The artifacts are how users think about the work, the concepts they use, and how they are organised to complete the work.

The **affinity diagram** is another method to visualise the scope of user problems and is where each observation from the contextual inquiry is written down on a post-it note then placed on a wall. The observations can then be grouped together based on their relationships. The fourth step is **visioning**. This step documents a story of how the users will carry out the new work with the new system. The vision is made up of the system, its delivery, and system support structures to make the new work practice successful. **Storyboarding** is the next step. It defines the details for the function, behaviour, and structure of the proposed system. The storyboards demonstrate the new steps a user will take through the system. The sixth step is the **user-environment design**. This step captures the plan of the new system showing how each part of the system and how it supports the users' work. It also highlights how the user gets to and from other parts of the system without securing the structure to any particular user interface. The penultimate step is **paper prototyping**. This step develops rough mockups of the system using notes and hand-drawn paper to represent the system interface. The final step is **interaction and visual design**. This process allows the design team to develop and test the interaction options with users.

The work by Rockwell (1999) provides an account of the development of a new software product, Ignite-UX, at Hewlett-Packard. This work describes how the contextual design process can provide integration into the software development process. Rockwell discusses each step from the contextual design process and he also adapts the prototyping step using remote prototype testing instead of the paper prototyping as outlined in contextual design. He does not cite the use of two work models (artifact models and culture model), the affinity diagram, storyboarding, and the user environment design. So Rockwell does demonstrate adaptation of the contextual design technique for his own use.

One problem of the integration work for contextual design has been addressed by the revised version of rapid contextual design (Holtzblatt *et al.,* 2005). For many projects a central criticism of contextual design was that it proved to be labour intensive for many software development projects. Rapid contextual design has attempted to address this by being more practically driven and flexible in its use. It has also addressed ways to provide synchronisation with the growing uptake of the Agile software development methodology. The Agile methodology in this aspect has been proactive in its own evolution against the progression of the software development methodologies.

Contextual design has been criticised for its fieldwork process, mainly for the way it can take a superficial look at work practice and develop new systems based on this. This limitation is placed on field interviews that are restricted to a few hours only. In comparison, ethnography is able to afford much more time. This does not fully allow contextual designers to become fully integrated members of a user group, so the ethnographic dimension of contextual design is a 'ticket' as opposed to a thorough ethnographic engagement (Hartswood *et al.,* 2002).

c) Usage Centered Design

Usage-centered design is a systematic, model-driven approach to improving product usability (Constantine & Lockwood, 1999). The technique was created by Constantine, who has refined it for further variations of usability work such as web design (Constantine & Lockwood, 2002) and Agile development (Constantine, 2002). The usage-centered design technique is made up of three models: a role model, a task model, and a content model. The role model captures the roles that users play to the system; the task model provides the structure to the users' work; and the content model characterises the contents and organisation of the interface (Constantine & Lockwood, 2002). The three core components of usage-centered design are developed in relation to each other and can be extended to integrate with other models such as business rules and data models. The process itself is model driven and is documented in Figure 2.3 where, starting from the left-hand side the system actors and human actors are separated. This is shown in Figure 2.3 as user roles and system actors. Moving right through the process involves task cases and abstract prototyping before finishing with the step of visual and interaction design.

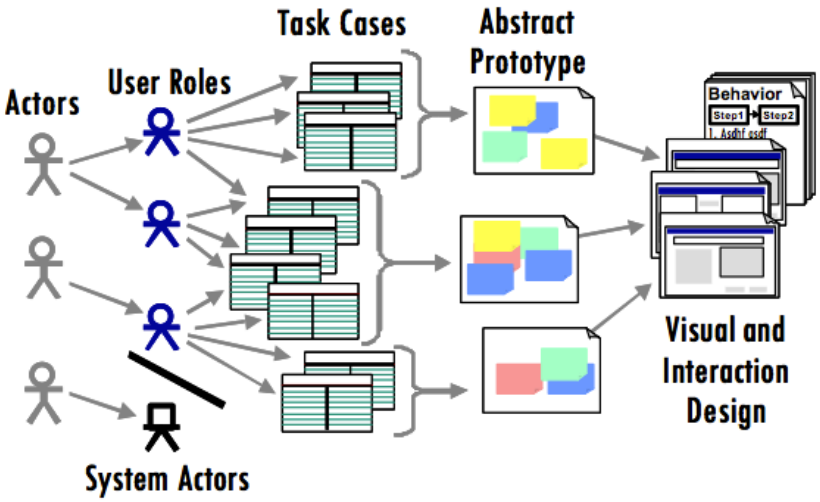

Figure 2.3: Process of Usage-Centered Design
(Constantine & Lockwood, 1999)

Constantine and Windl (2003) highlight how although the core models are connected in a logical sequence, practitioners will develop them in parallel, moving from model to model as information and insight develops. The first phase of the user role model is made up of descriptions of the roles played by direct users. The role of actors in usage-centered design distinguishes system actors as non-humans and user roles as humans. The role model describes the context, the criteria of the context, and characteristics of each role. A map is also used to document the interrelationships among user roles.

The task model details the task cases that model user intentions; the task case is a more detailed perspective of users in roles this is different from the traditional software use case models (Constantine *et al.,* 2003). The advantage of the higher level of detail for a task case over the more traditional design process is that it helps to uncover reusable tasks and it helps to identify relationships between tasks.

Finally, the content model is used to provide a clearer picture of the organisation of the interface and its components. The content model is made up of a collection of abstract prototypes representing the interface and a navigation map representing the connection between them. The abstract prototype is a way to explore the solutions of the contents and organisation of the interface without having to specify all the details. A more comprehensive documentation of the process can be found in the work of Constantine and Lockwood (1999).

Although other research has applied usage-centered design (Patton, 2003; Kubicki & Halin, 2010), Constantine and Lockwood (1999) acknowledge problems associated with introducing UCD into an existing software development process. These constraints force them towards new practices, processes, and tools to be introduced into an organisation. The potential solution suggested for this by Constantine and Lockwood (1999) is the use of training courses for the participants involved in the UCD activities. However, this does come at some cost in time and unless the UCD activities are managed internally by the organisation it can mean a lack of organisational learning for the UCD design methods. Constantine and Lockwood (1999) look at this aspect as the responsibility of organisations for building up an internal body of information for the use of UCD methods. This way the best practices can be adapted to the specific needs of the organisation.

d) The scenario-based engineering process

In scenario-based design, descriptions of how people complete tasks are used as a design representation. They serve to maintain a focus on situations and successive actions from people's activities. It allows learning about the dynamics of the domain and it allows seeing the situation from different perspectives (Carroll, 2000). Scenarios are stories about people and their activities. The other key elements of scenarios are the setting and starting point for the description, the relative positions of the relevant aspects of the given scenario, and the role of the person or people as the actors. Each actor will have his or her own goal and will vary depending on the status in the scenario. A simple example is documented below by Carroll (2000) where he describes the following scenario:

> *"An accountant wishes to open a folder on the system desktop in order to access a memo on budgets. However, the folder is covered up by a budget spreadsheet that the accountant wishes to refer to while reading the memo. The spreadsheet is so large that it nearly fills the display. The accountant pauses for several seconds, resizes the spreadsheet, moves it partially out of the display, opens the folder, opens the memo, resizes and repositions the memo, and continues working."*

In this example, the accountant plays the central actor. The goal in the scenario is to view the memo on the budgets. Potts (1995) identifies the characteristic that scenarios

have in a setting by identifying the person and objects in relation to them. Every scenario will also include at least one actor or actors and a minimum of one goal. In the above example, the further setting described is the situation of the accountant's desktop and the further task required for the account to reach the goal of opening a folder on the system desktop.

The nature of the integration of scenarios between UCD and software engineering is explored in the work by Benner *et al.,* (1993), where they claim scenarios are pervasive techniques throughout the software development process. Underlining their importance for design in any system situated in a complex environment, they present four areas where scenarios can be integrated into the software development process:

- Describing and clarifying the relevant properties of the application domain
- Uncovering system requirements
- Evaluating design alternatives
- Validating designs

More recent work by Sutcliffe (2011) highlights how scenarios share a common link between software engineering and HCI. They both use scenarios for design, although the form and function in each field differs. The use of the term "scenario" has taken on various permutations between the HCI and software engineering literature so a large number of definitions exist (Rolland *et al.,* 1998).

The key difference between the two fields is that in the context of software engineering, the use of scenarios favours a systematic process so more formalised processes are kept to elicit, analyse, specify, and validate requirements. However, in HCI iterative cycles refine the process suitable to users needs. The amount of formal analysis for checking or testing the design has been minimised (Sutcliffe, 2011).

e) Star lifecycle

Hix and Hartson (1993) formed the star lifecycle (see Figure 2.4). The central component is the evaluation. The other five components of the star lifecycle are made up of:

- Conceptual/Formal design

- Rapid prototyping
- Implementation
- System/task/functional/user analysis
- Requirements/usability specifications

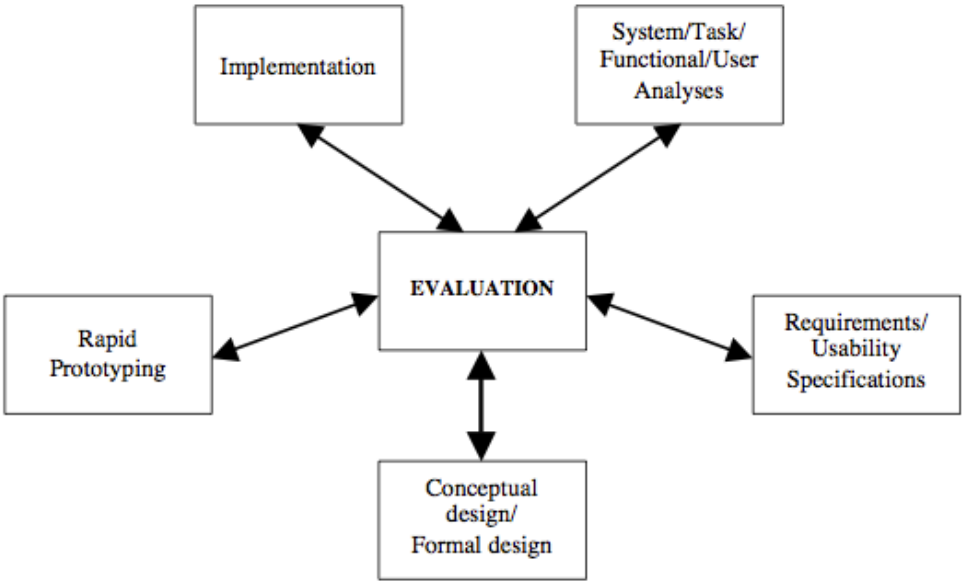

Figure 2.4: The star lifecycle (Hix & Hartson, 1993)

A significant advantage provided by the star lifecycle method is the ability to take a bottom-up approach. This is in stark contrast to the more traditional software development methodology of the Waterfall approach, which is a top-down approach. As the lifecycle allows the design to gradually evolve through each step, any of the five components could be used as the starting point. This allows the design process to become more clearly defined with each step of the evaluation. This iterative aspect of the lifecycle is shown with bidirectional links in Figure 2.4 As an example a full lifecycle could play out in the following way: Requirements/Usability specifications > Evaluation > Conceptual/Formal design > Evaluation > Rapid prototyping > Evaluation > Implementation > Evaluation > User analysis > Evaluation.

The role of integration between the star lifecycle and the software development process is discussed by Metzker and Offergeld (2001), who highlight how the star lifecycle specifically separates the development of the user interface from the development of the software system. The only connection between the two is via the systems analysis and testing/evaluation, where these authors acknowledge that the star lifecycle has a

weakness is in terms of complex process surrounding the communication of the lifecycle. This aspect of the work requires further research work to how this can be further supported. The work by Metzker and Offergeld (2001) also underlines that the star lifecycle addresses only the interactive parts of a software system, leaving questions on how to best integrate it with general software development methodology, which by default encompasses the full software development cycle.

f) Discount usability engineering

Discount usability engineering is a method developed and promoted by Jakob Nielsen (Nielsen, 1990; Nielsen, 1993; Nielsen, 2003). The technique grew from the recognition that people rarely made use of the recommended usability engineering methods (Nielsen, 1993; Whiteside *et al.,* 1988). Nielsen recognised that in practice usability engineering was not used because of its cost in terms of time and in terms of money for employing a UCD expert. The discount usability engineering method was consequently formed around the following three simple techniques:

- Scenarios
- Simplified thinking aloud
- Heuristic evaluation

The techniques are based on they are not complex to use, so they should have an increased chance of adoption. These three simplified techniques can be briefly described as follows:

- Scenarios are a variation of prototyping that allow a designer to reduce the complexity of the whole system. The types of prototyping that can be carried are either horizontal or vertical. Horizontal prototyping allows for a broad view of the system with limited functionality, while vertical prototyping allows for the full function of the system to be demonstrated.

- Traditionally, trained experts would carry out the studies by videotaping the subjects and conducting a detailed analysis. The simplified thinking aloud technique attempts to replicate this by using users, providing them with test tasks and asking them to talk out loud as they carry out the task.

- Finally, given the length of interface standards and collections of usability guidelines being formed at the time, Nielsen proposed a heuristic evaluation with ten core principles. These “rule of thumb” guidelines were again made to be more accessible for developers. The uptake and success of the heuristic evaluation, Nielsen has then revised the heuristics in the work usability inspection methods (Nielsen, 1994b). The use of discount usability engineering has also further evolved and been adapted for the software development process of Agile development (Kane, 2003a).

The role discount usability engineering plays in the integration between UCD and software engineering is in lowering the actual accessibility of UCD methods. UCD-led techniques have become more practical for use by software developers and non-UCD experts. However, discount usability has come under particular criticism because of this but also because of its wide adoption. Wixon (2011) identifies that the technique has become widely used, going from a few usability labs in the 1980s to hundreds of labs and thousand and thousands of tests conducted each year. This has been due to the recognised commercial and monetary implications in having usable software. Nevertheless, Wixon (2011) states that a clear low-cost process to work from was a key factor in such a wide adoption. The significant advantage and disadvantage of the method is revealed in its name, a ‘discount’ method. The integration of discount usability engineering and more modern software-development methodologies such as the Agile methodology can mean a lack of early influence at early stage of the design process. In this sense, Wixon (2011) cites the lack of championing measurable goals. Therefore the role of usability engineering is limited to the scope of usability testing at the very end of a process where the role of UCD is constrained much more to incremental improvements.

2.4 Where user centred design and software engineering meet

The problem of integration between the two fields of UCD and software engineering is widely established from both sides of the community. For example, the IFIP working group 2.7/13.4 on User Interface Engineering, which was established in 1974, has intentionally investigated the migration between software engineering and UCD (ifip 2011). The significance of the problem has been recognised more recently in various

ways: multiple workshops such as ICSE 2003 (Kazman *et al.,* 2003), INTERACT 2003 (Harning & Vanderdonckt, 2003), and a Computer-Human Interaction (CHI) workshop 2004 (John *et al.,* 2004); special sessions at conferences such as HCI International 2003 (Görasson *et al.,* 2003); and special issues of journals such as *Software Process: Improvement and Practice* (Kazman & Bass, 2003). The existing literature has asked different questions on how best to integrate the two different fields. Seffah (2004) highlights how during the past fifteen years the UCD community has formed various methods for the integration of UCD and software engineering. The purpose of the methods has been to promote usability, accessibility, and acceptability for interactive systems. However, the methods are still underused and difficult to apply for software development teams. Questions remain about how to efficiently and smoothly integrate UCD methods into established software development processes (Mayhew, 1999).

Figure 2.5 presents a summary of the core conflicts and discrepancies of UCD against software engineering and subsequent areas, which in turn highlights the gap between the two fields.

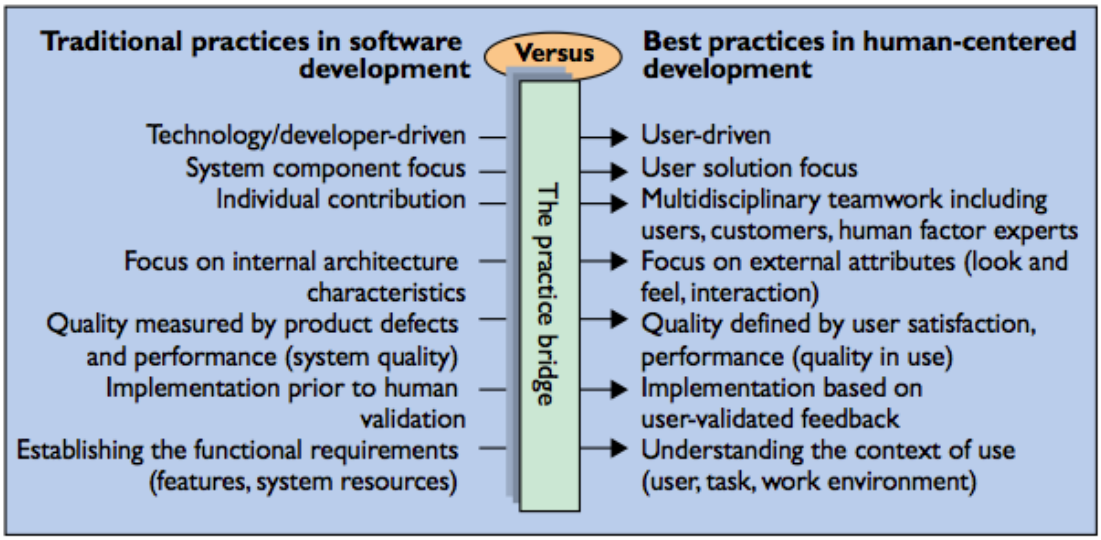

Figure 2.5: Practice bridge (IBM 2004 cited in Seffah and Metzker 2004)

There has been acknowledgement that the formation of a generalised template solution is likely to be challenging because each integration case is unique in its own right (Battle, 2005; Radle, 2001). The work by Battle (2005) addresses this challenge by the introduction of patterns of integration and subsequently divides the generic lifecycle into three phases of early, middle, and late. In any of these phases different situations and activities are applicable. For a given pattern, best practices are suggested with the

outcomes used through the development cycle. The four example patterns described by Battle (2005) cover 1) foot in the door for a internal usability group, 2) foot in the door for external consultants, 3) UCD focus on early definition and design, and 4) UCD in every phase. The range of different patterns has distinctive deliverables based on their area of focus. These generic phases are early, middle or late. Battle (2005) describes how the early phase deliverables are the aspects to target for foot in the door for internal usability group. The range of deliverables is a low-fidelity prototype, a requirement document, or a specification document. In contrast, Battle (2005) highlights how the pattern for a foot in the door for external consultants targets the latter phase of the UCD cycle so for UCD to make an impact the deliverables change to making recommendations for improvement and sometimes facilitating a usability test. The contrast between the two foot-in-the-door groups of internal and external usability means that the internal usability group can be in a better position to focus on adding value to an early design deliverable. Significantly Battle (2005) also offers best practices for UCD practitioners based on which pattern they use, this includes stop "telling people that their baby is ugly". This is a practice concerned with preventing UCD specialists of giving bad news before a release. Another practice covers how working collaboratively with multidisciplinary teams can help highlight the importance of meeting with developers so as to have insight into the technical constraints and opportunities. The work by Radle and Young (2001) has presented a study across three different organisations. They have formed key lessons for addressing usability for organisations: first, forming excellent interpersonal skills are crucial to forming relationships with development teams; secondly, understanding that most resistance to UCD comes from other pressures (such as schedules) and a lack of information; finally, observing user interactions should be done first hand. Work by Billingsley (1995) has also identified five factors for launching a corporate usability programme and maintaining its success: (1) preliminary strategic planning; (2) a high-level champion; (3) an experienced usability professional leading the corporate effort; (4) a careful selection and sequencing of initial usability activities; and (5) support and incentive for developer involvement. Work by Schaffer (2004) has also outlined the integration of usability from a corporate perspective. He suggests that the process for designing interfaces has been driven by technology. He proposes that the reverse happens, he argues that user design should occur first and the technical solutions fit the interface. This work formed the Schaffer Methodology, which is based on 10 steps to

institutionalise usability assurance. Schaffer (2004) also acknowledges and identifies the fundamental shift required for organisations to adapt usability. His explanation of this is one as deep changes because it involves changing the thinking and values of people in the organisation. Based on this, Schaffer (2004) calls for a deep philosophical change that must take place in the move towards UCD. This requires a change in the approach of the design and development process. The issues raised from the proposals of key lessons and templates are important for UCD although they have been formed within a different context. They do help to raise the questions to how the relationships and communication within each project are developed, the role of the usability activities, and support and incentive for developer involvement. These question all underline the types of challenges of working within a multi-disciplinary team that this research is part of.

Göransson *et al.,* (2003) argue that the integration of usability into software development requires a process perspective and that the usability roles must be deeply involved throughout the entire development process. They highlight particular weaknesses of usability engineering, particularly how it is made up of a large range of techniques from analysing users and specifying usability goals to evaluating designs, although it does not address the whole development process. The techniques are still rarely defined with the UCD practitioner being accountable for the usability activity. Göransson *et al.,* (2003) perceive such techniques to be problematic as they are bolted on to the software-development process as singular activity. They do not see that there is anything fundamentally wrong with usability engineering techniques. However, they argue that the integration of UCD with software development must be a natural process. Thus, according to Göransson *et al.,* (2003) integrating UCD into the process and allowing for a stronger focus on the design phase within a framework of UCD software development is an essential step that still needs to be accomplished.

Faulkner and Culwin (2000) also echo this call in their own analogy to the integration of UCD in the software development process. Faulkner and Culwin (2000) state "*if in the past we have made cakes with cherries on, we now need to change our approach and make cherry cake*". They stress that it is only by employing UCD throughout the whole process that we can then ensure that user needs are addressed from the very start. Siegel and Dray (2003) further underline this point, stating that the integration and responsibility for UCD roles need a shift in focus from a sole focus on doing studies and

operating on the peripheral to generating designs and products and directly affecting the process.

A significant contribution and a collection of integration factors has been made in the books by Seffah *et al.,* (2005) and Seffah *et al.,* (2009). Seffah *et al.,* (2005a) present a collection of work for integrating HCI and usability engineering techniques into the established software engineering lifecycles. This aims to understand and question the principles, myths and challenges behind the integration between UCD and software engineering. Several specific chapters from these two books are explored below.

The work by Seffah *et al.,* (2005b) analyses the range of relevant frameworks to understand how far they go towards the role of integrating UCD-led methods with software engineering. The work questions how the software engineering lifecycle can be re-designed so that end-users and usability engineers can participate actively. They highlight that this remains an open question as it is problematic to fully understand the adaptation of such frameworks and how much influence they have on software development and management processes. Critically, they also look at the role of artefacts in the integration between UCD and software engineering. They qualify artefacts as the elements that characterise a software engineering methodology. They question how usability techniques and activities will be gathered and they specify relevant usability artefacts as well as how they will be presented for the software development process. The techniques of patterns and use cases are highlighted to be beneficial for this level of cross-pollination between UCD and software engineering.

Pyla *et al.,* (2005) question the association of the role of UCD, to function under Agile processes. They have formed a methodology called Ripple, which is a database, which supports a shared design representation framework. Through the use of the system Ripple can identify the connections and dependencies within each lifecycle. The framework provides artefacts generated at each stage between the two development lifecycles, which are then filtered and messaged into the appropriate UCD or software development lifecycle. Critically, Ripple does not merge the UCD and software engineering development processes into a single lifecycle. The alternative approach taken is to co-ordinate each lifecycle separately using a shared representation. This way, the Ripple system is acting as a translation tool for communication and co-ordination between the two processes.

Further work by Seffah *et al.,* (2009) has since highlighted the converse side of these issues and the lack of well-established software engineering lifecycles that are missing the UCD quality required for the development of highly interactive systems. This is where a development process may be dominated by UCD concerns and there is a failure to integrate software engineering methods into it. These subsequent issues question and examine software engineering models, methods, and tools for challenging UCD issues such as adaptability, universal usability, and accessibility; re-engineering models and techniques (formal specifications methods and notations; software for supporting the UI development lifecycle: requirements, analysis, design, implementation, evaluation and traceability, and maintenance).

Hvannberg (2009) and Tarpin-Bernard *et al.,* (2009) provide such examples of research. Hvannberg (2009) created a finer model of evaluation that may be used alongside the design process. The goal of the work was then to question and learn how an evaluation model can help further support the understanding between cause and effect in user interface development. The purpose was to improve the understanding of the interaction between design and evaluation.

The approach by Hvannberg (2009) involves specifying different work products and asking questions about the implications of work to design, and the subsequent cause and effect from this. The resulting evaluation model is then informed to describe the design decisions and consequences the work models have on the design. The model also records the implications for the association between the design and problem domain. The work by Tarpin-Bernard *et al.,* (2009) outlines an architectural model-based approach for adapting interactive applications to various contexts. Their architectural model is called AMF and looks to utilise task, concept, platform, and user models as well as an interaction model. Their own work demonstrates the use of Unified Modelling Language (UML) models and a UCD approach of task analysis. In taking such an approach they are able to designate a software process that builds the application and safeguards the correct behaviour.

2.5 UCD and software engineering in a commercial context

The problem of integration between UCD and software engineering is also widely recognised in the commercial context. The work by Jerome and Kazman (2005), Boivie *et al.,* (2006), Verdenburg *et al.,* (2002), Gulliksen *et al.,* (2004), and Bygstad *et al.,* (2008) have all identified the difficulties of such integration. Again these studies identify common challenges from the separated environments between UCD and software engineering, pointing out a lack of guidance for using UCD and a general lack of awareness of UCD. The survey work by Jerome and Kazman (2005) identifies that despite the growing development towards the migration of software engineering and UCD activities, most industry professionals are still to follow the proposals from the academic and industrial research communities. They also recognise that in part many software engineers and UCD practitioners continue to work separately. The required crossover between the two fields disciplines happens infrequently and not early enough in terms of the software development process. The survey also underlines some misperceptions and miscommunications between the two fields.

Other exemplary work has been carried out by Rideout *et al.,* (1989) and IBM (2012). The work by Rideout *et al.,* (1989) highlights the work carried out at Hewlett-Packard. They cite how their own integration between UCD and software engineering has come from working within interdisciplinary software development teams. This meant they were able to take a system orientation as being a significant reason to allow for the migration between UCD and software engineering. The implications from working in an organisation the size of Hewlett-Packard can mean working across several teams. The usability engineers in this instance can significantly improve their effectiveness by transferring knowledge and skills to make others more effective. Critically, they also identify the role of good teamwork as a key component to build the bridge between user information and software design. More recently IBM has proposed strategies for adopting UCD practices within an organisation as well as strategies for persuading an organisation for UCD adoption, and staying committed to UCD. The aspect of integration between UCD and software engineering is significant to my own research as I am working between the Usable Image and OMERO projects. So questioning the approach of how the crossover is managed between the projects for my research is critical for preventing the separation of isolated working between the Usable Image and OMERO projects.

The work by Seffah *et al.,* (2005a) indicates that usability integration is limited to large organisations. However, the work by Fellenz (1997) and Aikio (2007) do investigate the role of UCD in smaller organisations. Both have explored how they have employed a UCD department for conducting usability activities. Further work by Bloomer *et al.,* (1997) and Venturi *et al.,* (2006) has also explored the tactics for adaptation in a commercial context. The work by Bloomer *et al.,* (1997) states that UCD could be integrated into an organisation by the development of a UCD strategy to support the organisational objectives. The findings by Venturi *et al.,* (2006) underline the organisational factors that play an important role in adopting UCD. They recommend that UCD should be part of the organisational strategy and should be endorsed by management. Further still, when designing a bestspoke system the system should be clearly defined with the user. The subsequent outcomes of the practice of UCD should then be communicated internally and externally to the organisation.

The challenge of examining the role of UCD in smaller organisations is pertinent issue in my own research as I am working with a small software development team. So the extent to how the organisational factors of the SSD context play a role in adopting UCD to how UCD is communicated internally and externally for the projects is meaningful for my own research questions.

2.6 Established integration challenges

The following seven points cited in Seffah and Metzker's (2008) research synthesize the well-established challenges in the UCD software engineering integration literature. The issues cover and recognise the central obstacles to the integration of software engineering and UCD. Where necessary I have explained these aspects with related work.

1) Separation between user interface and system functionality

The first issue can be traced back to the very different perspectives that each particular field adopts. UCD makes it drastically different from the way system design is approached in software engineering. Software applications have been developed so that the interface is separated from the underlying software system. This is defined by Seffah and Metzker (2008) as the thin layer that operates on top of the software in the

term “user interface”. The work by Mayhew (1999) also highlights another common misunderstanding, which is that usability engineering tasks do not arise until the more detailed phase of a development and the belief they can be done right first time. The limitation of applying only software engineering methodologies is further reinforced in the work by Hix and Hartson (1993), in which they recognise that user interface development is made up of both software development and UCD. If software engineering overlooks the fact that user interface development is made up of both software engineering process and UCD, it will continue to fall short. This can leave both processes of software engineering and UCD to evolve separately. This shortcoming has not only been acknowledged by the UCD community work. Bass *et al.,* (2001) has recognised this issue from a software architecture perspective and how usability scenarios affect the design of the software architecture.

It can be argued that this integration gap is compounded by how the two fields of software engineering and UCD have developed into two separate bodies of knowledge, making it difficult for the individual software engineer to acquire extensive knowledge within usability and for the usability professional to become a skilled software engineer (Göransson, *et al.,* 2003). What is deducted from this that it seems more difficult for software engineers and UCD practitioners to prevent their own misconceptions of each other’s process. This point is relevant to the discussion of the field being educated, which is discussed further in point 7.

2) Cultural gap between UCD roles and engineers

There is a significant gap between UCD and software engineering roles because of the different culture. The misconception between the two disciplines covers different terminology and a technology centric view in software engineering to the contrast of a user focused and way of working in UCD (See Figure 2.5).

Seffah and Metzker (2008) describe that software engineers must interpret UCD for their own cultural context. Also, UCD specialists must understand how and why the technical choices influence the end-design. Work by Seffah and Metzker (2004) has also proposed key skills for software engineers to develop better interfaces. These key skills cover an appreciation of the UCD lifecycle and the benefit of usability to the quality of the software. They point out that software engineers should be aware of the design decisions for dialogue types and input/output devices; that they should value the

benefit of using prototypes to evaluate the system; and finally that they should be aware of their own limitations in their knowledge and not be afraid to ask UCD specialists for advice.

3) UCD has to be adopted throughout the organisation

The significant role that the organisation plays in UCD integration is reported by Seffah *et al*., (2005). They highlight how both the work in the Usability Engineering Lifecycle (Mayhew, 1999) and usage-centered design (Constantine & Lockwood, 1999) are affected by organisational barriers. Despite this there is still a general lack of empirical studies on the effects and acceptance of UCD in organisations (Glass, 1995; Basili *et al.,* 1999).

Such UCD organisational surveys are viewed to be a key requirement to help support organisational learning. However, the work is difficult to conduct (Seffah & Metzker, 2008) as it would be required to compare the use of a UCD method in a project against a different project that is not using a UCD method. In addition, the ability to control factors such as skill, motivation, and the software engineering approach would also have to be managed. This underlines the potential variables, time, and costs that would be involved in carrying out such survey work. Mayhew (1999) says that the role of understanding and what motivates organisations and causes them to change are key factors. Mayhew (1999) argues that whether your role in the organisation is as a usability practitioner or not, critically you have to recognise yourself as a "change agent". The existence of change agents has also been discussed in other research, each of which has its own slight variation for their given context as the following demonstrate:

> "*A high-level champion for usability can help to establish organizational commitment, provide resources, and create opportunities for process change*."
>
> (Battle, 2005)

> "*Throughout its evolution, the software usability initiative has been championed by Lynden Tennison, Union Pacific's Assistant Vice President of Information Technology, with the full support of Joyce Wrenn, The CIO*".
>
> (Billingsley, 1995)

Schaffer's (2004) interpretation of a change agent is embodied by an Executive Champion. The Executive Champion is a leader who has the power to provide direction, support, and political management for corporate change. This representation is

significant as it represents a management role. The implementation of UCD has been identified as one of the more far-reaching challenges of integrating UCD throughout an organisation. Significantly, the implementation of UCD must come from the very top and be based on a clear vision and set of goals. For the individuals within a project team, the critical factor lies in the integration between the UCD and software.

4) Usability of UCD methods

The UCD and software development communities have questioned the usability of UCD methods. Gunther *et al.,* (2001) and Vredenburg *et al.,* (2002) have both raised concerns regarding the complexity and usability of UCD methods. Such concerns are with the benefit of identifying the perceived key advantages and weaknesses of UCD methods could be useful for the adoption and promotion by UCD practitioners of any given method. Another factor is the cost-benefit tradeoffs that play a major role in the adoption of UCD methods. This can mean that heuristic evaluations are more heavily used because they are relatively easy and less costly.

Work by Rosenbaum *et al*., (2002) has demonstrated the factors that UCD methods need to accommodate in order to be established in the software development process. He first identifies the core principles of UCD need to be clearly communicated to the entire team. The requirements of the UCD methods subsequently have to be seen to be beneficial. Rosenbaum *et al*., (2002) further discusses how usability testing will typically have to be adapted to the context to be more useful. The work by Vredenburg and Butler (1996) and Mao and Vredenburg (2001) underline this problem in the context of industry. In this there is a general lack of adoption of UCD techniques and consequently an absence of UCD methods that are effective in their use. This even covers more basic practices of UCD such as iterative design and prototyping failing to be used. Further calls have been made in the work by Thimbley (2000) for the development of more theoretical methods that are more readily accessible. The research by Gulliksen *et al*., (1999) has called for the development of a UCD framework or principles on how to perform UCD in practice. The lack of adoption of UCD techniques was the principle reason for the creation of the method of discount usability as previously discussed in section -. Based on this the challenge, the usability of UCD methods and how they must accommodate in order to be integrated into the software development process openly calls for new methods and frameworks of integration.

5) Lack of support tools

Much literature (Nielsen, 1993; Mayhew, 1999; Rosenbaum *et al*., 1999; Reiterer, 2000) reinforces the underlying problem of inadequate support for UCD processes. The requirements for process support tools should allow project teams to deploy UCD methods efficiently through an organisation without having a set of techniques forced on them. This is especially significant for the varied nature of software development methodologies. Any UCD tools presented should allow for the integration and flexibility into a given organisational software development process. Work by Metzker and Offergeld (2001) underlines the importance of any UCD software tools introduced being aware of existing UCD practices yet also allowing for the organisation to evolve and tailor their approach.

Existing work that has documented such best practices has been addressed by Rosenbaum *et al.,* (2000), Metzker and Offergeld (2001) and Spencer (2000). Rosenbaum *et al.,* (2000) conducted a survey on UCD practitioners at multiple CHI conferences. The findings demonstrate the diversity of problems ranging from resistance to UCD and lack of understanding of UCD to a lack of ability to communicate the cost-benefit and impact of usability results. The authors state that UCD professionals must look at ways to combat them. A recommendation they propose is to aid and develop business cases so that they may 'learn to speak the business language' of their working associates. Spencer (2000) has developed a streamlined cognitive walkthrough method specialised for use in a large software development company. However, this type of research is limited in software engineering (Basili *et al.,* 2004). This is a significant drawback for the UCD and software engineering community because if tailored methods are kept internal or retained to a given individual in an organisation, then the information can be easily lost if not published by companies or the scientific community (Metzker & Offergeld, 2001).

Metzker and Offergeld (2001) have also recognised such shortcomings and proposed the experience-based Human Centred Design (HCD) lifecycle. The lifecycle includes process models, tools, and organisational measures. The development of their prototype HCD tool, the MMI (Man-Machine Interaction) hyperbase, has been formed to support software developers. The tool provides this support by allow a user to perform various HCD activities and to organise the outputs of iterative design processes.

6) A collection of best practices is missing

Studies have shown that even highly interactive systems are developed without guidance or involvement from UCD specialists (Metzker & Offergeld, 2001). For this reason, Mayhew (1999) highlights that little knowledge is available within development teams. Seffah and Metzker (2008) discusses that development organisations who are inexperienced with usability can be overwhelmed by the process of UCD models and that they also lack an commonly understanding of basic project constraints (context of the domain, team size, and experience level of the team). So although many UCD methods exist, there remains a lack of available empirical evidence to guide meaningful adaption for software engineering (Seffah & Metzker, 2008). This critical question of how systematic tailoring of methodologies can be achieved has not yet been fully addressed for UCD. As previously outlined by Mayhew (1999), this will be left out when schedules are tight, until UCD is widely recognised as a critical component to software development.

7) Education gap

There are recognised communication difficulties between software engineering and UCD derives from an educational gap (Faulkner & Culwin, 2000; Pyla *et al.,* 2004; Chan *et al.,* 2003; Phillips & Kemp, 1996). The particular work by Faulkner & Culwin (2000) capture the cause of this problem, highlighting how UCD can traditionally be taught in a master's degree course and are delivered to people with diverse backgrounds. These students can have minimal or no software engineering training, so they may have little experience when they work alongside a commercial software developer once they graduate. However, at the opposite end, software developers leaving educational institutions have only a very basic introduction to UCD. This can and does leave the education gap for each field in a perpetual cycle. UCD students can be limited to how they would actually design systems in a real environment. Software engineers can be too busy in building the software and consequently forget that there will be real people who will have to use the software. Seffah and Metzker (2008) point out the two central areas of improvement for UCD and software engineering students. For the education of UCD students, it is fundamental that their technical awareness is increased. When educating software engineering students, it is critical that their awareness of the UCD process and in relation to software engineering is improved. This is essential for the mediation between the fields and consequent closer integration between the practitioners.

What these seven points conclude are a wide collection of the established challenges in the UCD and software engineering, from separate working practices between UCD and software engineering to critical gaps in the education practices of the two fields. There cannot be a single solution to all of these problems. The following section now goes on to discuss the specific context of the scientific software environment and its role to understand the integration gap with UCD.

2.7 SSD Environment

The work discussed and highlighted in the previous section acknowledges the established gap between UCD and software engineering, underlining that this is not by any means a new challenge specific to SSD. However, as covered in the work by Seffah *et al*., (2005), that is central to understanding the UCD software integration gap is an understanding of the organisation and context (Point 3 in section 2.3). Because of this, before I specifically examine the UCD SSD gap I will present the organisational and contextual factors of SSD, to ensure that the reader is sufficiently aware of the existing research and familiar with the organisational factors of SSD. Throughout this section I will question the roles the people, development process, funding, and community have regarding the impact SSD.

2.7.1 Scientific software developers

The following types of categories have been identified for academic scientific software developers from the SSD literature to structure the review against the characteristics of scientific software developers. Table 2.1 captures the various categories that describe the role of the software developer in the SSD projects.

The important categories outlined in Table 2.1 highlight the levels of problems for SSD work from the literature. From the perspective of this research, the categories are significant for understanding the UCD SSD gap. This is because of the different levels of experience and problems identified at each of the three levels that a scientific software developer might occupy. So any proposal for UCD SSD integration is open to

these different challenges depending on the type of scientific software developers targeted.

Table 2.1: Categories of scientific software developers

Category	Description	Example Literature Reference
Isolated scientists turned software developer	A scientist with no prior software engineering training.	Kane *et al.,* (2006); Kelly and Smith (2009)
Post graduate student developer	A scientist who has both an extensive knowledge in their technical domain and a higher comfort level in writing code.	Pitt-Francis *et al*., (2008), Segal (2004).
Specialised software development team	A software developer role with prior software engineering training and software engineering expertise.	Killcoyne & Boyle (2009); Basili *et al.,* (2008); Ackroyd *et al*., (2008)

The first category of "Isolated scientist turned software developer" describes the scientist who develops software for his or her scientific domain. In this category, the software developer will have little or no prior software engineering experience or training. They will be self-taught to program code so can be unaware of software engineering practices that would support them in their work.

The research by Sanders and Kelly (2008) explores this. In their survey, they interviewed 16 scientific software developers from a range of domains, and they found that 75% had not undergone any formal training in software engineering. In a separate survey by Hannay *et al.,* (2009) 58% of scientists reported that they did development on their own, 17% worked with one other person, and 18% worked in teams of three to five people, while only 9% worked in larger groups. Their work also identifies how the ability to develop scientific software is learnt via peers and through self-study.

However, this category can mean that the nature of the problems and mistakes that occur is critical. Kelly and Smith (2009) highlight an instance where a molecular biologist had to retract three papers from prestigious journals. This was because the software turned over a column of figures and so produced a mirror image of the actual protein molecule that was being designed. They also cite the work of Kelly and Smith

(2009 cite Hatton, 1997), where scientific software was studied from nine different seismic analysis software packages using the same algorithms. The output was different from each version. This variation is often affected by one-off errors but is again a major concern for scientific results (Kelly & Smith, 2009). The recognition of this shortcoming for scientist turned software developer has led to work by Wilson (2006) and his software carpentry project. Wilson (2006) has recognised the lack of software engineering tools that are used by scientists and advocated the requirement to educate scientists of software engineering practice for their benefit.

The second category of 'Post graduate student developers' is where the development process can be most affected by the turnaround of the academic workforce. This causes critical reliance on good software engineering practices for the shelf life of the software. Such a scenario is outlined in the work by Pitt-Francis *et al*., (2008) which gives the implications for the software development. They explain how in larger and longer-running projects that can require post graduate and post-doctoral researchers over periods of several years. The maintenance of the code can be neglected by either poor development practice or minimal documentation. This can lead to future problems with the development of further functionality or with the software development infrastructure. It is also in this category that the existing term of 'professional end-user developer' is most appropriate. The extensive research work by Segal (2001; 2005; 2007) defines this term as "people who do not consider themselves primarily as scientific software developers, but work in complex and technical, knowledge-rich fields and have the requirement to develop software in order to advance their own professional goals". Segal (2004) has identified scientists as "professional" end-user developers, pointing out that they have extensive knowledge in a technical domain and a higher comfort level in writing code. Typically, there is a greater chance of this occurring at a higher academic level, as a scientist would have spent a longer time learning how to code.

The final category specified in Table 2.1 is a specialised software development team, but in this category more established and well-practiced software engineering methods and practices are used. Baxter *et al.,* (2006), who describe themselves as veterans of large scientific software development projects, propose the following set of best practices. These points are a guide to how the scientific software development teams should operate for the benefit of the software development process.

- Design the project up front
- Document programs and key processes
- Use a quality control process
- Apply data standards where possible
- Incorporate project management

This list, as explained by Baxter *et al.,* (2006), is in response to maintaining the software development practices as would be done accordingly in a scientific experiment. Killcoyne and Boyle (2009) have formed a specific research informatics team to meet the challenges of software development within the life sciences. The ten-man team is involved with many projects across the institute as shown in Figure 2.6. The figure shows how the informatics team covers many different laboratories and projects, and how they have adapted a process to support:

- Good communication across the team
- Rapid development and delivery
- Project management to coordinate development and manage dependencies

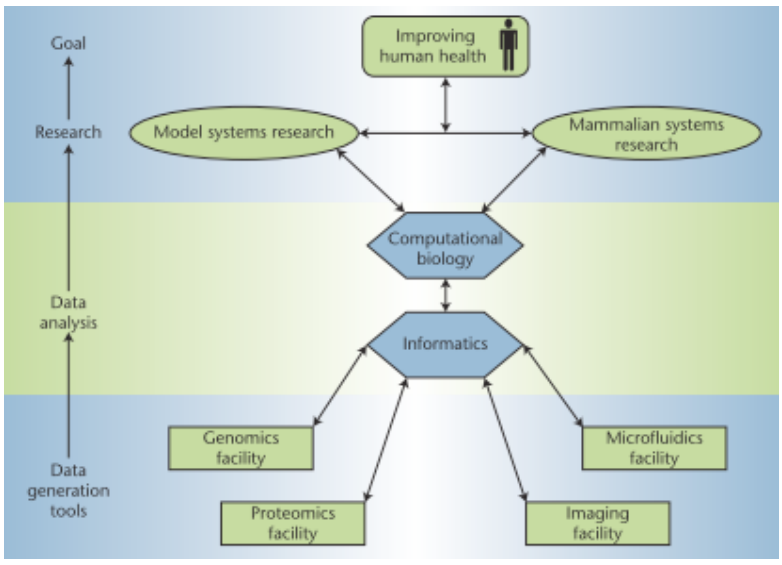

Figure 2.6: The Institute for Systems Biology informatics context (Killcoyne & Boyle 2009)

These steps were met by the adoption and refinement of the Agile software development methodology by the team, with the following key Agile practices to maintain communication within the team such as daily stand-up meetings, iteration planning meetings, and pair programming. The work by Killcoyne and Boyle (2009)

also underlines the continual communication of the ideas and work communication process required for working within the scientific domain. This contrasts with other development domains where this is not typical until a management role is occupied. The SSD process requires every developer to be able to suggest, present, and write about the software to fellow developers and scientists. This presents very different problems in contrast to the first category of a professional end-user developer.

The range of scientific roles that have been highlighted in Table 2.1 also re-enforces the role of science and how it should not be perceived as a single type of activity (Kling & McKim, 2000). They also say that even in similar fields scientists can have very different research styles, communication patterns and information needs. The implication can be for a scientific software developer to unknowingly move from the developer-centred "I-methodology" (Akrich, 1995; Oudshoorn & Pinch, 2003) as in the instance of isolated scientists becoming software developers, to a 'We-methodology' (Fry & Thelwall, 2003), which more closely describes the role of a scientific software developer in a specialised SSD team. How UCD must subsequently integrate and evolve through such a process in SSD is a pertinent question for the success of the software.

The software development environment in SSD also has an impact on the context. Work by Sanders and Kelly (2008) recognises that long development-cycles can be common in scientific software projects. The software in this instance is the result of a combined effort performed by several scientists over several years. Carver *et al.,* (2007) outlines that this is due to the software development characteristic where modules are individually added to an application. The work by Sanders (2008) reviews projects that have taken an iterative approach rather than a plan-oriented approach to software development. The variety of Agile based SSD in the literature is also further evidence of this (Easterbrook & Johns, 2009; Crabtree *et al.,* 2009; Kane *et al.,* 2006; Mugridge, 2003; Pitt-Francis *et al.,* 2008; Segal, 2005; Wood & Kleb, 2003; Kane, 2003b; Blom, 2010).

In terms of questioning the benefit of Agile practices, the work by Sletholt *et al.,* (2011) shows that the Agile approach can be valuable to scientific software development, especially for smaller-sized teams and projects. However, further work by Sletholt *et al.,* (2012) examines that scientific software development projects can embrace the Agile methodology in their focus on flexibility and communication, but otherwise are

selective in using the Agile methodology according to the book. The work by Segal and Morris (2011) also recognises that from their own experience developers mirror a typical scientific end-user development model, which is an iterative feedback approach. So what they say and what they do frequently does not match. However, they do recognise such exceptions to this, such as in the work by Ackroyd *et al.,* (2008) and Pitt-Francis *et al., (*2008), who both tailor the Agile method of extreme programming practices to their context of use.

2.7.2 Funding and exploratory scientific research

The academic environment, directed by the objectives set out by the research funding bodies, guides the software outcomes. A consequence of working within the academic domain of science is that the "Development of any long-term view is difficult to justify and even harder to realize" (Killcoyne & Boyle 2009). Further to this, work by Carver *et al.,* (2007) highlights the significant difference between traditional commercial IT software, and a academic project where funding for a academic project is often from a governmental source, and the users and wider community of the software may or may not be part of that same funding source. The success for an academic project team can mean meeting the desires/targets of the funding bodies and the user community. This measure of success can be problematic if a user community if the actual community may not be representative of the true community, as discussed by Segal (2009). She explains that a community may only come together for the purpose of funding, so the community has no previous collaboration and holds a false perception that the outcomes will be helpful for all involved.

Pitt-Francis *et al.,* (2008) also investigates how, in the academic context, the software development process can suffer from the high turnover of staff. The renewal rate of research staff is high because of short-term funding contracts. Pitt-Francis *et al.,* (2008) therefore, emphasised the necessity of good development practices and a suitable level of documentation in SSD projects.

Along with the academic funding and context of SSD comes the pressure of writing and publishing papers. However, as explained by Nuin (2008) where there is published work it has come under some criticism because many SSD applications are developed

towards a publication, so there can be little documentation, errors and bugs, difficult to use, and not portable. So frequently the publication of the SSD project is as a by-product of the central publication of the scientific project. Killcoyne and Boyle (2009) also recognise the importance of this point, noting that in addition to any software development process used within a scientific environment the SSD process must be aware that any success is measured by the metrics of scientific advancement. There are implications that any process within the academic context requires various ways to present information through a range of academic channels. This covers such things as scientific seminars and conferences and writing academic papers for science journals/conferences, and it extends to writing funding grants to obtain sponsorship and ensure the SSD project team sustainability.

An additional consequence of operating within the scientific academic domain for SSD is that the nature of the academic research based work is exploratory. The activity of science uses a trial and error approach to discover and eliminate ideas among a spectrum of thoughts to explore (Kane *et al.,* 2006). The work by Segal and Morris (2008) has studied this aspect of working with scientists to be particularly challenging for scientific software developers. Scientific software developers are following traditional software engineering practices and want the software requirements up-front; however, to scientists this is a very difficult and unfamiliar practice. The reason for this, as described by Segal (2008), is that traditional working practice for scientists allows the requirements to emerge. So the scientists expect to work to their own requirements for software in a similar way.

2.7.3 Community dependencies

The role of the scientific community can also play a large part in the success of the software. The work by Hine (2006) thus discusses the effect of the deployment of the scientific software. This deployment and subsequent success may rest upon a key individual scientist, a laboratory or an institution. This is referred to as a tragedy of the commons by Segal (2009) as these issues impact on the requirements both in agreements and priorities. In addition to this, the user community in turn will not have a single voice so this sets out further complications for a SSD project i.e. a laboratory may want features A, B, and C yet a scientist in another laboratory may request features

X, Y, and Z and make the most noise about directing the software project towards their own scientific agenda, and so the software is not entirely representative of the wider scientific community.

In addition to dealing with the funding and the context of the scientific community, SSD also has to work within a difficult and continually evolving environment with the underlying cultural obstacle of computational science (Wilson, 2006). This is exemplified in Microsoft Research's report Towards 2020 Science where they quote:

> *"Software engineering for science has to address three fundamental issues: (i) dealing with datasets that are large in size, number, and variations; (ii) constructing new algorithms to perform novel analyses and syntheses; and (iii) sharing of assets across wide and diverse communities."*
>
> (Emmott *et al.,* 2006)

Wilson (2006) highlights how "getting the right answer" does not make the list as it is not part of the computational science culture. This is a notable problem for any scientific software developer working in such an environment. From the perspective of my own research and literature review I would add, that making the software usable is another aspect that is similarly overlooked.

2.8 Integration of UCD and SSD

The existing literature directly addressing the gap between UCD and SSD is currently limited (Mohammad, 2009). This sparsity in the literature exists because scientific software development currently lacks a clear defined curriculum. SSD is unlike much more established fields such as project management or software engineering. Consequently, existing research documenting the decisions and process for the scientific software development is limited (Mohammad, 2009). From a software engineering perspective, the work by Hannay *et al.,* (2009) supports this by citing how few publications in software engineering focus solely on the development of scientific software. The re-enforces the previous work by Nuin (2008) and Killcoyne and Boyle (2009) who also discuss that the published work of a SSD project is often a by-product to the central publication of the scientific project.

Research within the scientific community has begun to recognise the shortcomings of usability in scientific software and the role usability engineering has to play in it (Carpenter *et al.,* 2012; Pavelin *et al.,* 2012; Bolchini *et al.,* 2009). However, there is a limited application of specific UCD roles in such projects. The work by Pavelin *et al.,* (2012) has spoken of the lack of UCD in bio-informatics (a field where SSD is undertaken). Pavelin *et al.,* (2012) cite several reasons for this, from the complex interfaces that are presented to the lack of incentive for publishing the UCD work, but also the cost for UCD for academic organisations, which lack a traditional business model. Pavelin *et al.,* (2012) own work underlines the gains that can be obtained from applying UCD practices in user research interviews, one-to-one usability testing sessions, and workshops that provide them with positive user responses for the work they have carried out.

Emphasising this problem further, several funding bodies have acknowledged the lack of UCD practice. The review e-Science 2009 by the Research Councils UK (RCUK) made up of an expert review panel to globally benchmark and assess the strength of the UK e-Science research. The report also highlights gaps and missed opportunities in e-Science. It was noted in the RCUK report how there was a lack of emphasis on usability and UCD. Another report highlighted the deficiency in understanding social obstacles to new technology (EPSRC 2010). The point below, taken from the EPSRC report to underline this issue, was made by one of the panel members about the lack of UCD (the term used in the extract is HCI) for SSD. The panel members also note that there are traditional practices of simply building the software or hardware within the context of e-Science without any consultation or involvement with the user community. This point of concern for the panel is shown in the extract below with their suggestion to move towards a more "human centred approach".

> *"Several of the panel felt that there wasn't enough systematic analysis of the software from a usability perspective and I guess I added the HCI because I associate that with the community and that maybe there needs to be more emphasis on a more human centred approach or a more participatory design process between the user communities. It's quite common for example in the HPC (High Process Content) arena for the people to go out, that run those machines to go out and buy those machines without necessarily asking the user*

> *community whether the architecture meets their needs or not. I think that in the case of the UK that's one of the reasons that HPC here largely means modelling and simulation, it doesn't include data, but it shouldn't be that way and there's much more nuance versions of what I'm talking about."*
>
> (EPSRC 2010)

The Joint Information Systems Committee (JISC), a charity that works on behalf of UK higher education to champion the use of digital technologies, carried out a human factors audit. Their audit of a selection of e-Science projects critically underlined the lack of UCD practices in SSD and they made a call for e-science projects:

> "**START WITH THE SCIENTISTS not the technology** - what are the problems that they want solved."
>
> (Kalawsky *et al.,* 2006)

However, despite this growing recognition for UCD in SSD, there has been little exploration of the application of the actual integration of UCD in the scientific software development context.

The UCD literature does acknowledge the lack of and need for UCD practices in SSD (Javahery *et al.,* 2004; Schraefel *et al.,* 2004; de la Flor *et al., 2010;* Warr *et al.,* 2007*;* Procter *et al.,* 2006*)*. Based on this, it would be expected that the research into this area of integration between them would be moving forward. However, this is not the case, as explained by Segal and Morris (2011), who note that the problem of integration is fundamentally ill defined because of the nature and influence of the end-user.

The role of scientific end-user development is documented in the scientific contexts of financial mathematicians (Segal, 2001), earth and planetary scientists (Segal, 2005) and structural biologists (Segal, 2009). Despite the differences between the domains, the model illustrated in Figure 2.7 shows a common scientific end-user software model of development practice. Segal and Morris (2011) explain that with the scientific end-user software model, both software design and usability become a smaller issue. Software design is neglected because the software is relatively small and can frequently be disregarded once the scientific answer is obtained. The usability of the software and testing and the requirements of the software is not a major issue. Additionally, the end-

user is the developer of the software and the software can have a limited exposure to a wider audience. This perspective does aid my own research as it provides begins to reveal a view of one model of SSD. This model explains a limitation that end-user development has placed on SSD that has limited the approach of applying established software engineering practices and importantly for my research question limited the use of UCD.

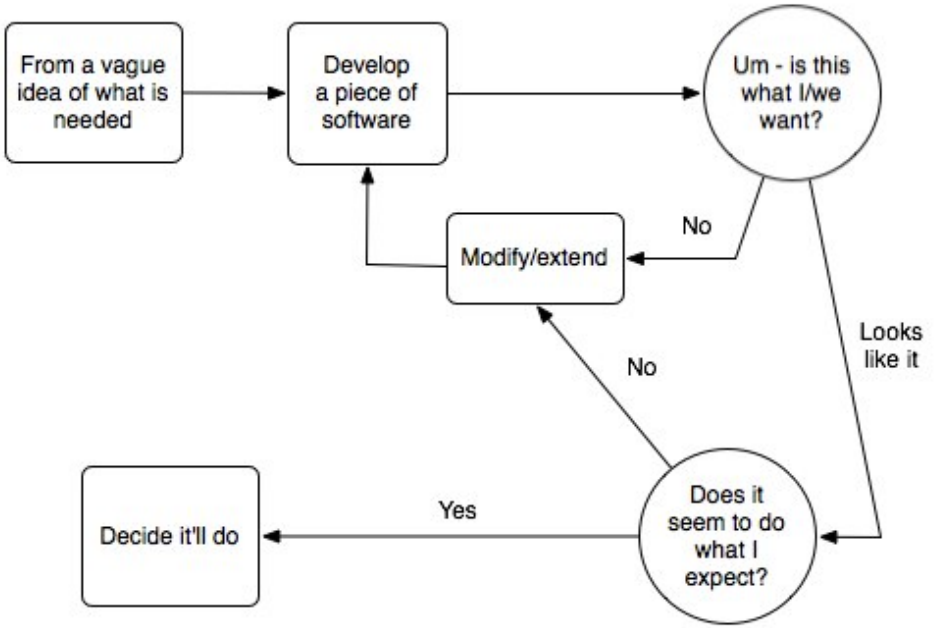

Figure 2.7: A model of scientific end-user software development (Segal & Morris 2011)

However, the scientific end-user software model of software development is not applicable to all levels of SSD. Table 2.1 (page 44) categorises the types of scientific software developers and identifies the work by Killcoyne and Boyle (2009), who are working within a specialised software development team. They are a team with software developer professionals with prior software engineering training and software engineering expertise. Their solution included a set of software processes and design principles for their life sciences research environment. They have a specialist ten-man SSD team to develop the software. This helps to promote a discussion of ideas about the projects developed by the team so identifying the project dependencies within the context in the life sciences research environment. What this means is that they can operate more with practices common to a software development team (Killcoyne & Boyle, 2009).

The reasons there is a need for more specialised scientific software developers practicing the established software engineering process and methods is underlined in the growing complexity of scientific software. The added advantage for scientists in this process is that it allows them to deal with their own scientific questions without needing to make the crossover into understanding the software development process. However, with this the difficultly of UCD and its integration into the scientific software development context is significant, as are the challenges that Killcoyne and Boyle (2009) emphasise regardless of what industry is of dealing with a rapidly evolving domain:

- Skill gaps among team members, with a growing number of developers who have no formal training
- Poor specifications resulting from conflicting use cases and a lack of requirements
- No overall project vision, leading to feature creep or outright failure
- Software developers being expected to play too many roles, including hardware experts and IT support
- Managing the complexity of required technologies and standards

Other relevant UCD work has captured the best practices for SSD (De Roure & Goble, 2009; Letondal and Mackay, 2004; Baxter *et al.,* 2006). De Roure and Goble (2009) discuss the two software systems of myExperiement and Taverna. These two tools are designed for increasingly data-intensive scientific practices. The myExperiment software facilitates the discovery and sharing of scientific digital objects and is comparable to Facebook for scientists but the focus is on scientists' specific requirements, such as the need to attribute work, and link with distributed data collections. Taverna is complementary to myExperiment; it is a software tool that provides automation of scientific data processing tasks, making them systematic and repeatable.

De Roure and Goble (2009) attribute the growth of these tools to six design principles for SSD. The first principle is "fit in don't force change", and this is summarised in the myExperiement motto of "bring myExperiment to the user" rather than force the user to come to myExperiment. What this means is that scientists already using websites and

wikis would find it easy to bring the functionality offered by myExperiment to their existing interfaces.

The second principle is “jam today and more jam tomorrow”. This principle acknowledged the time taken by scientists to use their software. In recognising this aspect, De Roure and Goble (2009) adopted an incremental development approach to give incremental content. This is so that the scientists get core functionality and the software has quicker user uptake of the software. The third principle is “just in time and just enough”. This principle is based on solving the problems defined by the myExperiment project and trying not to be too smart by attempting to build the complete solution. This is to ensure that problems users know they have are solved so they do not have to wait for solutions to a problem they might never have. The fourth principle is “act locally, think globally”. The approach was to target a community the myExperiement team already knew. More specifically, they targeted local pioneers who are stereotypical examples of a class of scientists with a certain type of problems, and they built their system for them. Their findings were that when their local scientists were happy, so were those scientists had not worked with before.

The fifth principle is “let user add value”. This principle recognises that the project does not need to create software extensions but rather provide support and training for others to do so. This principle was applied with the myExperiment team through the software development process with maximal reuse and reusability of the software code. The myExperiment software also supported lightweight programming models for the ease of integration of loosely coupled systems. This principle nevertheless depends on scientists having the skill to develop software or have access to developers in their own laboratory. The final principle is “design for network effects”. This principle involves working with numerous researchers conducting routine processes on a daily basis, hence harnessing this long tail to enable network effects and provide community intelligence. In myExperiment, it was critical to find easy workflows so sharing and adding them to other scientific assets should be straightforward.

The work by Letondal and Mackay (2004) have examined scientific software design within a participatory design context, focusing on the collaborative development of the scientific software called Biok, which explicitly supports end-user programming. Biok allows biologists to manipulate DNA strings and protein sequences, and to visualise

their features. The tool is designed to be programmable by biologists. In working with participatory design in this context Letondal and Mackay define participatory programming as a natural extension of participatory design, in which users participate in the creation of software tools they can ultimately tailor and programme themselves. It is their belief that the design of a tool, not solely its structure but also the process by which it has been designed, plays a significant role in how well it can be adapted to user‘s needs.

Based on this, the selection of core features for the Biok software and the ability to tailor are dual concerns regarding design. Consequently, the better adapted to users needs it is the less tailoring will be required. But at the same time, the easier it is to customise, the more likely it can be adapted as users needs change. Their design process has discussed the following four steps. First, designing for flexibility, this principle was based on the work by Letondal and Mackay (2004 cite Stiemerling *et al*., 1997) who calls for the identification of potential dimensions for evolution and creating an interface for modifying tools. Secondly, the practice of finding potential dimensions for evolution examined then how these features may evolve. Letondal and Mackay (2004) identified the stable parts of the system that would not require any programming, whereas variable parts must be subject to tailoring. Such an example of this in the Biok software was with the visual alignment tool. The observations showed that they were typically inflexible. What biologists preferred was spreadsheets or text editors so they could manually tailor specific sections. This functionality requires explicit tailoring support.

The third principle of design of meta-techniques was based on Letondal and Mackay’s (2004) use of scenarios and workshops about designing meta-level features. Their example describes how scenarios sometimes reveal programming areas as side issues. Their goal, however, is not to describe the programming activity *per se*, but instead to create an analogy between the task and how to perform it, to expose the relevant programming techniques. Because of this, Letondal and Mackay identified different types of end-user programming scenarios with examples, scripting, and command history to use for the design of the Biok software. Finally, the fourth principle is setting the context for tailoring situations. Letondal and Mackay’s (2004) observations of biologists showed that most programming situations correspond with breakdowns. This caused the scientists to reflect on their activities and trigger a switch to programming to

find a solution. It was significant for the work by Letondal and Mackay (2004) to acknowledge this last point, as this is a means of fixing a problem. Therefore, the scenarios they developed played an important role in identifying first when the breakdowns occur and then how scientists would like to work around them.

Baxter *et al.,* (2006) recognise the lack of development practices in scientific software. They also acknowledge that scientific software development brings together different cultures of scientific and software engineering. Critically, their work goals are that specialists and generalists can work effectively on scientific software projects, so they can benefit with the options to increase project efficiency, software longevity, user community acceptance, and translational impact.

They have gone on to suggest a set of five recommendations as a set of guidelines for practitioner's peer reviewers, and project leaders of small (single-lab) to medium (collaborative, academic projects) sized projects for successful scientific software development. The five points that they have drawn on are from working in large scientific software development in business, non-profit, government, and academic settings. These points are subsequently explained below.

The first point is design the project up-front. This describes how software projects should be proactively and thoughtfully designed. This means answering the two key questions of "what will the software do?" and "how will the results produced by the program be verified?" A clear design document should include the software inputs, outputs, and how the program(s), will transform those inputs.

Baxter *et al.,* (2006) describes how the design phase should also account for the usability requirements. If only the programmer uses the software, then the usability of the software might not be a major concern. However, sustainable software academic funding requires a software project to disseminate the work, share tools, and use statistics to help justify renewal of funding. For this reason, usability should be a much higher priority in scientific software development. The second step is documenting programs and key processes. The best practices include that programs should be well documented, modular, and easy to read by users who did not write the program. Such documentation might also include a user guide. The third step addresses the quality

control of the software. The level of quality control requires three aspects: software testing, version control and bug tracking.

The fourth recommendation concerns applying data standards when possible. This recommendation is based on the need to disseminate and share research results with the community to aid scientific progress. The goal is that inputs, outputs, and the results of scientific software are available in standard formats.

Finally, the fifth recommendation is to incorporate project management. In software development projects, a project manager ensures that the software meets a defined set of procedures. Principal investigators in science who will traditionally have no prior background in software engineering may find themselves filling a software project manager role because they supervise people in their laboratory who write software. A project management role in this instance for a principal investigator can be a minor role for a small project involving one or two programmers, although it would require involvement for the full range of tasks including informal design and code reviews, regular meetings to track progress against an established timeline, and reviews (and sign-offs) of testing results. However, this role can easily become more complex for larger projects. A common approach suggested by Baxter *et al.,* (2006) is to break the tasks into manageable subprojects, with a series of release cycles interleaved with user or stakeholder feedback.

In my review of the research by De Roure and Goble (2009), Letondal and Mackay (2004), and Baxter *et al.,* (2006) there is limited application and discussion of such usability engineering techniques as previously described usability engineering methods in section 2.3.1. The importance of the requirement of UCD for user feedback is shown through all three pieces of work.

Letondal and Mackay (2004) describe participatory design and their development of scenarios for tailoring their design and development. The scenarios are integrated for the benefit of end-user development. This use of scenarios in the work draws on existing practices of scenario-based engineering (see Chapter 2 section 2.3.1d). Although not directly discussed in the work by Letondal and Mackay (2004), it can be deducted from their work that the scenarios have supported the description and clarified the relevant properties of the application domain but also help uncover system

requirements. This was demonstrated in how Letondal and Mackay (2004) drew prototyping themes from the scenarios, which were based on interviews and observations of scientists at their own institution. Letondal and Mackay (2004) also discuss the result of three end-user programming scenarios that they formed: programming with examples, scripting, and command history. However, there are no other recognised connections between the range of usability engineering techniques reviewed and the pieces of work examining the best practices for UCD and software development.

With the aim to further understand and explain this, De Roure and Goble (2009) note that the principles 'jam today and more jam tomorrow' and 'Just in time and Just Enough' led them to adopt a perpetual beta software development methodology. This was in part due to the challenge that De Roure and Goble (2009) were facing. De Roure and Goble (2009) expand on this by citing how they have developed systems in the past that were good examples of well-designed software. However, they were still neglected by their intended users. Again this aims to emphasize that scientists have challenging and changeable applications that they might understand but that are hard to communicate or stabilise. It is the nature of these aspects and question that my own research picks up. In the specific context of an expert software development team, I will examine how UCD may be integrated with SSD. I shall do this by utilising my access to the Usable Image project and SSD project of OMERO in my research work.

2.9 Summary

Throughout this literature review, the integration UCD and software engineering has proved to be a difficult problem, as the two fields were formed at different periods of time. Usability engineering later emerged in response to the need to integrate UCD concepts and techniques with software engineering. It has subsequently allowed for the problems to develop; this is exemplified in the discrepancies of terminology that has formed between UCD and software engineering in addition to usability engineering. However, given these problems, there is a well-established area of research that has specifically investigated the integration between software engineering and UCD. The work by Seffah *et al.,* (2005a) and Seffah *et al.,* (2009) consider the collection of approaches that so far have been taken for integration. The range of these approaches

covers software engineering models, formal specification methods, and notation software for supporting the UI development lifecycle. In addition usability engineering has to integrate UCD techniques into established software engineering lifecycles.

Within SSD there is wide recognition of the requirement for UCD (De Roure & Goble 2009; Letondal & Mackay, 2004; Baxter *et al.,* 2006; Javahery *et al.,* 2004; Schraefel *et al.,* 2004). However, as explored in the work by Segal and Morris (2011), there is the recognition that integration between UCD and SSD is fundamentally ill defined because of the nature and influence of the end-user. This problem comes with the further issues of a lack of a structured curriculum for SSD (Mohammad 2009) and a limited focus on the development of scientific software, as the focus on the development of scientific is still be considered a by-product of the central publication of the scientific project (Hannay *et al.,* 2009).

This is not to say that the integration of UCD in SSD has a completely unique context. The seven points summarised in section 2.6 synthesise the established challenges of integration with UCD and software engineering. However, the significance of the SSD context cannot be emphasised enough. The work by Iivari (2006) discusses the contextual issue for UCD with software development by questioning the organisational cultural context and with it brings its unique problems. Further still, in trying to solve the variety of contextual problems and effort to find a generic solution for UCD and software engineering integration is likely to be difficult because of the amount of variable factors for any given integration case (Aikio, 2006). The range of contextual factors identified in the SSD literature has covered the nature of the role and categories of scientific software developers to the influence of funding, exploratory nature of scientific research, and wider community dependencies. This point stresses a call for a more specific solution for UCD and SSD integration against a proposal for a more generic solution for UCD and software engineering integration.

The requirement for understanding the UCD development process is necessary and this would further support the literature in the general identification for further background research and empirical evidence to be undertaken about the issue of UCD integration with software development (Aikio, 2006). The direction of this thesis supports this call

as well as taking the specific step of investigating the question from within the SSD context.

Existing work has captured best practices for SSD (Baxter *et al.,* 2006; Letondal, 2006; De Roure and Goble 2009), yet much of this work highlights the role of SSD where the scientists themselves are more directly involved in the SSD work. However, this work also included end-users in the SSD process. My own research question and goals are distinctly different from such existing work as it investigates an alternative scenario, where there is a dedicated expert scientific software developer team as well as a dedicated expert UCD team. This distinguishes the work from the previously identified work in Table 2.1.

This lack of research documenting the SSD UCD gap has given me the scope to explore the following three research questions:

1. Why is so much of academic scientific software still unusable and/or poorly accepted by scientists?
2. How is scientific software development undertaken in academic contexts?
3. How can the uptake of user centred design philosophies, methods and thinking in the application of academic scientific software development be improved?

Chapter 3: The Teams and Challenges

3.1 Introduction

Technological advancements in the area of imaging, molecular biology and genomics have fuelled a transformation in cell biology. Through this revolution it has quickly become the norm to visualise and measure molecular and structural processes of the cell (Swedlow & Eliceiri, 2009). The research by Swedlow & Eliceiri (2009) also underlines how this transformation has been driven by technology and the specific advancements in computational tools for the acquisition, visualisation, analysis and distribution of the image datasets. This has fuelled the growth in various forms of scientific microscope image processing software in order to manage the acquisition, visualisation, analysis and distribution of the image datasets. The introduction of the OMERO platform is significant in the context of scientific microscope image processing software, as it is an indication of how the image processing software is required to move past image analysis and be able to provide image management.

The OMERO software project is a step towards an image management system for the bio-informatics community by presenting an enterprise level solution software system. This is software that is designed for a particular type of organisation and controls many aspects of the business (Macmillan Publishers Limited 2013). The level of software development complexity for OMERO can be characterised against table 2.1 (Chapter 2 section 2.7.1) and the categories of scientific software developers – the scientific software developers that make up the OMERO project would be categorised as a specialised software development team. This is where the complexity of scientific software development increases beyond scientist programmers (or professional end-users). This shift is representative within the context of the OMERO project as the software is a data management tool for scientific images, so as scientific software moves beyond an individual scientific problem and requires the expertise of software engineers to develop software for the scientific community.

The following details about the developers of the OMERO project, the Usable Image project, and the fieldwork data in chapters 5 and 6 covering the scientists and developers have had the names changed to pseudonyms, expect my own.

3.2 Background to the OME project

The Open Microscopy Environment (OME) consortium was created in 2000 by Sergey a principal investigator (PI), Miles a second PI and Larry, a Post Doc working under Sergey at the Massachusetts Institute of Technology, who all recognised that the growth in biological imaging data was occurring and was only going to increase further (Eisenstein, 2006). The complexity and increase in size of image files along with the demand for supporting metadata[3] are becoming standard requirements for scientific software in order for the bio-image informatics software to provide an enhanced capability for scientists. The OME consortium is developing a standard suite of tools for microscopy image file formats to aid scientific discovery. In addition to this the OME consortium identified a key component was the creation of a universal file format – the OME data model says that one of the major problems in microscopy is the proliferation of proprietary file formats (Eisenstein, 2006). The OME data model continually evolves in order to support changes of existing formats and to support emerging techniques.

The impact on the scientists is that it leaves them tied to work at specific workstations that are compatible with the proprietary file format only. The OME consortium introduced a file format, OME-XML, that both contain the image pixel data and the experimental metadata in a readable XML-based file. Further development to the OME consortium was introduced by the creation of the OMERO software (initial steps June 2006). Since the first two members of the development team joined in 2003 (see Figure 3.2 for the Project Timeline), the software development has been led by Miles, the PI in a biology laboratory based in the Wellcome Trust Centre in Dundee. This role, the OMERO software developers and the project have been an integrated part of the biological scientists "wet-lab team" (led by Miles). The OMERO software has been created to support predominantly biologists working with microscopy images by allowing them to import, organise, view, analyse and output their images and therefore take on the challenges of bio-image informatics software. The OME consortium has centrally been led by the OMERO project, with the OMERO project incorporating the OMERO software and the bio-formats project. The OMERO project setup is presented in Figure 3.1. The discussion of the different components of the software and the

[3] According to Eisenstein (2006), metadata is the data describing experimental information and the acquisition system, links between images, any processed versions of an image, and any analytical results generated about that image.

screenshots show the OMERO.insight view of image data, image analysis, and data manager are shown in Appendix 1.

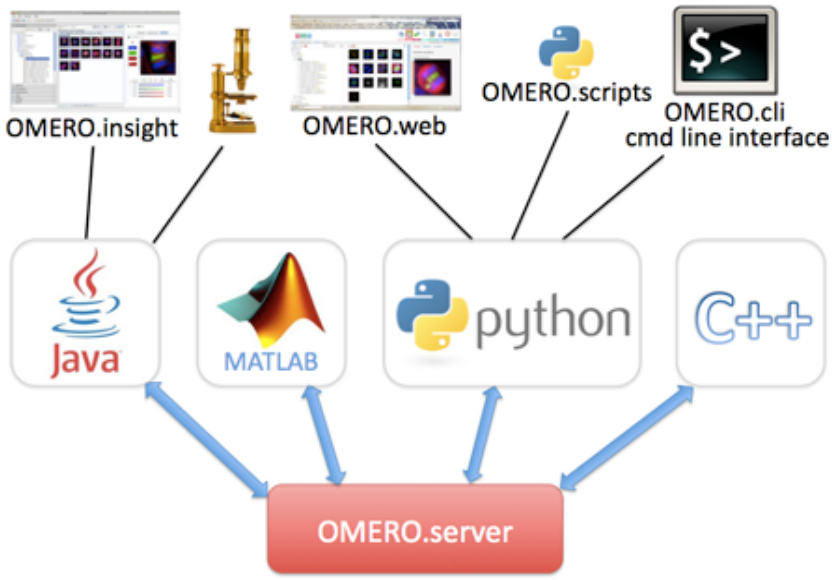

Figure 3.1: The OMERO Architecture. (Courtesy of J Swedlow)

The project goals for OMERO are focused on interpretability, the image metadata and interfaces. Figure 3.1 provides the overview to the OMERO server, which provides the storage, data management, visualisation and quantitative analysis for images. Figure 3.1 also includes the coding languages that the OMERO server has interoperability with. The top level of Figure 3.1 shows OMERO.insight, OMERO.web, web scripts, and the OMERO.command line interface (CLI). These are various ways that allow the users to connect to the OMERO server and access the image data. For this research the interest and focus is with the OMERO.insight and OMERO.web clients, which was the scientific software, developed by the OMERO team to allow the scientists to manage their scientific data.

3.3 The Academic Environment for OME

The OMERO software development process operates within the scientific academic environment of the Wellcome Trust Centre for Gene Regulation and Expression at the University of Dundee. As discussed in Chapter 2, the implication of operating in the academic environment impacts on the development process in terms of how the project is funded. The academic environment guides the software outcomes, directed by the

objectives set out by the research funding bodies. The central funding for the OMERO project is provided by several sources:

- The Wellcome Trust
- Biotechnology and Biological Sciences Research Council (BBSRC)
- Engineering and Physical Sciences Research Council (EPRSC)

The funding environment shapes the development of the OMERO project and how it provides scientists a software tool. The OMERO funding environment does not remain static given that academic funding is typically between 1 and 3 years. Therefore, the working practice of the software development requires the ability to embrace and act on the needs of the wider scientific community and the many voices making up that funding body and scientific community.

3.4 Background to the OMERO Software Team

The OMERO software team is composed of 15 developers, 8 being based in Scotland, (Yvan, Jack, Bob, Steve, Levi, Anthony, Kurt and Cathy) and the remaining 4 contributions to the OMERO project are distributed between Germany (Luis), Portugal (Alain), and the USA (Terry and Eli). A more detailed background to the OMERO developers based in Scotland is discussed in Table 3.1 and Figure 3.2 documents the evolution of the development team throughout the period of the research. In Table 3.1 and Figure 3.2 they document the 8 developers based in Scotland, as these were the developers that were central to my own research interactions. The evolution of the OMERO development team is a reflection of how academic SSD is affected by funding and this can directly influences the team's growth. Figure 3.2 also shows the parallel commercial running project of Glencoe software. This was a company set up in late 2005 to provide commercial support for OMERO server customisation, client customisation, and consulting and support contracts. This, as discussed by Ambati and Kishore (2004), is a frequent model in academic software development as the software has a commercial value. My research work does acknowledge the commercial role of Glencoe Software but for the purpose and the context of this research the commercial aspect has intentionally not been considered. This was because it goes beyond the remit of my research questions.

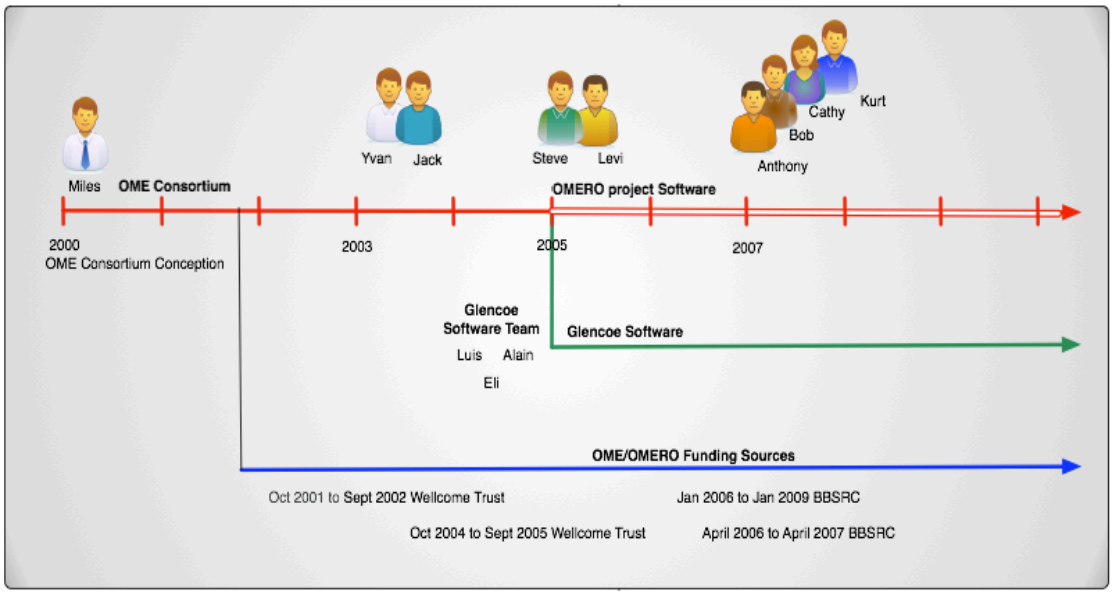

Figure 3.2: OMERO Team Diagram over Time

The project is an Open Source project adopting some Agile development practices. Further details of the development team who are based in Dundee, Scotland are available in Table 3.1 The OMERO project has been built on allowing the software development team to work with the scientists for the purpose and benefit of managing image data within the laboratory and context of the University of Dundee, as well as the recognition that this is a major challenge for the rest of the academic scientific microscopy community. For the time of the project, the development team responsible for OMERO software applications are all co-located in Dundee, with the scientists. The OMERO server and bio-formats data model output do not directly interact with the scientists themselves, so there is no strong requirement for the developers to be co-located. Nevertheless, all of the external developers typically meet with the rest of the members of the Dundee team every 2–3 months, depending on the software development schedule of planning and release.

The advantage of the context of the OMERO project being led by Miles, a light microscopy expert and laboratory PI, is that the historical and current context of the on-going software development work has allowed the core software development team to be co-located alongside the scientists. Although this is not exclusive to all those involved in the project, the core team based in Dundee is able to benefit from this through certain activities:

- The division Monday morning coffee meetings where the biologist and software development teams talk over coffee.
- The OMERO developers have been involved in several lab retreats together with biologists as a team led by Miles, the PI.
- During the OMERO project, several of the software developers have spent a period of time sharing offices with biologists.
- Many of the software developers attend biology seminars.
- Software developers and scientists regularly meet in the cafeteria over lunch.
- A scientist (Bob) has moved across to take a full-time role in the OMERO software development team.

These factors have all helped to enrich the SSD environment, and they help to forge a scientist-software developer relationship. In the software development process, with the software developers being embedded in the scientist's environment, they have been able to comprehend why and where the requirements are requested. This close relationship has helped to gain an understanding of the local users of the software, thus aiding in the provision of strong insights into the nature of the scientific work that scientists are involved in. This context holds similarities to the work by De Roure and Goble (2009), who describe how their own scientific software developers were working closely with onsite customers. The insight gained from being embedded in the environment for up to 7 years for several of the software developers in the OMERO team has led to a very strong understanding of the scientific context.

However, despite this embedded context of the OMERO team working alongside the scientists and with feedback from the funding body for the OMERO project, the OMERO team recognised that there was a need to supplement the team's domain and technical expertise with additional UCD input.

Consequently, the Usable Image project, a multi-disciplinary group including design ethnographers and interaction design specialists, was formed to support the OMERO development team but also investigate novel approaches to UCD in a complex environment (Sloan *et al.,* 2009).

3.5 The Academic Research Software Development Environment in OMERO

The OMERO project employs 8 full-time scientific software developers based in Dundee (Jack, Yvan, Steve, Levi, Bob, Anthony, Cathy, and Kurt), with a range of diverse experience and background expertise (for a detailed description of the full-time developers in Dundee see Table 3.1). Outside of Dundee are the four software developers, who work in direct collaboration with OMERO: Luis, a software developer working at home, who works on the OMERO Server based in Germany; Alain, a developer working at home (in Portugal), who works on the commercial web development of the project; then there is the Bio-formats team of Terry and Eli (based in the USA), who are software developers working at LOCI, a biophotonics instrumentation laboratory.

Table 3.1: Background of the full-time developers in the OMERO Team

Name	Description
Jack	Jack is a software developer and systems administrator from Vancouver, British Columbia. He joined the Swedlow laboratory in early 2003 and is one of the well-established members of the team. His early contributions were the initial implementation of mass-storage infrastructure and network groundwork to support the University of Dundee's Light Microscopy Facility (LMF), but he now spends most of his time working on the OMERO.server project. One of Jack's phrases which he recalls from the early days of working on the project, was that he would not become involved in the programming; these were his famous last words, as he puts it, as his role now covers some of the software development of OMERO and systems administration, managing the releases and packages of the OMERO software, and involvement with the OME file formats. Jack gained most of his systems administration and software development expertise as a Vancouver high-school student working in the commercial sector doing security, systems, and network consulting for local companies and as a contractor/employee in the GT Trust high-security solutions team of the Group Telecom Services (now a division of Bell Canada). His interests lie in the area of distributed computing, secure programming practices, and encryption. When not with his head down in Java, C++ or Applied Cryptography, he can be seen playing for Team Fife in Scottish Volleyball Division 2. Jack's partner is a PI who is also based in Dundee; she gives Jack that extra insight into the life and troubles of running a laboratory as well as a stronger background to the scientific work in which she is involved.
Yvan	Yvan joined the staff of the Miles laboratory in 2003 along with Jack; he is one of the longest serving members on the team. Yvan's background and area of expertise is in mathematics; he received his PhD in

	mathematics from the University of Brest in 2000. His research focus was in the area of harmonic maps. After completing his PhD, he went to work in a private company as a software engineer and later took up a post-doctoral research position with the Geometry Group at Lund University, Sweden. Yvan's role in the project is as the developer of OMERO.insight. Outside of the OMERO project, Yvan enjoys refereeing the rugby pitches of Scotland, since after retiring from playing several years ago he took up refereeing.
Steve	Steve joined the Dundee team in early 2006 and is currently developing the various import tools used by OME. Originally an aeronautical engineering student, he migrated to software development in the early 90s and has been programming ever since. He spent most of his early programming days working in the private sector, primarily in the telecommunications industry. As a Canadian, when he is not working on OME he enjoys cycling, hiking, snowboarding (when he can find a snow-covered slope) and playing social video games.
Levi	Levi started work as a developer on the OME project in January 2006. He received his PhD in Computer Science from the University of Paisley. His research background is in Statistical Natural Language processing, Data Visualization and Image Classification, specifically Remote Sensing images. Levi has worked in the academic context both as a researcher and lecturer; he has also worked in the commercial software development context. Outside of work, Levi enjoys the outdoors and can be seen cycling or walking through the Scottish countryside on most weekends. He also enjoys making the most of the Scottish mountains for skiing during the winter.
Anthony	Anthony joined the OME project in 2007 to manage the data model and project documentation. Anthony also has the responsibilities of managing the OMERO project promotion and marketing work of the project. His role ranges from the organisation and planning of the project posters and leaflets that are presented at the conferences to the organisation of travel details. His promotional role is possible because the OMERO data model is on a different release schedule to that of the OMERO.Client. During his background in software development, he worked on games, e-learning, and personal development applications. In his spare time his interests include historic re-enactment and costuming, and he is very handy with a sewing machine. He has typeset and published an illustrated book on armour making. He studied at the University of Dundee and the University of St Andrews, and he has lived in Fife since 1989.
Cathy	Cathy joined the OME project in 2007 as a software developer. Her role in the project is the development of the OMERO.web client. Before joining the team, Cathy studied at the Universite d'Artois in France at UE Socrates-Erasmus student and the Technical University of Lodz where she received her Master degree in Computer Science, Engineer. Her specialisation is in internet technologies. After graduation she has worked on e-learning, enterprise, and personal development web applications in the commercial market. In her free time, she relaxes her body and mind: riding a bike, windsurfing, skiing and mountaineering. She also enjoys watching F1.
Bob	Bob started in Dundee as a cell biologist to do his PhD and then joined

	Miles' lab as a Post-Doc in 2003. Bob's role in the team is unique. As a result of being interested in the OME project from a user's point of view, he decided on a change of scene and left the lab to do an MSc in Applied Computing at Dundee University. He then returned to Miles' lab for his MSc project. His goal was to make it easier for biologists to record their experimental metadata in a digital form. This was the start of the OMERO.editor development, which continued when he joined the OMERO team as a developer in October 2007. Bob is a father of two girls and when he has free time, his other interests include mountaineering, sailing, and motor biking.
Kurt	Kurt is the newest member of the team, joining the OMERO project in late 2007 as a Python developer. Originally graduating in physics, and then computational physics, he has worked in psychology, physiology, and astronomy as a developer of scientific software. In his work in astronomy, where he was involved in developing software for telescopes, Kurt even spent some time working with the telescopes in Hawaii, which he talks about regularly, and makes everyone very jealous. His role in the OMERO project has focused on the OMERO server and the development of the OMERO.FS DropBox that allows for automated import of files into OMERO. Kurt has fitted in particularly well in the OMERO team; his active lifestyle and interests include cross-country skiing meetings, orienteering, fell running, and the occasional mountain marathon.

Miles, the PI, leads the OMERO development team. He plays a very active role in the project and is renowned by the microscopy community through his active participation in conferences and meetings around the world. His background and experience in the multidisciplinary field of biophysics allows a bridge into the understanding of the software development process. This experience is valuable when conducting technical communications with the software development team. The scheduled communication among the team covers a range of meetings; the following table provides an overview of the scheduled meetings and occasional meetings that take place in the OMERO software project (Table 3.2).

Table 3.2: OMERO Team Meetings

Description	**Details**
UI client Meeting	Wednesday 10 am each week. This is attended by the OMERO software development team and the Usable Image team. This involves the people present in Dundee and the remote members of the project connecting into the meeting via Skype/ TeamSpeex.
Developer Conference Call	Friday 2pm each week – held via TeamSpeex

	conference software or Skype tools to allow discussion with remote members. The focus of this meeting is specifically led with the ongoing technical developments of the project, as well as any outstanding project organisation issues.
OMERO Team Meeting	Face-to-face meeting with OMERO team based in Dundee Scotland.
Developers Dundee Meeting	Face-to-face meeting with the discussion of the current and future developments of OMERO.
European User Meeting	A regular yearly meeting at the Institut Pasteur in Paris to discuss the software with current and potential new users.
American Society for Cell Biology (ASCB) Conference	This is an external meeting once a year other academic and commercial meetings.
Meetings with External Scientific Institutions/ Laboratories	Infrequent meeting may be planned with interested laboratories through the arrangement by the PI.

Through my participation in the Usable Image and OMERO project I attended and was involved in all but one of the meetings described in Table 3.2 (the American Society for Cell Biology meeting). It was from this involvement in the range of communications that I was able to select the second round of fieldwork for the research (the full details of this decision are documented in Chapter 4. What was significant in my involvement for me as the researcher was that it helped lay a foundation for a broad understanding of the project. From client meetings, which were more focused on user feedback, to developer meetings, which were very much focused on technically. Meetings with external institutions focused on a mix of technical and user perspectives.

Table 3.3 provides a comparison of the OMERO project with the traditional features of academic software research and commercial software projects. The particular distinguishing features of the OMERO project are evident in its purpose: it aims to serve primarily as a practical tool for data management in the scientific community. However, in working in the academic context it also enables the development of the work to have some scope and contribution to the research context. Table 3.3 also highlights a further characteristic of how the OMERO team of programmers are a step away from the traditional characteristic of SSD project where novice student programmers are used (Ambati & Kishore, 2004; Liu *et al.,* 2008). The OMERO team is made up of a combination of experienced software developers with a variety of technical expertise. Consequently, the OMERO team's approach to the software process

and products is distinctly similar to a commercial approach than an academic approach. What distinguishes OMERO is that because of the experience of the software developers in the project, the development process involves group meetings, code management, testing, documentation, and a project management schedule. These are all aspects of scientific software development that do not apply to the previously described model of scientific end-user software development (Segal & Morris, 2011) (see Chapter 2 section 2.7). In making this comparison I aim to further distinguish the position of this research and the type of SSD project that the OMERO is.

Table 3.3: Comparison between Academic Software Research and Industrial Software Projects against the OMERO project (modified Liu *et al.,* 2008)

Project		**Academic Research Software Projects**	**Industrial Software Projects**	**OMERO Project**
Purpose	***Purpose***	Research oriented, sometimes even no practical benefits	Practical benefits	Primarily practical benefits, does serve some research goals through research papers
	Scientific Contribution	Often research deep exploratory question, new idea with scientific contribution	Often no scientific contribution	Has a contribution with an OS setup of a client-server management technology in the domain. Contributes to new algorithms for image analysis techniques
People	***Programmer Expertise***	Novice	Middle expert	Middle/high-level expert
	Team Management	Very flat	Loose co-operation Hierarchy	Very flat
	Programmer turn over	Normal predictable turn over	Unpredictable turn over	Normal predictable turn over
	Designer	Professor does not join the actual design	Software designer	Professor **does** join the actual design plus the addition of the Usable Image Research Team
Process	***Development Process***	Agile most likely	Various: waterfall, Agile	Agile practice
	Group Meeting	Scarce or as needed	Regular	Regular twice a week
	Code Management	Depending on individuals, often not used	A system often used	A system used
	Budget	Little budget needed, a big issue	A big issue	A continuing big issue
	Testing	No systematic testing	Special testing team testing	Systematic testing with a special period, performed by whole OMERO team.
	Schedule	No strict even no defined schedule	Defined deadline	Defined schedule with 2 major releases a year.
Products	***Character of Software***	Single facet	Often multi-facets.	Multi-facets.
	Size and Complexity of Software	Small to big; algorithms can be complex	Medium to large; both design and implementation can	Medium to large; both design and implementation can be

			be very complex	very complex
	Interface	Not much attention to the interface design	More requirements on the interface	Working with a specific user experience team
	Documentation	Little	Complete often	Little to middle, strongly connected to funding with what resources maybe available
	Open Source	Traditionally Open Source	Traditionally propriety/licensed software	Open Source

3.6 Background to the Usable Image Project

The Usable Image (UI) project was also situated in the academic context of the School of Computing at the University of Dundee. The UI project was a separately funded 3-year academic research project, supported by the EPRSC and began in September 2006. The two project teams in Dundee were operating between the School of Computing and the Wellcome Trust building. This is shown below in Figure 3.3.

Figure 3.3: The Locations of the Projects

The members of the UI team are described in table 3.4. The team is composed of 5 members with skills ranging from design ethnography to usability (Table 3.4).

Table 3.4: Roles in the Usable Image and Background to the Usable Image Team

(Usable Image 2010)

Name	Background
Hester	Hester is the co-investigator and research manager for the Usable Image team. Her expertise is as a design ethnographer. Hester's educational background covers a BA in Communication Studies, an MSc in Information Systems and a PhD in Computing. She has also spent time working in broadcast news analysis, community work of various kinds, and international emergency relief. Hester has led the set-up of the Interactive Media Design degree programme at Dundee University; this programme crosses the School of Computing and the School of Design. In her spare time she has outside of looking after her two young boys, she enjoys photography, writing and sailing around Scotland's beautiful coast.
Ernest	Ernest is the interaction designer for the Usable Image Project and is responsible for translating user needs into practical information that can inform the development of OMERO. Ernest originally trained as a cartographer, graduating with a BSc (Hons) in Topographic Science from Glasgow University and working for four years with mapmakers before coming to work in Dundee. Outside of work, Ernest feeds his addiction to maps through the sport of orienteering; he also enjoys curling and football as well as supporting his beloved Aberdeen. Other interests include spending time with his family, cooking, listening to music, and drinking nice wine, ideally all at the same time. He is also project lead of the Digital Media Access Group, a research and consultancy unit based in the University of Dundee's School of Computing, which provides advice on accessibility and inclusive web and software design for commercial clients. He currently co-ordinates the University of Dundee Web Accessibility service, and he completed his PhD in 2006. His PhD was in investigating the effectiveness of web accessibility audits for inclusive web design.
Leo	As the Usable Image team's design ethnographer, Leo explores life scientists' environments using qualitative approaches in order to understand what they do and why. This has allowed Leo to do what she enjoys in simply listening and talking with the scientists. Leo's background is interdisciplinary: humanities, cultural studies, and social sciences. She has a BA (Hons) (Nanjing University) and MA (Fudan University) in English Literature and a PhD in media and cultural studies (Cardiff University). Leo's has previously worked as a researcher with the Health Protection Agency (London), Cardiff University, Nottingham Trent University, and as a lecturer at Fudan University (Shanghai). Her role has covered discourse analysis of literature, risk migration, and community and health. Outside of work, when she has time, she enjoys visiting the local independent cinema as well as reading a variety of literature. She also enjoys travelling and particularly likes to spend time life drawing.
Danielle	Danielle is a part-time Usability Expert in the project. Danielle has an academic background in science. She originally studied Applied

	Biology at the University of Hull, followed by an MSc at the University of Aberdeen. She then went on to complete her PhD at the University of Edinburgh where her work focused on the effect of climate change on plant/fungal root interactions. She has been working part time at the School of Computing at Dundee University for the past 2 and a half years, initially on a project looking at how to improve computing for older people. This prompted an interest in Human Computer Interaction, which led to her undertaking a part-time MSc in Applied Computing while still working within the department. The opportunity to work in the Usable Image project has enabled her to combine her interests of computing and biology. In her free time she enjoys skiing, painting and drawing, and growing things in her garden that she can eat. She also spends quite a few weekends in cold muddy fields either watching her son compete at motocross or her daughter riding.
Scott	My Role in the project is as the PhD student on the Usable Image team. My own background is in Computer Science with my undergraduate degree in Computer Science and MSc in Computing and Software Technology. I also have work experience in Information Technology support and in a commercial software development environment. This background has exposed me to a range of different programming languages and development techniques, as well as given him/me an insight into software development culture. My interest in interaction design came through the observational and investigation work into the re-design of a medical syringe pump device. This opportunity led to the enjoyment and fascination of questioning the context for designing technology. Through my role in the Usable Image project, I have been implicated in the investigation of the gap between UCD and academic scientific software development. In my free time I love to travel and experience new cultures, and I am attempting to learn French.

The methods used by the UI project are discussed later in this section (see Table 3.5). Being in the academic context, the UI team was able to benefit from the exploratory nature of the academic research work, but was also constrained by the funding model of academic research, similar to the constraints that the OMERO project is operating under.

As identified and discussed in Chapter 2, the application of UCD within scientific software is still being established. The Usable Image team has promoted user-centred design in the OMERO project through a range of methods (for more details, see Macaulay *et al.,* 2009). Traditionally, the adopted usability engineering methods include user evaluations and heuristic evaluations (Nielsen & Mack, 1994), which have

typically been applied at strategic points of the development cycle. Beyond the role of advising on the usability of the software under development, we also adopted techniques that were less focused on the interface and more on the users – current users and potential users. This was driven by the need to better understand the diversity of the user community targeted by the software developers who, while situated in a life science research institution of some 700 employees and being managed by a professor who also leads a team of scientists, had a relatively restricted view of the scientists for whom they were developing. While there was awareness among developers regarding the range of technical requirements and scientific practices across the institution, it was acknowledged that the day-to-day coding demands meant they did not always have the resources to extend this awareness beyond mining existing relationships they had with scientists (for example colleagues they met for coffee or had dialogue with via email). The issue here then was one of resources and tools, as within the team there was already a significantly user-oriented mindset.

3.7 The Methods of the Usable Image Team

The background of the fieldwork contribution was made accessible through the background work conducted within the UI team. The range of methods conducted by the Usable Image team is listed in Table 3.5.

Table 3.5: Usable Image Methods

Name of UCD Method/Technique	Description	Reason of use
Ethnographic Observations/ User Observation	A qualitative data report both written and some visual data. The ethnographic observations may be task focused or involve background information about scientists.	The ethnographic work is conducted whenever an opportunity occurs to provide a rich account of scientists' work practice or when a specific area of investigation arises for which we would like more information from scientists.
Usability Test	The interface is evaluated against the following formal interface review methods: Heuristic evaluation (Nielsen's 10 usability heuristics); Accessibility guideline check (IBM Software	The methods provide insight into the general usability of the software interfaces.

	Accessibility checklist); Interface consistency check (icons, menus, tooltips, labels, language); Keyboard operation check.	
Group Taster Session	An informal demonstration of OMERO functionality	Provides a preview of the software to a group of scientists for feedback that can be collected by the UI team.
Individual Getting Started Session for a Scientist	An individual walkthrough with a scientist that covers the process of installing and getting started with OMERO.	This is used when a scientist expresses enough interest to want to start using OMERO.
Focus Group	A structured discussion around topic of enquiry.	The central purpose is to explore specific issues that have been identified from other Usable Image activities or question any conceptual issues from the OMERO team. Such an example was the understanding of the scientific workflow through the software.
Design Workshop	Structured discussion around a specified design topic.	A further technique used when a more specific interface or presentational issue has been identified from other UI activity.
Requirement Gathering Meeting	A structured discussion around the requirements in question.	This is conducted when an understanding of a specific requirement is needed.
Other Discussion with Scientists	This activity covers any unstructured discussion held between Usable Image researchers and current or potential users.	The technique can be used to help support requirements for future development.
Discussion with Developers	Informal discussion with OMERO developers.	A conversation with the OMERO developers would happen when the developers require further information about the scientists. Alternatively, a discussion may happen when the Usable Image team need to share findings with the OMERO development team.
Survey	Survey to find out working practices of scientists.	The survey has been used to aid the scope of the background information of the Usable Image project. The benefit for the use of the survey allowed access for a wider set of scientists and

		different scientific institutions.
Interface Review	The work will examine examples of existing software/web interfaces as inspiration for how others have solved the problem facing OMERO.	This is conducted when an interface issue requires research into other relevant software interfaces.
Requirements Definition Meeting	The Usable Image team contribute feedback from the range of Usable Image activities.	This is used to aid the definition of requirements in advance of a new development cycle.
Software Outreach and Promotion	The Usable Image team along with the OMERO team participated at local and external academic events to promote the use of the software.	This was directly used to increase the uptake of the software and where possible allow for further usability feedback.

Usable Image user research has included employing an ethnographer Leo, who produced, over an 18-month period, a collection of ethnographic stories mainly focusing on the scientists and their day-to-day work activities. Example stories included working through a particular experiment, using a complex microscope for the first time and the role of the lab-book in work group discussions on attitudes to sharing data. This work provided a rich source of information on which to base future user research activities, and it established connections with the scientists beyond the lab, most of which were directly associated with the development project. Another activity in which the Usable Image team played a lead role has been in outreach and "marketing" of the software, extending the uptake of the software to people who could become local "Champions" of the software. This was working in a similar way to the principal described by De Roure and Goble (2009) of "act locally think globally" (see Chapter 2 section 2.8). It was realised that while there was some awareness within the institution of the OMERO project, this was not guaranteed among the scientists who were potential users; of those who were aware of the project, there were sometimes misassumptions and underestimations about the usage potential beyond the laboratory in which it was being developed. Additionally, beyond the host research institution, we had contacts with other organisations which we knew little about, other than that they were potentially valuable sources of new users.

The interactions of the Usable Image team were led by the UCD methods of the project. The work was directed by two aspects 1) the ethnographic observations and 2) the continuing requirements for to aid the design of the OMERO software. The second aspect was accounted for by the other methods employed by the Usable Image project, as well as the ethnographic observations supporting systems design process. The output of the work by the ethnographer was the user observations. The information gathered through this work was presented as stories. The value of the existing ethnographic work provided an initial foundation for insights of scientific working practices. The ethnographic observations were also reviewed and discussed by the entire Usable Image team during key phases of requirements gathering. The analysis of the ethnographic work carried out by the UI ethnographer is examined in Chapter 5. The Usable Image team was also in part involved in the promotional led work for the OMERO software. This corresponds to the role and aim of the Usable Image project in the optimisation of the usability of the software, but it also aims to provide for evolving requirements based on the user research, to increase OMERO's marketability to new users (Sloan *et al.,* 2009).

The survey work carried out by the Usable Image project had the aim of questioning how life scientists approach the task of capturing, storing, and analysing biological images (Sloan *et al.,* 2009). Three web-based surveys were created: two were constrained to specific research institutions (the first institution was where the OMERO project was based and the second was a major research institution in Europe) and the third was open to all respondents using a microscopy email list.

The survey work was directly aimed at supporting the information for the OMERO development process. This was evident in how surveys one and two were examined for key trends, and in feedback that the OMERO developers lacked sufficient detail. This subsequently helped define questions (along with the other Usable Image activities) for survey three of areas of investigation that required further information.

The range of usability testing carried out by the Usable Image project in Table 3.5 cover the techniques of "individual getting started" session for a scientist, "design workshop", and 'usability testing'. The "individual getting started" session for a scientist was a technique to observe the users of the OMERO software and understand where in the interface the problems are. This was complementary to the long-term method of

ethnography work carried out in the Usable Image project. During the usability testing the Usable Image team was able to work with the pre-released beta OMERO software. The type of usability testing was determined by the way in which the Usable Image project was acting as an external usability consultant to the OMERO project. This links back to the work previously outlined by Battle (2005) and the pattern he described of a foot in the door for external consultants (see Chapter 2 section 2.3). With this particular pattern the usability is limited to targeting the latter phase of the UCD cycle. This is reflected in many of the usability techniques used by the Usable Image project.

Of the usability engineering methods reviewed in Chapter 2 section 2.3, the Usable Image project did not directly consider these techniques. Though the Usable Image project did with the use of the heuristic evaluation draw on the discount engineering method.

3.8 Summary

In discussing the background to the OMERO project and the Usable Image team, Chapter 3 has presented the environment and context in which the research is being conducted. The position for the research is to investigate of the role of UCD in SSD through the participation of the Usable Image team and OMERO team. Chapter 3 has discussed the application of the Usable Image methods and techniques used within the project.

The critical set of challenges identified through the OMERO project lie within the integration between the Usable Image work and the OMERO team. This integration lies in supporting the OMERO team through the SSD process embedded in a scientific environment. The setup of the OMERO project, in contrast to existing projects, is significantly different from the role of traditional academic SSD projects, with a team of dedicated developers co-located within the same building as scientists. This presents the challenges of UCD working with expert users in the scientific context and expert users for the scientific development process of such complex software. Finally, in being situated within the scientific domain the research must also take into account the level of scientific credibility and responsibility to the scientific community of the software. Chapter 4 now goes on to present the methodology of this research work.

Chapter 4: Methodology

4.1 Context of the fieldwork

This chapter describes the research methodology. The first part outlines the research perspective formed from within the Usable Image project and particularly examines the secondary analysis of the ethnographic work the Usable Image team undertook. The remaining parts reviews the methodological choice made during this work and details the approach taken for data analysis.

Many of the methodological choices made were driven by the fact that I occupied various roles and had several perspectives during this work. As illustrated in Figure 4.1 below, I viewed the research question from the perspectives as an individual PhD student, a member of the Usable Image team, and later a member of the OMERO team. Although occupying multiple perspectives was challenging, it proved a fascinating opportunity to view the research problem from many points of view that helped to provide a platform for cross analysis.

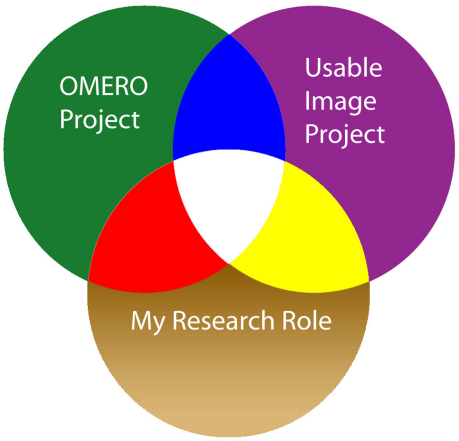

Figure 4.1: My roles during the research

4.2 My role within the Usable Image team

My role within the Usable Image team was in the participation of the support and development of OMERO software: I provided expertise in a range of user centred design techniques, each aimed at understanding more about the users and usage environment of the software. In this role, I participated in diverse UCD activities:

design workshops, interface reviews, requirement specifications, design and technology research, user insights, usability studies, and software taster sessions. However, I was not directly involved in the ethnographic observations and user interviews either internal or external to the University of Dundee where the main part of this research took place. These ethnographic activities will form the secondary analysis and produce the first part of this research (study one). Although I did not directly participate in the collection of this data, I did conduct the analysis described in the analysis for this this research methodology. All the activities undertook for the Usable Image project are illustrated in Figure 4.2 and the description of these methods is documented in Chapter 3 (section 3.6).

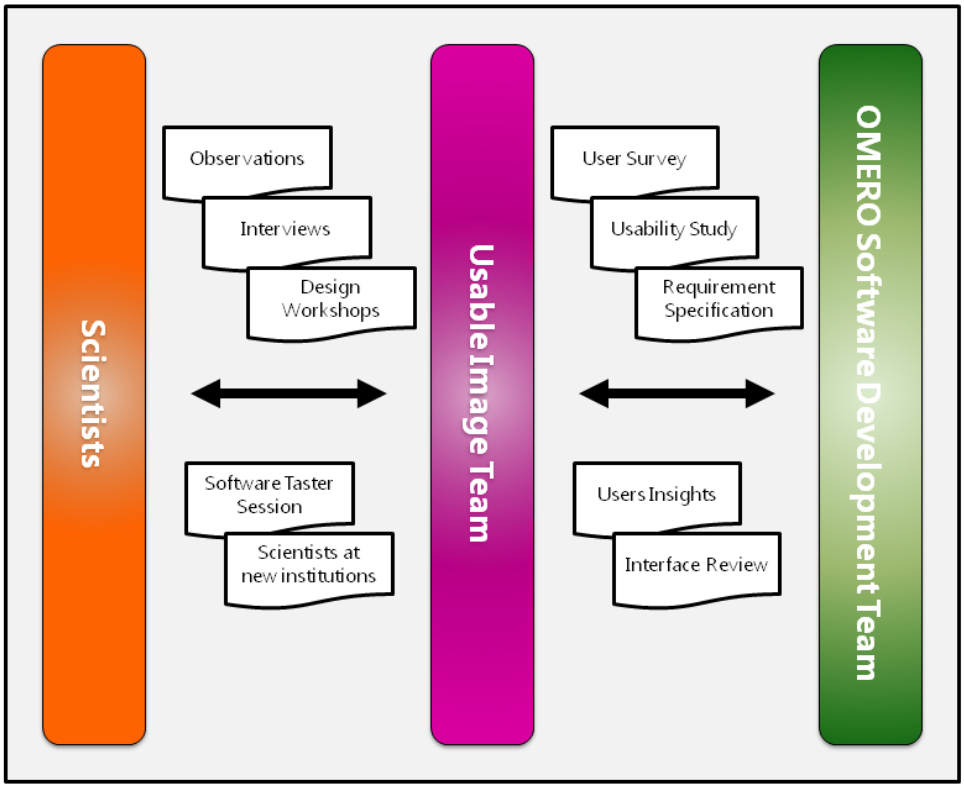

Figure 4.2: Scope of UCD led techniques used by the Usable Image team

As Figure 4.2 shows, the methods listed on the left hand side between the scientists and Usable Image project were used by the UI team to interact with the scientists. The methods on the right hand side between the scientific software developers and Usable Image project were used by the UI team to interact with the scientific software developers (see Chapter 3, section 3.7 for details of these methods). Indeed, the direct participation of the Usable Image team with both the scientists and the OMERO software development team helped build a corpus of insights for the development of the

OMERO software. More specifically, the Usable Image project team worked with the OMERO team within a weekly user evaluation cycle (as shown in Figure 4.3). This contact is illustrated in Figure 4.3 between the Usable Image team and OMERO team (and indeed between both teams and the end-users). The exchange and discussion of user feedback was done during a weekly meeting of the two teams. An example of such feedback is presented in Appendix 2, where there is a selection of points made during user-feedback sessions on the software.

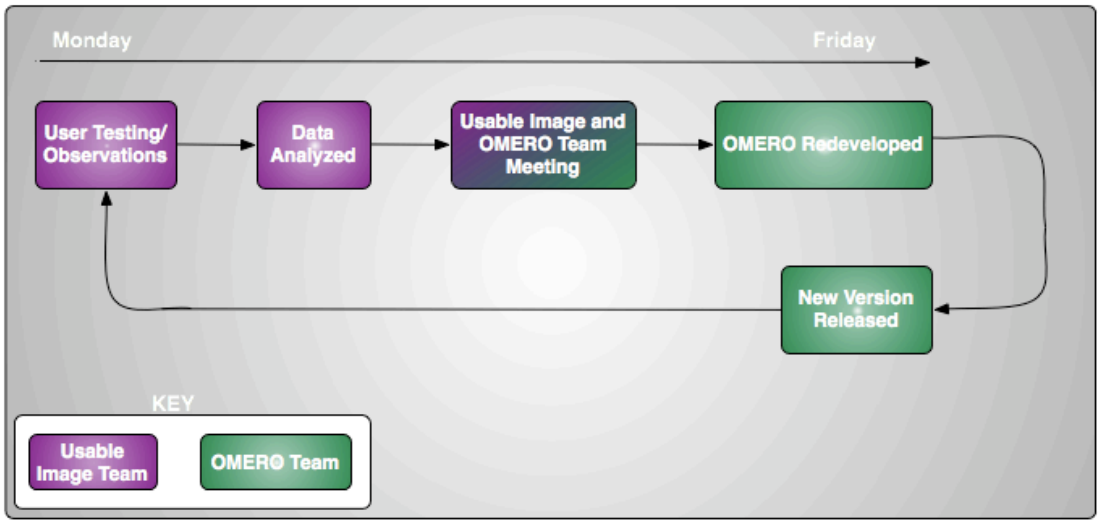

Figure 4.3: Weekly evaluation cycle (Adapted from Macaulay *et al.,* 2009**)**

This interaction between the Usable Image project and the OMERO scientific software development team, as illustrated in figure 4.3, aided an understanding UCD for the OMERO project. This 'relationship building' was important as what was becoming more apparent to me in my embedded role within the Usable Image team, which has been previously argued by Seffah *et al.,* (2005a), is that UCD and software engineering are indeed very different perspectives. These different perspectives are shown in Figure 4.4. In the 'System 1' perspective, the focus is on the interface and the application of the software; in the 'System 2' perspective, the vision is more global and aims for an integration between the users and the interface/application. The OMERO team, broadly speaking, adopted a 'System 1' perspective, whereas the Usable Image team (equally broadly speaking) adopted a 'System 2' perspective. These differences were not just philosophical: they had a significant impact on how the software development project management was defined, the activities that were conducted, the selection of tools used, and the staffing of each team.

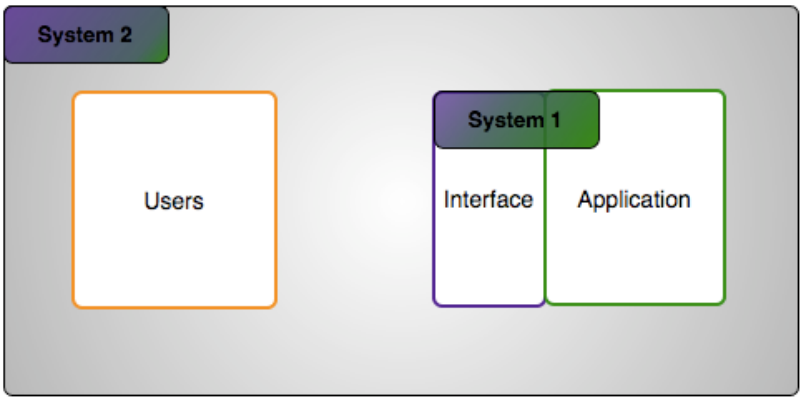

Figure 4.4: System perspectives (Seffah *et al.,* 2005a)

For the fieldwork and the analysis, the data comprises two sources of data. The first part (study one) has been taken from the ethnographic observations of the scientists from the Usable Image project and the OMERO project meetings the second part of this research. Both studies are discussed later, in Chapter 4 section 4.5.1.

4.3 My role within the OMERO software development team

During the usability work within the Usable Image project, I moved to work alongside the OMERO software development team. While I was still able to participate in the usability activities of the Usable Image project it also meant I shifted my own perspective, from the focus of the Usable Image team to the OMERO team. The major change meant a change of location, as I moved to the Wellcome Trust building, where both scientists and OMERO software developers were located (see figure 3.3. in Chapter 3 for the location map). This transition to the OMERO software development team was aided by my previous contact with the work in Usable Image project, so I was working more closely with people who already knew me. Thus, my integration into the group was less intimidating and the team knew I was there to work and support their work from my existing work in the Usable Image project. My integration into and relationship building within the OMERO team was also facilitated by following the usual routine of working in the OMERO team (see Chapter 6 section 6.2) for a summary of working week within the team). This way was significant, as it allowed me to have an insight into the regular practices of the group. The importance of embedding in a team was also underlined by Agar (1980, cited in Lazar *et al.,* 2010). These insights into SSD work practices were crucial as they allowed me to form a complementary dual

perspective with my time in the Usable Image project. The full implications of this double perspective for my research are discussed in Chapter 6 section 6.6.

4.4 Strategy of inquiry

The philosophy of UCD, as described throughout Chapter 2, incorporates a broad history of multiple disciplines. All these have influenced the direction UCD has taken and what it is today, and such a scope is relevant to the methodology used for this research. I will therefore provide a short overview of significant milestones for UCD.

As UCD has its origins in HCI and its history follows a related path from HCI and software development. This context influenced the motivation of this work, because in software development the focus should be on the integration of tools and methods (Carroll, 2003). This has perhaps emerged from previous difficulties. For example, the problems experienced during the 1970s where there was little or no learning about the mistakes in the software development process and was known as the software crisis (Naur & Randell, 1969). This challenge of the software crisis led to the study of computer programming as an activity, where the first studies from the viewpoint of HCI were of those programming computers (Carroll, 2002). This marked the early work in HCI with the introduction of an influx theories and methods for systems development. The first wave of HCI borrowed from cognitive and psychology approaches.

One of the key methods formed in this period derived from work by Card *et al.,* (1983) and their Goals Operators Methods Selection (GOMS). The method provided a prediction based on the various factors of GOMS to anticipate the actual task. The GOMS model is made up of methods that are used to achieve goals. A method is made up a successive list of operations that a user performs and sub goals that must be completed.

In light of these changes that brought theoretical developments into HCI, the second major milestone stemmed from the disciplines of anthropology and sociology. Examples of second wave HCI and frameworks that emerged during this period are ethnomethodology (Suchman, 1987), phenomenology (Winograd & Flores, 1987; Dourish, 2001), activity theory (Bødker, 1991; Nardi, 1996), distributed cognition

(Hutchins, 1995), and grounded theory (Fitzgerald, 1998). This second wave saw a shift from the fields of human factor's research and cognitive science and it has been characterised by Bannon (1991) as being "*from human factors to human actors*". Because of the criticism of HCI and call to recognise users as humans rather than cognitive machines, this should be echoed in the concepts and theories used (Kuutti, 2001). Bødker (2006) says that this second wave of work focused on understanding group working. She further highlights that theory focused on work settings and interaction within well-established communities of practice. The range of theories that emerged through this second wave was a reflection of this; one of the first was outlined by Suchman (1987) in her book *Plans and Situated Actions*. She discussed the observation of a photocopier and its use, and the usability problems associated with its interface. The outcomes from the study helped form the concept of situated action (defined as a term to account for every aspect of action, dependent on its material and social circumstances).

The 2000's brought the more recent wave of theoretical development that sought to address the challenges caused by technology moving out of the work place and into different domains and contexts. Bødker (2006) notes "*conceptually and theoretically, the third wave HCI focuses on the cultural level*". This covers a variety of different emergent techniques including the move to the pragmatic focus on experience (McCarthy & Wright, 2004), aesthetic interaction (Petersen *et al.,* 2004), empathy, and emotional design (Norman, 2004). The move into the third wave provides a stark contrast to the second wave of challenges with the more recent third wave having attributes of being non-work related, non-purposeful, and non-rational (Bødker, 2006).

As my first two research questions concern why so much academic SSD is still unusable and/or poorly accepted by scientists and how SSD is undertaken in academic contexts, I have chosen ethnography and a qualitative method for analysis. This draws on the second major theoretical milestone that stemmed from the disciplinary fields of anthropology and sociology. Ethnography is based on the notion that true understanding of complex human practices and context requires in-depth and engaged study (Lazar *et al.,* 2010). For my own research, this presents a suitable approach for answering these two particular research questions stated above because of the role of understanding the complexity of scientific and scientific software development practices. Furthermore, using ethnography means that opportunities from contextual situations can be built on,

instead of avoided (Harvey & Myers, 2002). So for my own research this can mean utilising the experiences gained from the research. The choice for an ethnographic approach was also informed and influenced by my role in the Usable Image project, where like Sloan *et al.,* (2009) the project was already using ethnography to inform the systems design process. I also considered my research position when deciding on this approach, as it required a technique that would not conflict with the projects I was working within. Ethnography therefore allowed me to utilise my role within the Usable Image project and the OMERO project. My experiences within the Usable Image project where I had participated in various usability testing activities (see section 4.2) also had also benefited and encouraged the selection of ethnography. It provided an introduction to the context of the OMERO software and scientific context. The use of ethnography was also relevant as it provided a different level of focus from the usability testing activities I had been involved in with the Usable Image project (see section 4.2), particularly as I had been involved with user testing. Siegel and Dray (2005) describe the difference between ethnography and usability testing is that usability testing focuses on evaluating solutions whereas ethnography focuses on understanding problems. The application of ethnography in my research therefore aims to allow me to provide an understanding of the SSD context by being emerged within it first-hand. This also helps to situate the research for questioning the social organisation of activity within SSD. This further helps to support my particular research question of understanding how SSD is undertaken in academic contexts. The section below will describe the ethnographic fieldwork and subsequent analysis of the fieldwork for the research.

4.5 Ethnographic fieldwork and the construction of the methodology

Ethnography is a research technique adopted from sociology and anthropology; it has been used to observe human interactions in their actual social setting. The work by Burke and Kirk (2001) distinguishes the different goals of sociological ethnography and ethnography for systems design. The goal of sociological ethnography is to understand an individual's or a group's interactions within the culture. The goal of ethnography for systems design is to understand and improve a system in the context. In my own research, the aspect of understanding and improving a system in the context was being applied while working within the Usable Image project, as we were actively using ethnography to support the systems design process for the OMERO software. However,

my own ethnographic and research goals are centred on my own research questions. Therefore, my individual research goals were not solely focused on the system design process of the OMERO software.

In both areas of my research, the use of ethnography has grown. Since such work by Suchman (1987), ethnography within the field of HCI has become widespread, with applications in a broad spectrum of contexts from understanding email management (Bellotti *et al.,* 2003), collaborating and sharing photographs (Crabtree *et al.,* 2004), and organising systems in family life (Taylor & Swan, 2005) to controlling air-traffic systems (Bentley *et al.,* 1992).

As discussed by Potts (1998), the application of ethnographic studies to software engineering has also become more widely recognised. Perry *et al.,* (2000) discuss a general growth of empirical-based studies of software development over the last 10–20 years. This has covered various areas of the software development process from software testing (Denaro & Pezze, 2002) and bug tracking (Zimmermann *et al.,* 2007) to the adoption and evolution of software quality management systems (Sharp *et al.,* 2004). The viability and benefit of empirically based studies of software development are also recognised by major commercial software vendors (Bird *et al.,* 2011; Basili *et al.,* 1994). Work by Kitchenham *et al.,* (2001) formed preliminary guidelines for empirical research in software engineering. These guidelines have been created of supporting researchers, reviewers, and meta-analysts in designing, conducting, and evaluating empirical studies.

Empirical software studies have not been without their criticism. Weyuker (2011) cites one aspect that is significant for this study. She recognises the earlier limitations placed on software researchers and how they had little representation of real-world software practice to use in empirical studies. Empirical studies frequently used university students for their research so representation in these was regularly questioned. Now, with the wider development of open-source software, this limitation is no longer the constraint it once was. Nowadays, software researchers have full access to a range of different open source projects, which allows for a much wider view of the open source development process (Paulson *et al.,* 2004; Xie *et al.,* 2009). This is significant to my own research method as the OME project is open source. Consequently, my research benefits from open public access of the OME project material; it also has the benefit of

working alongside professional scientific software developers from within a UCD team. My methodology can utilize the OME project material with its open and online access.

The following section now outlines the purpose of the original study of the Usable Image project and the process of the original work. Chapter 3 previously noted that the Usable Image team's role within the OME project was to support the development of OMERO by providing expertise in a range of user-centred design techniques, and by pointing out how the software may support users' needs. The role of the Usable Image project is also further described in the work by Sloan *et al.,* (2009). The ethnographic fieldwork therefore informed the design process about a more holistic understanding of a system's current and potential users, and its usage environment and context. The observations were anonymously written up into short stories, and they were shared with the rest of the Usable Image team. The stories were published on the Usable Image project wiki and made available to the OMERO development team. The OMERO team were then encouraged to read and react to the stories. The points arising from the analysis were discussed between the Usable Image and the OMERO team. It is important to note that these guidelines from the Usable Image project form a prerequisite to the secondary analysis (for details on the secondary analysis, see section 4.6).

4.5.1 Data collection

This thesis has used two resources of fieldwork data for the analysis. The first used the existing work of the Usable Image project. The goal of this was to create a holistic view of scientists and their work. (See Appendix 3 for the full timeline of ethnographic fieldwork data analysed for this research). The Usable Image ethnographer gathered this fieldwork and I have analysed it for this thesis. The first ethnographic story analysed was dated 27.02.2007 and the last 20.06.2008; a total of 11 ethnographic stories were analysed. The analysis examined each ethnographic story by the date acquired. Please note throughout the remainder of this thesis that the fieldwork carried out by the Usable Image ethnographer is referred to as fieldwork study one.

The second fieldwork resource used was the meeting notes of the OMERO team. This data documented/recorded decisions throughout the project. (See Appendix 4 for all the

fieldwork data analysed for this research). The first meeting notes analysed were dated 27.10.09 and the last 23.03.10, with 27 meetings being analysed. Again, the analysis studied each meeting notes by the date they were acquired. Please note throughout the remainder of this thesis that the fieldwork conducted from within the OMERO team is referred to as fieldwork study two.

The decision to use the existing material created in the OMERO software project drew on Norman's (2007) work where she utilised the material of the project, from emails to meeting notes. As the OMERO developer meeting notes used for study two are an archival data source and the data is static and impersonal. The benefit to this as explained by Lazar *et al.,* (2010) is that as a researcher you are able to take your time reading the data and it can help avoid asking inappropriate questions. An advantage was that all members of the OMERO team were based in Dundee. I was also aided by multiple takes on the meeting data, as I was attending the meetings in person but was then able to go over the data and further digest information. The implication for my own research work was that this did allow me to ask follow up questions and clarify aspects of information from the meetings. However, I had to account for the drawback of such a resource that Angrosino (2007) mentions: these materials can be error prone, incorporate a bias, or be incomplete. The choice was also based on a two-week review of the material in the OMERO project and what information would be accessible for me, the researcher, from working more directly within the project. I was aware of some existing tools of the OMERO development from my work within the Usable Image project, but the time spent working in the same office and seeing the work and tools used by the OMERO developers gave additional insights into what information would be most appropriate for my research questions. The two candidates at the end of this two-week period were the team's group instant messenger tool and the meeting notes. The meeting notes were eventually chosen because of their more formalised summary, insights into the project, and contributions from the entire team. In contrast, the group messenger tool was used to communicate more specific technical issues, which for the purpose of this work would miss the wider holistic picture that the meeting notes offered. The meeting data was also openly available for public access through the project's website as part of its open source philosophy (see http://www.openmicroscopy.org/site/community/minutes/conference-calls). This availability of the data presented no ethical problems regarding its use as the information was already in an open domain. In contrast, the messenger tool would have

had ethical implications for the project, as the information was not openly available. Moreover, the feasibility of analysing both sets of data was considered but was deemed impractical because of the amount of data created each day through the messenger tool.

Hall (2004) makes a significant point about using meeting notes for an analysis. For example, the originators of the documents (in the case of this research, the OME team) do not anticipate their use for academic research purposes so is that the meeting notes analysed have been created for the purpose of the OME project rather than for my research. Hall (2004) describes that the meeting notes have greater authenticity and credibility as a data source. In that sense, the meeting notes used in this research are not biased toward the final goal of this research and were created in complete objectivity. The main disadvantage Hall (2004) reported is that identifying the meeting documents that give important insights is an extremely labour-intensive process. However, I overcome this in my research by analysing the meeting data I was present for, which covered about 4 months (October 2009 to March 2010) so 27 meetings were analysed, which is not as extreme as Hall suggested. In using the two data sources in my research analysis a clarification has been made for using primary and secondary data analysis. Identifying each data analysis as either primary or secondary involved consulting work by Jary and Jary (2000, cited in Smith, 2008). In this, Smith defines secondary analysis as ‘any inquiry based on the re-analysis of previously analysed research data’. This may or may not be by a creator of the data. Primary analysis concerns data collected first hand and analysed directly. For my own research the primary data source corresponds to the OME meeting data. I was not the creator of this data but was present during all of the recorded OME meeting data. Smith (2008) nevertheless acknowledges a lack of consensus regarding the secondary analysis definition but some points are applicable to the present work. Here, as I was present for all the material collected in study two but NOT study one, I therefore identified study one as a secondary data analysis and study two as a primary data analysis.

4.5.2 The process of analysis process

As noted in the previous section, the ethnographic material was sourced from two different resources (the Usable Image project for study one and the OMERO project

meeting notes for study two). Before discussing the role of the secondary analysis, I shall first explain the global analysis process used for study one and study two.

When selecting the analysis method, I have considered the work by Dourish (2012). Within the role of ethnography and design, Dourish (2012) states that there is no single answer to understand ethnography in the design process. Instead, ethnographic work at the conceptual level may work best, not by providing answers but by raising questions, challenging existing understandings, and creating new conceptual understandings. Dourish states that this can be explained by the way ethnographic work can be used based on solely looking at the implications for design. This point is explored in the work by Dourish (2006) and has been discussed in Chapter 2 section 2.2.3. The significance of this debate discusses that the implications for design are not always a necessity for the use in ethnography.

In the recognition of this work described by Dourish (2012), the application of ethnography in the research has been applied in light of accepting how ethnography is evolving to be used within design process. As described by Dourish (2007), ethnographic research may inspire design practice, but the value that it offers is in an encounter with design rather than in its own terms. Such is that the implications for design lie not within the ethnographic text itself but rather in the way in which it reframes the contexts and questions of design.

This has had the effect in my own application of the ethnographic method as a way to reframe the context of SSD for questioning the role of UCD. Considering the work by Dourish, as a trainee in qualitative research, I also opted to apply this recognised use of ethnography as a design process and use a generic qualitative analysis based on two points. First, the selection of my analysis approach was based on allowing the concepts to emerge from the raw data. This way, they can be categorised using a coding process. This will provide a descriptive, multi-dimensional, and framework for my later analysis. Secondly, knowing the emphasis of my own research work, I avoided diversions, for example with the broad scope of analysis models my research could employ. As an inexperienced researcher in sociology methods, these techniques aided my research and such example work by Dey (2003) and Seidel (1998) served as useful guides, particularly as I sought to draw on a collective set of principles to analyse qualitative data.

The process used in the present work for the analysis of fieldwork data was taken from Seidel (1998). This work provided a simple foundation for the complex and rigorous practice of qualitative data analysis and has been used as a guideline for this work. This complexity of data analysis is also recognised by Seidel (1998) who also acknowledges the variety of methods for qualitative data analysis but it is the three principles of collecting things, noticing things, and thinking about things. Figure 4.5 captures the process Seidel (1998) describes based on these three points.

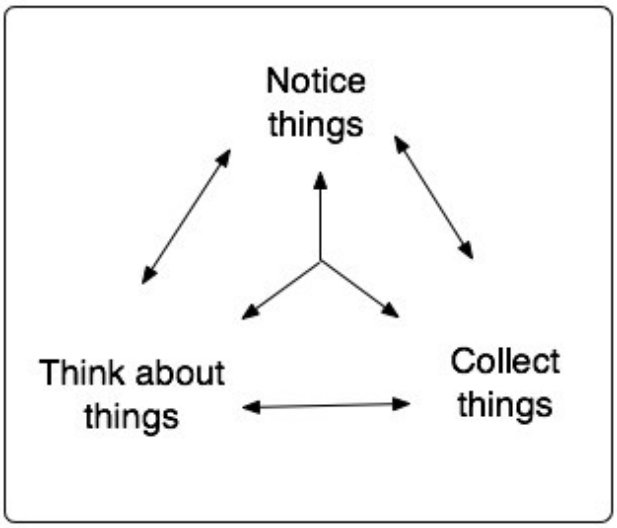

Figure 4.5: The data analysis process (Seidel, 1998)

The core set of principles is captured in three steps: collecting things, noticing things, and thinking about things. These steps are not carried out in a linear fashion but instead have three processes: 1) an iterative and progressive process spiral because it is a cycle that keeps repeating; 2) a recursive process as it has the ability to iterate back to the previous part, and 3) a process that is holographic as each step in the process contains the entire process. This final step is connected to the other steps because even when you first notice things, you may be mentally collecting and thinking about those things. The step of noticing things is principally involves taking notes of the context so you have a record e.g. making observations, field notes, interviews *etc*. Seidel (1998) identifies the steps of the production of the data, the reading the data, and the noticing of things in the data, following these steps allows for the data to be coded.

In the context of this research, once I had decided on the data to be analysed, I carried out an initial coding cycle (see section 4.5.3); in addition, memos were also recorded to support the coding process (see section 4.5.4). For the second step of the process, the

collection and sorting of the codes utilised the *HyperResearch* software, which is described on page 97 below. This particular step involves the separation of the data into elements. With this breakdown of information into more manageable elements, the information can be sorted and searched for types, classes, sequences, processes, patterns, or wholes. The goal of this process is to reconstruct the data in a meaningful way (Jorgensen, 1989 cited in Seidel, 1998). The third and final step of the process has three key goals: 1) making sense out of each collection; 2) looking for patterns and relationships both within and across collection(s); and 3) making general discoveries about the phenomena being researched. This process is illustrated in Figure 4.6 below where the key points of collect, notice, and think are shown. Additionally, the figure illustrates the aspects that emphasis the iterative nature of the process shown in arrows that move around the labels of think about things, reading of the case files from the coding of the data files and then the discoveries.

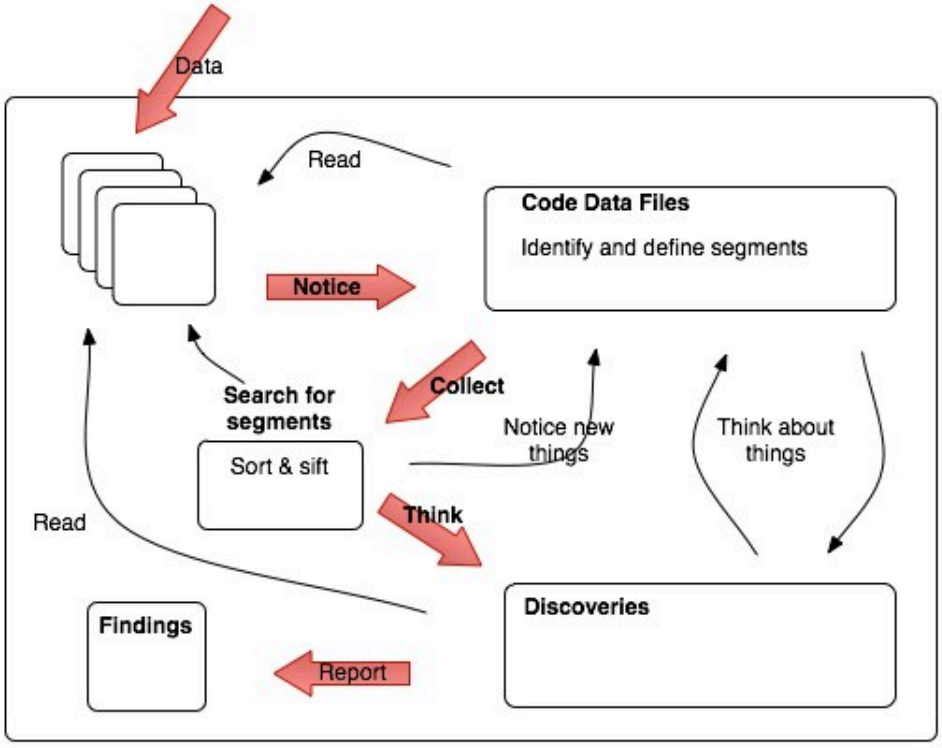

Figure 4.6: Analysing qualitative data (Seidel & Friese, 1994, cited in Seidel, 1998)

Seidel's work (1998) was formed around the development and use of the *Ethnograph* software. As I did not have access to the *Ethnograph* software but did have access to *HyperRESEARCH* (ResearchWare, Inc), the research therefore used *HyperRESEARCH* to analyse studies one and two. The *HyperRESEARCH* software was also more modern

and so also had more general benefits from the ways of inputting data to viewing the analysed data. The use of the *HyperRESEARCH* software replicates the use a computer-supported software tool developed specifically for the process of qualitative analysis. Stern's work (2007) highlights how computer-supported tools can help in the management of data. However, Glaser (2005, cited in Stern, 2007) also notes that creativity is required in the coding process and that sorting the codes by hand releases the necessary creativity to help draw out the memos, and it helps for the comparison between codes carried out during the coding process. Therefore, as well as working with *HyperRESEARCH* I used a pen, paper, and post-it notes but also a diagramming process to develop the core categories (see Appendix 5). The data analysis was done by building the typologies of the codes (see Appendixes 6 and 7). Throughout this process, the typologies allowed continual questioning of the data to identify gaps in the data after each cycle.

The particular challenge presented in the process of 'thinking about things' is in organising the work to minimize negative effects. The problem is that by breaking down data through the coding process, it can actually misrepresent the data and mislead the analyst. This can be overcome by working with two copies of the data: a whole and a broken-up version (Wiseman 1979 cited in Seidel 1998). For my own research, the *HyperRESEARCH* software tool enabled me to readily overcome this problem as it allows multiple views of the data, both in its entirety or as a specific set of codes.

4.5.3 The coding and categorisation of the data

A code sets up the relationship between the data and the people being studied (Star, 2007). The coding of data involves defining what the data actually is. For my own research, I have considered the following set of coding styles of line-by-line, incident-by-incident, word-by-word, and focused coding. For my own coding process, I selected for the first pass through the data to use the incident-by-incident and word-by-word coding approaches. The further iterations of the coding process took the approach of focused coding. The coding styles are explained below.

The initial incident-by-incident coding process, as described by Charmaz (2006), means that codes formed during earlier incidents are compared to those that have already been coded. The technique to code on an incident-by-incident approach was chosen over the alternative of line-by-line because although the latter gives insights regarding what data

to collect next and prevents immersion in the respondent's perspectives, it is best applied on rich and more descriptive data such as interviews (Charmaz, 2006). The ethnographic stories and meeting notes that this research was dealing with worked like observational notes, where line-by-line coding did not fit well. This applied to study one and study two. An example extract from the incident-by-incident coding process is shown in Table 4.1 (for the full transcripts of study one and two, please see Appendix 8). The first pass through the data also used the word-by-word coding approach, this allowed me to attend to the complex terminology in the analysis and to make sense of the various scientific and software development terms.

Table 4.1: Example code study one

Code	**Source**	**Material:**
Technology trouble shooting	070719EthnoObs_Graeme.txt	Graeme summarised several problems with the software. It didn't allow the user to customise the information and save the settings. Every time when Graeme opened a new image, he had to click and choose again.

Further iterations of coding were in the form of focused coding. This process filtered out the most significant and frequent codes that had occurred during the coding process. Both fieldwork studies moved through an iterative analysis of the coding process; this gave the focused coding rich and well-defined codes from the analysed text. The coding tool *HyperRESEARCH* directly supported the focused coding process as it presented the analytics of the codes; therefore, viewing the most frequently used codes of the two separate analyses of fieldwork was easy (see Appendix 9 for a print out of the statistics). On this basis, the three most frequent codes were chosen for each theme and were explored in Chapter 5 and Chapter 6. The decision to analyse the three major codes was based on the fact that they give a better reflection of the fieldwork. Every code could not be explored in detail in my analysis in Chapter 5 and Chapter 6, as this was a total of 64 codes for study one and 58 codes for study two.

4.5.4 Memos

Memos were used to support the coding process. Glaser (1998) says memos give the researcher the opportunity to reflect on the codes created. The creation of memos for my own research was built on this principle, to allow for the introspective thinking for the generation of my own codes and analysis. The role of memos is further described by McCann and Clark (2003) where they discuss a memos ability to reflect a researcher's internal discourse at a point in time. The relevance of the memos to my own research work was with the purpose to help to guide me towards my analysis but also given my relative inexperience with dealing with qualitative data as a way to ensure that my thinking of the codes were externalized. The techniques of how to write memos is explained by Charmaz (2006) and in this work she suggested that memos are written in a manner that works for the individual. For my own research, I have adopted this approach based on the guidelines provided. Moreover, she also states that for creating a memo a researcher can do various things:

- Define each code or category
- Spell out and detail processes subsumed by the codes or categories
- Make comparisons between data and data, data and codes, codes and codes, codes and categories, and categories and categories
- Bring raw data into the memo
- Provide sufficient empirical evidence to support the definitions of the category and analytical claims about it
- Offer conjectures to check in the field setting
- Identify gaps in the analysis
- Interrogate a code or category by asking questions of it

  (Charmaz, 2006)

A sample from the memos formed in the analysis of study one is shown in Table 4.2; the same for study two is in Table 4.3. The full memo tables are shown in Appendix 10.

Table 4.2: Memos extracted from study one

Description	**Personal notes and extracts from the text**

Long hour culture (*Documenting the long working hours that scientists work – point made experiments repeated 3 times as one source of this.* Critical in having insight into work – helps in having a role of empathy for their scientific work.	Rachel usually does one experiment three times to make sure the consistence of the cell behaviour. St Kilda2 This system, while aiming to encourage outstanding performance, also adds on the pressure – as Joseph described, 'People are working like slaves.' Indeed, long hours are common: it is not unusual for people to spend a good 10-12 hours on a workday and extra visits to office during weekends. But this doesn't seem unusual

Table 4.3: Memos extracted from study two

Description	**Personal notes and extracts from the text**
Current work practice - *This describes how the team is currently working capturing both the tools that are being used and their own identification of where the breakdowns are.* *Needed to define how working with new tools*	2010-01-19 Tuesday meeting - practices - coding, style, docs, etc. - not breaking trunk, etc. - refactoring - training - planning - more than ticket writing - we could formalize with "weak" / "strong" reqs - have done it before with planning poker, but didn't for 4.2 - we don't schedule at all - going back to iterations! with demo. - tools - simple: shouldn't bog down the process - visible to the community - planning/scheduling/prediction - visualizing what's in the system and not just as tickets - prioritizing: we've basically ignored this - provided list were the most sophisticated - Luis: adjusting our existing tools has a non-trivial cost - Levi: we still haven't defined a process

4.5.5 Validation of the ethnographic approach

Ethnographic research results are frequently unreliable and lack validity (LeCompte & Goetz, 1982; Seaman, 1999) so such issues need considering for this research. Seaman (1999) discusses how triangulation is an important tool for confirming validity of the data. Data triangulation is where different types of methods and data (quantitative and qualitative) are used to validate the findings of the analysis. The types of triangulation considered were method triangulation and data triangulation. The former uses alternative methods to support the existing method such as survey or experimental methods.

Data triangulation assesses validity when multiple data sources are used in a single study. Thus, the data is supported by multiple sources from multiple perspectives, different people, and multiple locations (Guion *et al.,* 2011). For my research, data triangulation was implemented in study one, but not in study two as the focus of the SSD work was in Dundee. The data used in study one had access to different roles of scientists. The variety of roles covered PhD students, technicians, post-docs, and PIs. Study one also observed scientists from two separate institutions (Skye and StKilda). The option considered for study two was searching for a second SSD project but I decided against this, as it would introduce more variable factors beyond my control. The possible variables identified were the different levels of experience of the scientific software developers in any project and the very different project aims and goals of any given SSD project.

Internal validity of the ethnographic method is described by LeCompte and Goetz (1982) as "*the extent to which scientific observations and measurements are authentic representations of some reality*" (LeCompte & Goetz, 1982). It is supported for various reasons. Firstly, the informant interviewing is formed more closely to the empirical categories of the participants. Secondly, observing the participants in their natural settings provides a reflection and insight into the actual working experiences than any contrived setting. Finally, the ethnographic analysis process exposes the research to repeated questioning and re-evaluation. Altogether, it allows for the repeated integration of the research data. Although I have not interviewed the participants for the data I have analysed, I have applied the last two points in the method, which has allowed for the continual questioning of the research data and insight into actual working laboratory practices.

External validity is reported by LeCompte and Goetz (1982) as having five major issues: researcher status position, informant choices, social situations and conditions, analytic constructs and premises, and methods of data collection and analysis. A **researcher status** questions research members of the studied groups and what positions and access they hold in the field of observation. This emphasises that those who have little access to the fieldwork site would have entirely different results from those who have a role on the inside of the fieldwork site with plenty of close connections. The second point of asking who the **informants** are is significant as it questions what types of people are represented and what different groups of people are observed in the fieldwork. This particular problem for my own research was handled in such a way that my informants were beyond my full control for my research as the data from study one and the secondary data were taken from the Usable Image project. The role of the informants for study two was not questioned as the full representation of the OMERO SSD team is documented in the meeting notes. The third element recognises the **social context** in which they are gathered. This accounts for what information can be appropriately revealed. I was able to account for this in study one as Leo accounted for feedback and correction from those interviewed. The OMERO team has an open review of the meeting data, so I did not consider this aspect any further for my research. The fourth step of **analytic constructs and premises** plays a key role in reconstructing a study. The work included in this covers all the codes and themes constructed for the research are in the Appendixes 6 to 8; the steps carried out for the coding process are documented in this Chapter section 4.5.3. Finally, related to point four is **the method of data collection and analysis**. This point is associated with principle of presenting the details of the research method and has been described in this Chapter so that fellow researchers may follow the research. The reliability and validity of the analysis will be revisited in Chapter 9 to address the possible further improvements (see Chapter 9 section 9.3).

4.6 Secondary Analysis

This section describes the background to the secondary analysis, which was the approach used to direct the secondary analysis of the ethnographic observations from the Usable Image project. This was the fieldwork data I did not collect myself but have analysed in study one.

Secondary analysis, as data generated by other researchers or even in other projects, has a long history in the social sciences. Early work by Glaser (1963) notes that secondary analysis carried out by an independent researcher could offer new insights into social knowledge. Since then, various authors have addressed the scope and discussed the role of secondary analysis. For example, Hinds *et al.,* (1997) comments on its application in the exploration of interests distinct from those of the original analysis, as well as for the analysis of an extract of a sub set of cases for a more focus study. Heaton (1998) describes the use of secondary analysis to apply a new perspective to or give a new conceptual focus on the original research issues. This factor is reflected in my own research as the data used in study one provided a new conceptual focus for the data. The original focus for the data was for the UCD work to inform the systems design process in the Usable Image project. In using the data in my research, I have clearly asked different questions both in how is SSD undertaken in academic contexts and further still how can the uptake of UCD philosophies, methods and thinking in the application of academic SSD be improved. Further related work that has been used to help in the categorisation of my secondary analysis is discussed by Heaton (1998). The work categorises types of secondary analysis summarised in Table 4.4 below. This singles out three types of analysis against three permutations of data sets that have been reviewed against the data for my own research. The first analysis identified is additional in-depth analysis shown through cells 1a, 1b, and 1c in Table 4.4. Heaton (1998) cites work by Bull and Kane (1996) and Kirschbaum and Knafl (1996), showing how their studies investigate in detail problems encountered from original data. Both these existing studies fall into 1b, in regards to table 4.4.

Table 4.4: Types of secondary analysis (Heaton 1998)

Main Focus of Analysis	**Single Qualitative Data Set**	**Multiple Qualitative Data Sets**	**Mixed Qualitative & Quantitative Data Set**
Additional in-Depth Analysis	1a	1b	1c
Additional Sub-Set Analysis	2a	2b	2c
New Perspective Conceptual Focus	3a	3b	3c

The second category is the analysis of an additional sub-set from the original study (or studies). The variations of this are covered through cells 2a, 2b, and 2c in Table 4.4.

Such an example cited by Heaton (1998) is the work by McLaughlin and Ritchie (1994) who use multiple qualitative data sets in their secondary analysis about claimants of Invalid Care Allowance where the focus of the original research issue was on ex-carers. The third and final category is a new perspective and retrospective analysis of the whole or part of a data set. This type of secondary analysis involves examining concepts, which were not central to the original research. The variations of this type of study are covered through cells 3a, 3b, and 3c in Table 4.4.

The type of secondary analysis that has been used for this research is the category of new perspective/conceptual focus. I have taken this strategy in my research as my goal of understanding how SSD is undertaken in academic contexts contrasts with the original overarching goals of the Usable Image project which are to provide expertise in a range of UCD techniques and to understand how the software may support user needs and be further developed to meet user needs (see Chapter 3 section 3.6 for a full overview of the Usable Image study).

4.6.1 Challenges of secondary analysis

Hinds *et al.,* (1997) identified two key challenges in utilising secondary analysis: 1) to what degree the available data is amenable to a secondary analysis, and 2) how far the purpose of the secondary analysis can differ from that of the primary study without invalidating the effort and findings. Hinds *et al.,* (1997) go on to explain a range of further issues that must be considered when adopting secondary analysis as a method:

- Consent of those involved
- Completeness and quality of the research data.
- Sensitivity of the research to the context
- Care of the researchers involved
- Currency of the data set

The discussion and background experiences described in Chapter 3 highlights how I came to the secondary analysis with various experiences including working with the scientists to record usability feedback and alongside the OMERO developers during meetings about the Usable Image project. This helped me to understand how I could

address the issues Hinds *et al.,* (1997) raised. In my position as the researcher based within the Usable Image project conducting the secondary analysis of the ethnographic observations, I had a greater insight into the data creation and more confidence with the data, since my own research interests were very close to those driving the Usable Image ethnographic work. The following points now address each issue raised by Hinds *et al.,* (1997) in turn.

- Consent of those involved

In working under and within the context of the Usable Image project, the appropriate consent to use the ethnographic material was acquired, as I was part of the project team entitled to access the data generated. The consent form used in the project is available in Appendix 11.

- Completeness and quality of the research data

This aspect, as explained by Hinds *et al.,* (1997), can be managed by an individual collecting the material. With the role I occupied while working in the Usable Image project, I was in a position to assess the quality of the data, as I was gathering the secondary data. The material was assessed based on the qualitative research experience of Leo carrying out qualitative research, the ethnographer, who was gathering the fieldwork (as previously discussed in Chapter 3 table 3.4). Her background and previous training is within social sciences as well as cultural and media studies; she also has several years of experience as an ethnographic researcher and a lecturer. Therefore, the ethnographic work collected was of high quality. The observations carried out by Leo took place over a period of 18 months and ranged from 30 minutes to full days of observations. They were primarily focused on individual scientists, but also, on occasion, group activities, such as lab meetings or more informal social gatherings away from the lab (Sloan *et al.,* 2009). The quality of the data was judged on the depth and breadth of its value to the study. The data was collected through an open interviewing method and avoided any methods that would provide a limited or narrow response such as yes no answers (as highlighted in Hinds *et al.,* 1997). The completeness of the data was based on the following: condition of the data set, accuracy of transcription, comprehensibility of the data, and interpretability of the data (Hinds *et al.,* 1997). These factors were adopted for this research to assess the data. I conducted this work using the information and experience gained from working within the Usable Image project. Again, this helped me address many issues. The assessment was informed by the

reviews of the ethnographic work described in Chapter 3 section 3.7 that were carried out by the entire Usable Image team, including myself. When carrying out the secondary analysis, I could refer to Leo when I needed clarification of any points raised in the data, and I could also refer to the other two researchers of the Usable Image project. Because of this process, no pilot study was conducted to assess the data as reviewing the data and participating in the Usable Image project was deemed sufficient.

- Sensitivity of the research to the context

This is related to the validity of the study. In my role of being the researcher for this thesis, I was able to gain an understanding and suitable control. This understanding came from being situated in both projects (see Chapter 4 sections 4.2 and 4.3). This work provided access to the scientists and scientific software developers, so it implies that my own research role and work were not separated from the data I was analysing. It was critical that I participated the range of Usable Image activities (shown in Figure 4.2), as this would give me a stronger sense of the research context.

- Care of the researchers involved

The care of the researcher is most relevant when the subject of the research is sensitive e.g. This is where the data holds personal information or delicate information such as can be found when working with medical data. This is explained to be important because where in secondary analysis the researcher analysing the data will have not been involved in directly collecting the data. This consequently can leave a researcher with a disconnection to the research context and consequently a lack of sensitivity with the data. There is a recognised limitation to how a researcher may react or be affected by the data. This aspect for my own research was not problematic for two reasons as the data was not sensitive and for my research and I was not isolated from the process of the primary data collection.

- Currency of the data set

This helps to address the phenomena and processes that interact and consequently change over time (Hinds *et al.,* 1992). The question this raised, for the secondary study, is whether the analysis should be done after or parallel to the primary study. This was not feasible as my own analysis as it was running at a later schedule to the work carried out in the Usable Image project. Therefore, I ran my secondary analysis after the primary study. The issue of the currency of the data set was beyond the control of this

work so was not accounted for in this thesis. This is, however, a factor identified to further improve the quality and understanding of the work and it will be discussed in Chapter 9 section 9.3.

4.7 Summary

Chapter 4 has covered the scope and details of the methodology, for which I have used ethnography to collect and analyse my data. The methodology suits my research work in my individual role, and as a member of the Usable Image project and the OME project. This has a bearing on the analysis of the research, as the fieldwork analysed through Chapters 5 and 6 use the fieldwork to situate the perspectives of these projects. The core components for this research work derive from work by Seidel (1998) and his three principles of qualitative data analysis: collecting things, noticing things, and thinking about things. This chapter also documents the factors reviewed and considered for carrying out a secondary analysis of the Usable Image project data. Chapters 5 and 6 will now apply the method that has been described in this chapter. Chapter 5 covers the first phase of the analysis of the Usable Image ethnographic fieldwork (study one).

Chapter 5: Usable Image Fieldwork

This chapter analyses the ethnographic stories carried out by Leo, the ethnographer in the Usable Image project (The full details of and background to the fieldwork are described in Chapter 4; all the ethnographic work presented in Chapter 4 and subsequent coding analysis for the fieldwork is available in Appendix 8). Seven key themes have emerged from this ethnographic work (How they were constructed from the analysis is shown in the visual diagrams in Appendix 6). The seven themes are as follows:

- Working life
- Microscopy
- Tools
- Practices
- Workflow
- Collaboration
- Roles

From these seven themes, I have chosen to explain the first five because these are the ones that have most influenced the analysis. This point is expanded on in the summary of the Usable Image fieldwork analysis in section 5.4.

5.1 Ethnographic analysis

5.1.1 Working life

The theme of 'working life' emerged from the observation and collection of codes of 'dedication', 'alternative career', and 'long hour culture'. Importantly, from the perspective of the fieldwork, this theme generated a sense of empathy to me of the work that the scientists carry out. The empathy for the scientists was represented in the analysis by the codes of the dedication scientists have for their work and in the long hour culture that is part of the working life of a scientist. Extract 1 has been taken from the fieldwork that I feel best exemplifies this position.

Joseph described, 'People are working like slaves.' Indeed, long hours are common: it is not unusual for people to spend a good 10-12 hours on a workday and extra visits to office during weekends. But this doesn't seem unusual – I was told scientists around the globe tend to work long hours. PI like Gottfried and senior staff like Karl also set an example themselves, as Ralf put it: 'Look at Gottfried or Karl – they come early and leave very late and they always come to work during the weekend. So we have nothing to complain about.'
Extract 1 - 070831EthnoObs_StKilda-2

Extract 1 first captures the response of Joseph to Leo about the long working hours scientists put in during the week and how they also visit the office during the weekend. The perspective added at the end of the extract by Ralf indicates how Gottfried and Karl (the laboratory PIs) set an example of working long hours, so the laboratory cannot moan. This reinforced the acceptance of the hard-working culture as Gottfried and Karl are successful PIs, so they are examples to follow.

The 'long hour culture' code was also observed in the two different institutions the ethnography work was carried out in. Extract 2 below is taken from Skye. It captures how a scientist had her plans changed and was working all Saturday afternoon and evening until 10pm. The scientist, Sasha, had to work late because of technical troubleshooting with the microscope. The final implications of Extract 2 meant that Sasha would also be working on Sunday morning to catch up further with her work because of these technical problems with the microscope.

Instead of leaving the building at seven as planned, it was not until after 10pm that we left the building. Sasha would have an early start the next morning – 8am on a Sunday morning. **Extract 2 -** 080620EthnoObs_microscope

A further associated code connected to the theme of working life was the 'dedication' code. Extract 3 describes Lisa, a post-doc in Gottfried laboratory who is working part-time at StKilda. The extract describes the support of Lisa's boss (Gottfried), which she needs as she has a young baby daughter. The extract explains that because of the dedication required in the world of science, it would not be possible for Lisa to take a break. This time away from science would also affect Lisa's career choice as she hoped to become a PI and run her own laboratory. Extract 3 implies that the world of science

is so demanding that it makes it extremely difficult for people in it to have a balanced family life away from the world of science, given the level of dedication it requires. Extract 3 concludes with the view of Leo, which has been included because of her underlining point about the required dedication of scientists. This aspect of the extract has helped to develop the connection for the theme of working life, as concluded in Extract 3 that as a scientist you need a *passion and love of science* in order to work the 'long hour culture' that science can frequently demand.

> *To get Gottfried's support to keep a part-time job was also essential for her to 'stay' in the world of science. If she became a full-time housewife for a few years, it wouldn't be possible for her to come back – she would be 'out of touch'. She felt it was almost impossible for her to be so dedicated to her work for aiming at the professor's position if she wanted to stay long in the university. This pressure confronts all the academic staff. I believe it is passion and love of science that makes people dedicated to pure scientific researchers.*
>
> **Extract 3 -** 070831EthnoObs_StKilda-2

The analysis of this theme also benefited from the ethnographic work that was carried out between two separate institutions. The codes of 'dedication' and 'long hour culture' observed in both institutions gave insights into the wider scientific culture beyond the institution in the UK where the main research was being carried out. In this respect, these two codes became a way to generalise about wider and general scientific culture practices.

5.1.2 Microscopy

A second key theme to emerge was microscopy. The insight into the scientists' background work with a microscope aided understanding of the OMERO software as it revealed the work scientists are involved in with acquiring scientific images. This theme comprised the following codes: 'data storage', 'microscope setup', 'metadata acquisition', 'metadata creation', 'microscope training', 'learning curve', 'microscope maintenance', and 'microscopy'.

This theme also helped to reveal the challenges of working with microscopes e.g. the

procedures required to be followed for setting up a microscope and the regular training required for microscopes to ensure that the correct procedures are followed. These challenges were particularly prominent in the 'microscope setup', ‘learning curve’, and ‘microscope training’ codes.

Extract 4 below shows the 'microscope setup' code. The 'microscope setup' code explored the details and, in many cases, the difficulties they experienced when working with a microscope. The extract details the laboratory meeting and explains that for particular DV microscopes, the upgrade of a new chamber allows it to heat up quicker for the benefit of scientists carrying out live experiments. After the upgrade, the users of this system will have to be aware of this modification, will have to adapt to it, and might require brief extra training to understand how to use this new chamber. This already indicates that the 'microscope setup' code is closely linked to the ‘microscope training’ code, as will be discussed later in this section. From a personal perspective, the code also gave me awareness of the work carried out before the image data was imported into OMERO. This would benefit me later when I worked more closely with the OMERO team. An instance of this involved understanding the term ‘DV’. The term ‘DV’ means ‘DeltaVision’, which is a particular type of ‘widefield’ microscope. The DeltaVision microscope produces DV image file formats. The DV file format can be imported into the OMERO software. Although OMERO supports many other image file formats, the basic example of the DV file helped me to learn about the scientific image file formats process that could then be used in understanding the role of OMERO. In this example I had picked up the particular terms of DV, DeltaVision, and widefield microscope. This significantly built up a background of scientific terminology of the domain, and it became rapidly clear that the whole Usability Image/OMERO project needed to understand the terminology for the scientific context. This aspect of scientific terminology is discussed in detail in the ‘practices’ code in section 5.1.4.

> *For some DV microscopes, a new chamber will be introduced to improve carrying out live experiments (it took long for some old machines, for example, to switch among different temperatures like heating up to 37'C from a much lower degree).*
> **Extract 4** - 080620EthnoObs_Microscope

The ‘microscope training’ code again served to capture and underline the technical

difficulties of working with a microscope. Extract 5 provides an account of the microscope training code. The context of the discussion is with Bruno – a scientist who has recently had the training for the newly upgraded hardware for the microscope. Extract 5 shows that even though Bruno is a regular user of the microscope that has been upgraded, he still requires the training to ensure he works correctly with it. A reason for this is that '*certain things are done differently'*. Extract 5 concludes with Bruno acknowledging the need for the training. The questions of what how the 'microscope training' code is connected with the 'microscope setup' code is explored in detail in Extract 6. This relationship between the two codes is to do with setting up the microscope, where switching on the microscope in the correct order is crucial. A similar conclusion has been made here: when a microscope is upgraded, which is the context of extract 5, a scientist must be aware of any changes to the microscope setup.

> *Although Bruno was one of the regular users of this machine, it was still crucial for him to have the training as 'certain things are done differently'. Besides, Bruno knew about the upgrading and was aware of the necessary training.*
> **Extract 5 –** 080620EthnoObs_microscope

A 'microscope setup' code that demonstrates the overarching problem and difficulties experienced for the 'microscope setup' is shown in the extract below. The extract is a discussion between Sasha (a scientist working with the microscope on a Saturday afternoon) and Leo. However, there is a problem with the microscope's camera. The statement by Leo underlines how turning the switches on in the correct order are crucial. It is this that puts pressure on Sasha to ensure the methodical and correct setup of the microscope. Although the excerpt mentions that the computer next to the microscope and the monitor are turned on, Sasha was unable to turn the microscope camera on based on the places she knew would turn on the microscope. This final sentence in the extract highlights the unspoken aspect and the need for microscopy experience and training. This is given the complexity of working with microscopes, the scientists are challenged if problems arise.

> *However, it turned out that Sasha couldn't get the camera turned on. There is a small cabinet beneath the worktop where all the switches are held. And when we arrived at the microscope room, all were switched off and Sasha was not sure that they were supposed to be off. To switch them on in the right order seemed to*

be crucial. It took Sasha a while to have the screen of the computer on, although she couldn't get the camera on after trying several times to switch things off and on again. Sasha checked all the places over the machine that she could think of.
Extract 6 - 080620EthnoObs_microscope

The 'microscope setup' code does indirectly explore the aspect of microscopy experience and training. This is because a scientist's ability to carry out the necessary microscope setup is dependent on the experience and training they have received. It is critical to allow a scientist to solve the problems that arise. This observation led to the emergence of the related 'learning curve' code, as in the analysis I questioned the difficulties of working with a microscope. Extract 7 underlines this connection and highlights the principle of the 'learning curve' code. The extract is taken from a discussion between Leo and Helen – a scientist who works infrequently with microscopes. Leo characterises this as Helen emphasises the difficulty of working with the microscope system and the learning curve. Given the barrier of the learning curve, the discussion between Leo and Helen later in the 080620EthnoObs_microscope (see Appendix 4) highlights how it would be a good idea to pull together the microscope documentation to help provide a centralised resource to overcome this issue. For the 'learning curve' code this point seems even more valid, given the difficulties that a scientist may encounter. For my analysis, this is linked to the 'microscope setup' code and the problems documented in Extract 6 for turning the microscope cameras on.

It is a very complicated system and once you know how to work from inside out, it is fine. But it is a steep learning curve to reach that point'
Extract 7 - 080620EthnoObs_microscope

A further example of the microscope theme is shown in Extract 8, which was tagged with the 'data storage' code. Extract 8 simply describes the setup Rachel has for saving her scientific image data: what image data is saved and where she saves her image data. In this extract, the terminology used to describe the image data (the 'raw image') is significant for understanding the background to the microscopy work, as this is the original microscopy image data. It is critical to store the raw image data safely for scientists' research work. The 'deconvolved images' are images that have undergone the 'deconvolution' process. This process is a computational method used to reduce out-of-focus fluorescence in three-dimensional microscope image (McNally *et al*., 1999).

The image data (raw and deconvolved images, her Excel spreadsheets) are stored in Zeus, hard drive, DVD.
Extract 8 - 080226EthnoObs_Rachel

The insights into the microscopy process were important for the research for building a more complete view of the microscopy imaging process. These insights have ranged from identifying the challenges for setting up a microscope to the necessary microscope training to challenges that helps one to overcome the difficulties of working with a microscope. This deeper insight and iteration through the data helped to form and to identify the workflow theme discussed in detail in section 5.1.5.

5.1.3 Tools

A third key theme to emerge was 'tools', which covered the range of tools used by the scientists. The range of tools covers both scientific software and physical scientific tools. The theme comprised of the following codes: 'Volocity' (21), 'ImageJ' (10), 'non-scientific software' (9), 'complicated software' (6), 'Excel' (6), 'PubMed' (4), 'lab book' (4), and 'workstation' (2). The three most frequent codes of the theme 'Volocity', 'ImageJ', and 'non-scientific software' are explored below. The two codes of Volocity and ImageJ emerged as a part of the coding process that allowed for the breakdown of the 'image analysis' and 'scientific software' code. The breakdown of the 'image analysis' and 'scientific software' codes are illustrated in Appendix 6. Both these codes were originally part of the tools theme in the initial analysis. Through my experience with the OMERO development team, my familiarity and understanding of other scientific software and image analysis tools developed so I was able to break these original codes down to describe the software tools. A full discussion of how I arrived at the cross comparison between the two field studies is explained in Chapter 6 section 6.6.

The 'Volocity' code captures the use of the Volocity tool. Three scientists in the observations (Ivan, Graeme and Rachel) used the Volocity software. The principal extract of the 'Volocity' code captures how the software was used to measure the image data and subsequently export it as text into Excel. Extract 9 captures this and clarifies

how Ivan uses Volocity for counting his cells (The wider background to Extract 9 is that the ethnographic observation also describes how Volocity supports Ivan's results by allowing him to also count the cells manually). With the results Ivan can then use a separation application of Excel to work with the data.

> *Ivan uses Volocity for the counting. This data can be exported from Volocity as text and then be opened in Excel where he does his numbers to get graphs.*
> **Extract 9** – 20080306EthnoObs_Ivan

The 'ImageJ' code describes the use of the ImageJ tool. In the ethnographic piece 070831EthnoObs_StKilda, Leo describes the feedback for ImageJ across an institution. She describes the role of ImageJ for scientific work in Extract 10 below. This covered image viewing, image manipulation, and drawing regions of interest (ROI) on an image to make measurements.

> *ImageJ is used mostly for initial viewing, basic manipulation such as changing the contrast, cropping, merging the images among several channels (to detect co-localisation), and using ROI tools for getting data for, e.g. fluorescence intensity.*
> **Extract 10** - 070831EthnoObs_StKilda

However, as I worked within the OMERO team I gained a wider picture of the ImageJ project. This was about both the project and the software. The ImageJ project shared similarities with the OMERO project as it was also an open source academic research project. The project was formed in 1997 (Collins, 2007) and has a large online community that has subsequently extended the functionality of ImageJ because the ImageJ software supports plugins. Collins (2007) mentions that there are over 400 plugins and that this figure is continually growing. A screenshot of the basic ImageJ toolbar is displayed in Figure 5.1. The screenshot covers the menu options of image, process, analyse, and plugins. The plugins label is the option where once installed you may access the additional functionality that has been added to ImageJ. The range of plugins for ImageJ can cover tools for basic viewing options to more in-depth microscopy analyses such as 'colocalisation'.

Below, the menu options are the icons that represent the various drawing and image

manipulation tools available in ImageJ.

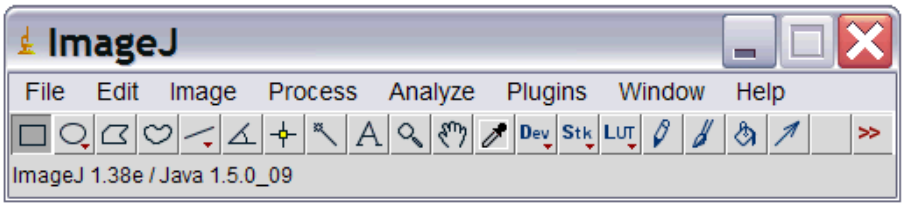

Figure 5.1 ImageJ Main Menu (http://rsbweb.nih.gov/ij/features.html)

Finally for the tools theme, the 'non-scientific software' code has been used to capture the supporting software that was not designed to support the context of science. This code emerged to raise and highlight further questions about what existing software was being used and what the positive and negative experiences of it were made. This aspect is explored in Extract 11 and Extract 12. Extract 11 discusses the Journler software used by Ivan to organise his scientific folders. The benefit Ivan receives from using this setup is in the creation of smart folders, which allows a user to group entries according to inherent properties such as title, category or date. As concluded in Extract 11, Journler will automatically update the contents of the folder if an entry's properties or a smart folder's criteria is changed.

> *Journler' software for a comprehensive organisation. Whenever one changes an entry's properties or a smart folder's criteria, Journler automatically updates the content of the folder to reflect those changes.*
> **Extract 11 –** 20080306EthnoObs_Ivan

Extract 12 further captures the positive and negative experiences of using Journler. Extract 12 explains that the positive feature of the software is its ability to link folders as it allows Ivan to pin related information together. Ivan does this by clicking on the folder links to view the image movie that is associated with the microscopy notes. The drawback of the software though is that it does not support the ability to link to other applications. This problem is explained from Ivan's view in Extract 12 as he is required to import his Excel files into Journler to create a link. What is missing for Ivan in the Journler software is the ability to open applications without having to import the file. This would allow Journler to become a more central platform where the data can be stored which will consequently allow a user (in this case Ivan) to work with their data easily with another application.

> *One useful feature of this software is that the entries among folders can be linked together. For example, one of Ivan's notes of microscopy data is linked to a time-lapsed movie. Clicking this link will bring out the conditions that he noted in another folder. But the software only supports links among it, NOT with any outside link. As a result, he has to import his, for example, Excel files, into the system to make a link.*
> **Extract 12** – 20080306EthnoObs_Ivan

An additional outcome for the 'non-scientific software' code was that helped for the design of OMERO in terms of making appropriate recommendations. The example shown in Extract 12 of promoting the ease of integration with other scientific applications for OMERO was a prime example of this design recommendation.

5.1.4 Practices

A fourth key theme to emerge was 'practices'. This theme covered a wide spectrum of codes, which led to a further iteration of analysis and breakdown of the theme during the analysis. Subsequently, the 'workflow' theme emerged, and this will be described later in section 5.1.5. The codes under the 'practices' theme, with their associated occurrences, are as follows: file management (20), image analysis (14), practice (12), deconvolution (3), file naming (8), scientific discovery (1), visual organisation of data (2), health and safety (2), techniques (3), lab book (4), image centric (3), image viewing (7), new ideas (1), experiment (9), presentation (1), protocols (1), and publication. The three most frequent codes in the 'practices' theme of file management, image analysis, and practice are discussed below.

The code of 'file management' was frequently observed throughout all the ethnographic data. The understanding taken from the code and its frequency was a sense of how difficult a challenge it is for scientists to manage their data files. The overarching point of the code is the organisation of a scientist's data. Extracts 13 and 14 below provide an account of the actions a scientist takes for file management. This code highlighted how a scientist struggles with the amount of data but it also showed how a scientist manages these files.

Extract 13 is a conversation between Leo and Ivan; the latter explains that he uses a structured folder system. When Leo probes further about how well the software available to Ivan helps in terms of managing his data, his response includes 'ok but not so good'. The extract also underlines Ivan's problems and his mood during this with his final remark of 'Always complicated, always'.

> *Leo: How do you organise your data?*
> *Ivan: Make a folder.*
> *(Ivan creates multi-layer folder system to keep his data – many of them are well over ten layers.)*
> *Leo asked how does various software that Ivan uses help with his work,*
> *Ivan answered: 'It is ok, but not so good.... Always complicated, always...'*
> **Extract 13** – 20080306EthnoObs_Ivan

An insight into the type of complications Ivan experienced is given in Extract 14. Leo follows up with a comment of the naming structure of Ivan's files and folders. Ivan says that there is no standard procedure for this that he is aware of and, as the extract explains, file naming is typically being done based on what works for an individual scientist. However, despite this individual file naming, Ivan highlights the problems he is still experiencing with finding his data and indicates that even searching for his data can be difficult because he often forgets how the data he needs to find was named.

> *Leo noticed Ivan's naming system of his files and folders. Ivan said: 'I don't know what is the standard way. I always name them in the suitable way for me...' However, it is still not easy to find what he needs. Ivan said, 'Sometimes I feel like using the "searching" box to search for my data.'*
> *Leo: Does it work?*
> *Ivan: Only if I remember how I named them but I don't always do....*
> **Extract 14** – 20080306EthnoObs_Ivan

The implication of the 'file management' code was especially useful for both the OMERO software and the OMERO development team. This was because the ethnographic observations helped to show how the scientists currently managed their image files and to reveal where there were critical problems (e.g. Extract 14 highlights a

problem with remembering a term a scientist can search for). This could help inform the system design for OMERO because the file management and associated tasks are the core functionality of the OMERO software.

The ‘image analysis’ code emerged as the code to identify the step after the microscope images were acquired and a scientist was examining the data. The code was also identified to be relevant for the development of the OMERO software because the ‘image analysis’ code was a feature that was due to be supported in the software. This point is shown in Extract 15 where Levi is explaining to Leo that the OMERO software will support the ability to draw the regions of interest on an image. Levi describes the regions of interest as the ability to draw shapes on an image. Scientists can then use this function to carry out basic image analysis. This aspect of the analysis is discussed in further detail in Chapter 6 section 6.6 in the cross analysis between the two different fieldworks.

> *Levi explained to me that in future, maybe with 18 months, OMERO will be able to have pretty sophisticated functions to draw different shapes and measure them.*
> **Extract 15** - 070316EthnoObs_Calum

Extract 16 below provides a general account of the role of the ‘image analysis’ code. The extract describes Ivan making a measurement from two parts of a cell, namely the chromosome and the spindle. The context of the extract is a situation where Ivan is talking about his images with Leo and explaining his steps of working with them. The extract reveals what the ‘image analysis’ code represents as a particular phase of work for all scientists. A further point of Extract 16 is the use of particular scientific terminology to describe the measurement for the image analysis. The scientific terminology is discussed in the specific ‘practice’ code and described below. However, for this extract, it is a further example of how, when working in the scientific research domain, the terminology used can be overwhelming. In this instance, the terminology of ‘chromosome to the spindle’ refers to two parts of a cell, when the cell is dividing.

> *One of the measurements that Ivan does, for example, is when does this happen, what’s the distance from this chromosome to the spindle.*
> **Extract 16 -** 20080306EthnoObs_Ivan

The initial perspective of my analysis had focused on the aspects relevant to the Usable Image project and OMERO work (e.g. frequent codes were 'file management' and 'image analysis' – see Appendix 6 for the illustration of the first pass). This first perspective was missing the specific details describing a scientist's work. When looking back over the data, it was always challenging to understand and read, given the type and amount of details in the scientific practice described. To account for this difficulty in the analysis, the 'practice' code was formed to capture the scientific terminology and the details of a scientist's work. Extract 17 below describes the background of the imaging work that Ivan carries out.

The purpose of the extract is to underline the range of complex scientific terminology related to a scientist everyday's work that was found in the ethnography work and that covered the 'practice' code. The key terms used in extract 17 are chromosomes, metaphase, and spindles. The scientific terms are not easily understood without having a scientific background. However, the ethnographic work covered scientific work that was carried out by scientists with a PhD or a qualification at a post-doc level, so their level of biological scientific expertise was vast in comparison to my own. A full understanding of the research work that was being carried out was a challenge but it was one that could not be overlooked in my analysis.

> *Ivan images live yeast cells to observe the transport behaviour of chromosomes. To take one of his images as an example, in this case Ivan creates an artificial condition during the cell metaphase so that one of the chromosomes will stay somewhere distant from the spindle. Usually the chromosomes congregate around the elongated spindle.*
> **Extract 17 -** (20080306EthnoObs_Ivan)

To conclude, this 'practice' code emerged because of the scientific context and served to account for the complexity of the environment. As already stated, a general understanding of the scientist's work was important, even if not complete because of my own lack of expertise in microscopy and biology. This point is exemplified in the work by Chilana *et al.,* (2010), who underline the problems of working in complex domains and the challenge of domain-specific terminology in terms of how it is difficult for UCD work. This challenge was met in my research in three ways: 1) the support of

the range of methods used by the Usable Image project (see Chapter 3 section 3.6) that I was involved in helped inform and build an awareness of the key concepts; 2) the regular contact with the scientists allowed for the formation of a general understanding of the scientific terminology and gave a general overview explanations of the scientific research work; and 3) continually coming back to the fieldwork analysis of the work over a period of time helped to overcome the difficulty of the complex environment. These set of practices were also applicable to when I moved over to the OMERO project, which is examined in Chapter 6.

5.1.5 Workflow

The 'workflow' theme emerged as a combination of several codes and observations of the work the scientists carried out. After the initial analysis of the Usable Image fieldwork shown in Appendix 3, further questioning went on to break down the larger theme of 'practices'. Through the analysis, phases of work were recognised to be central to a scientist's workflow for image acquisition and image analysis process. This awareness was taken from the 'image analysis' code (see section 5.1.4) and the 'workflow' theme was constructed around this code. The codes that emerged which subsequently formed the workflow theme are shown in Figure 5.2a. The linear layout of the codes from left to right represents the steps of work and the order of the process the scientists are involved in. The codes are protocols, experiment, microscope, deconvolution, image viewing, image analysis, Excel, and publication. The theme has been constructed based on the observations taken from the analysis and not the frequency of the individual code. All the codes included in the workflow have formed a broad representation of the imaging process to help understand the workflow of the scientists who work with microscopy. This theme has provided a wider picture of a scientist's work practice, with the added benefit of understanding where OMERO fits in to this process. OMERO is illustrated in Figure 5.2a by the red rectangle that encompasses the two codes of image viewing and image analysis.

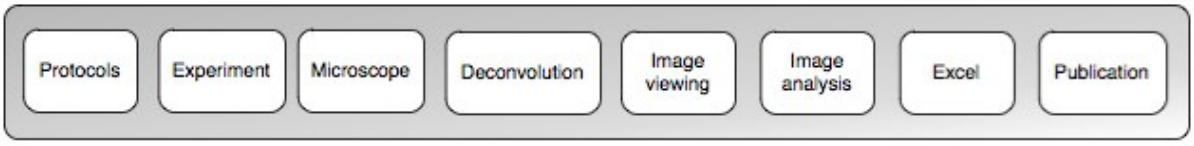

Figure 5.2a Workflow theme

The first two codes ‘protocols’ and ‘experiment’ represent the actual scientific laboratory bench work. Extract 18 presents an example of the ‘experiment’ code. The role of this code was to capture the steps and explanations provided in the analysis of a scientist’s experiment work. In the extract below, Leo has described the goal of Ivan’s experiment and notes that the multiple experiments Ivan is carrying out help to form the answers for his wider research question.

> *The purpose of such experiments is to see the percentage of chromosome loss in normal (called ‘wild type’) vs mutant cells. The cells used here are just for this experiment and won’t be imaged under DV microscope. To say in another way, this is another assay for the research question that Ivan is interested in – the phenotypes of mutant and wild type cells. In general, the mutant cells don’t lose chromosomes as wild type cells do.*
> **Extract 18** - 20080306EthnoObs_Ivan

The third code in the ‘workflow’ theme is ‘microscopy’. This theme was discussed in section 5.1.2 and was placed in the ‘workflow’ theme because it captures the central tool for imaging and the associated codes that emerged within the ‘microscopy’ theme. Section 5.1.2 documents several of the codes from the microscopy theme.

The fourth code of ‘deconvolution’ was a specific term learnt during the analysis process, which documents the use of a computational method to reduce out-of-focus fluorescence in three-dimensional microscope images (see Extract 8 in section 5.1.2). Extract 19 provides further context to the deconvolution process. It is taken from a discussion between Leo and Rachel, a scientist who explains that her laboratory has their own deconvolution software (Volocity). It is taking a very long time to deconvolve her images, so with their own version of the Volocity software her laboratory is not dependant on using the shared workstations where it means sharing Volocity with multiple laboratories in the institution.

> *Newland's lab bought three packages of Volocity: Deconvolution, Visualisation, and Measurement. To have their own licensed Deconvolution will save them time for using the workstations downstairs, which need to be booked in advance – this is especially time-constraining and consuming for Rachel, for example, as*

she images movies and they sometimes need a long time to get deconvolved.
Extract 19 – 080226EthnoObs_Rachel

The fifth code of 'image viewing' covers the software that was used to visualise the images. Extracts 20 and 21 below show Volocity as the visualisation tool for Rachel and Adobe Photoshop as the one for Frank. Extract 20 shows how Rachel uses the X and Y axis in the image viewer to pick up on the spots on the cell. Extract 21 shows the alternative non-scientific software to the Volocity software.

For Rachel, she mainly uses Volocity's visualisation to view her images. And the ability to use X and Y axis to view her images is important as, ultimately, Rachel is looking for the fluorescent spots (nascent RNA) and sometimes they are very weak and therefore viewing them from a different angle will help her to spot them better.
Extract 20 – 080226EthnoObs_Rachel

Frank was viewing his images. He used Adobe Photoshop and on his shelf were a few reference books about Photoshop.
Extract 21 – *270207EthnoObs_Frank*

The sixth code of image analysis represents the phase of work where a scientist would carry out an analysis of their image data. The activity of image analysis can cover drawing regions of interest on an image in order to quantify the intensity of a structure in the cell for example.
For the details of the code please see Extract 16 section 5.1.4.

The seventh code, 'Excel', is used to capture the scientist's use of Excel. It was placed in the workflow theme as the code captured a general tool used in the later stages of a scientist's work. Extract 22 documents how Arthur uses Excel in his presentations. Extract 23 also presents the link of the 'Excel' code to the image analysis stage. Rachel, the scientist, has tracked down the spots in her image; she has marked down a '1' in an Excel spreadsheet and has then added them to another spreadsheet to calculate how many cells are expressing a specific marker at each time point. What was gained from this particular extract was the value of the image analysis software and how closely they need to work with further analysis tools such as Excel, which allows a scientist to do

further analysis. The 'Excel' code highlights that scientists need to perform further image analysis on their data (i.e. for quantifications that are often needed for their publications). This was a very important fact that gave me greater awareness for OMERO on how to integrate image analysis tools and what output file format to use.

> *Arthur uses Excel for his numbers and graphs and Illustration for presentation which works well to handle graphs*.
> **Extract 22 -** 080306EthnoObs_Arthur

> *After all the counting is done, she will input this information to an Excel file*.
> **Extract 23 -** 080226EthnoObs_Rachel

A weakness of the 'workflow' theme is that it represents an abstracted view of the ethnography work so the rich picture built through the fieldwork is lost. It is difficult and problematic to convey the move through each of the codes in the workflow without having been fully immersed in the data. Because of this, I have reviewed the existing literature to seek similar concepts and principles, to both situate the theme against prevalent concepts and examine my own proposal of the theme for my analysis.

Being aware of this, I went back to one of the scientists interviewed in the ethnography study to question how my interpretation of their work holds up. I spoke with Sasha and explained that I wished to have her opinion on the workflow theme. I presented my coding process and the full list of codes created from the analysis (as shown in the diagrams in Appendix 6). I conveyed how I had arrived at the code terms and then selected the codes that I felt worked to capture the image acquisition and image analysis process. Here are her comments:

> *"The different steps in the workflow diagram seem correct to me. All the scientists using microscopy for an experiment will have to go through those steps, except in some cases the deconvolution step. This step is a bit more specialised. But I would still leave it in your workflow diagram. For the Excel step, I think this one is part of the Image Analysis step. And as a general thing, I would actually change the layout of the diagram from this rectangle box to an arrow shape, so it shows the progression in the workflow."*
> **Extract 24 -** Sasha 2010

The revised figure that has accounted for the comments made in Extract 24 is illustrated in Figure 5.2b.

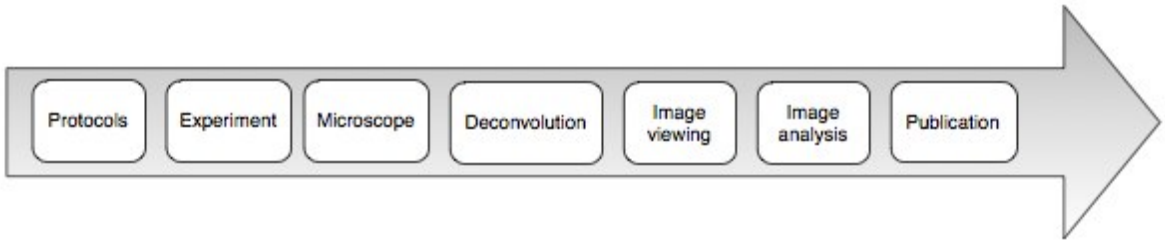

Figure 5.2b Revised workflow theme

Before going on to discuss the summary of the analysis I shall examine two final sections of work for the Usable Image fieldwork. First, a short examination of existing scientific ethnographic work to help further validate my own analysis. Secondly, the findings from my time working within the Usable Image project. The final summary of the Usable Image analysis is in section 5.6.

5.2 Existing scientific laboratory studies

Latour and Woolgar (1979) and Latour (1987) have analysed various science-based studies. The work by Latour (1987) examined how science and technology must be studied "in action". Because of the difficulties of understanding science and technology, it is argued that it must be studied where the discoveries are made. With reference to my own research, this reinforces the use of ethnography for the research. Latour (1987) also discusses how scientists work in a ‘black box’. This is a metaphor created by Latour to define how a scientist draws a black box around parts of their day-to-day complex work such as the methods and techniques that they use. The inner details of the complexities of the black box are then not required to be remembered by the scientist. The question of complexity for the scientist is reduced to how the work needs to be done and how it is used in their everyday activities. In the context of my own research, this can be linked with how the development of digital microscopy imaging has evolved to make the image acquisition easier so that scientists don’t have to know all the theoretical principles of optics to use a microscope.

The earlier work by Latour and Woolgar in the book Laboratory Life (Latour &

Woolgar, 1986) examines the scientific practice of neuro-endocrinology at the Salk Institute. The work constructs a picture of interconnected practices that form part of laboratory life. A model constructed in the work is the cycle of credibility, shown in Figure 5.3. The cycle of credibility demonstrates the continual cycle in which a scientist works. The beginning of the cycle highlights the need to work to find a grant. After receiving money, the time comes to buy the scientific equipment, to acquire the data, to form a scientific argument, to publish articles, and to gain scientific credibility. Then, the cycle starts over again. By this aspect of credibility, Latour (1979) underlines scientists' motivation. The cycle of credibility represents the conversation between the capital required for scientists to progress within the scientific field with the benefits of the information. Latour and Woolgar (1986) highlight "*The cost-benefit analysis applies to the type of inscription devices to be employed, the career of scientists concerned, the decisions taken by funding agencies, as well as to the nature of the data, the form of paper, the type of journal and to the readers possible objections*." (Latour & Woolgar, 1986, p. 238). The cycle of credibility is a key point for the Usable Image ethnographic work. The work presented by Latour (1979) is critical for two aspects of:

1) In combination with the fieldwork analysis, it has helped in developing a level of empathy for understanding scientific practices (the role of empathy is discussed in further detail in section 5.3.2).
2) The diagram (shown in Figure 5.3a) has helped to further ground my own understanding of the analysis of the scientific work and the principal goal of credibility within the scientific community that drives them. The cycle of credibility also supports the ability to put the ethnographic work into perspective as it can take place over a period of years, depending on the length of funding received for the scientific work. The publication is the culmination of the work's central results, which a scientist will be working towards over a certain period – omitting a large portion of the data amassed throughout this period.

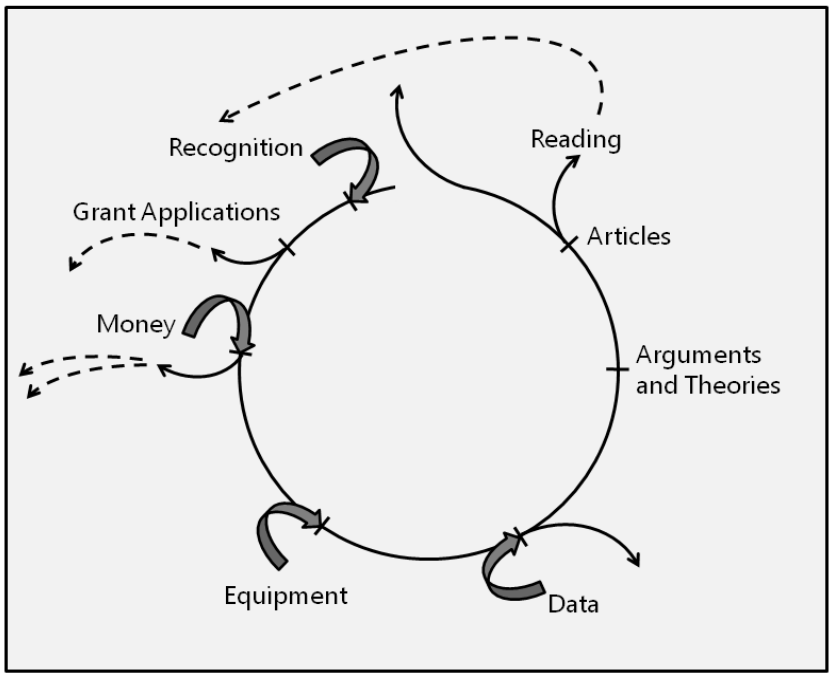

Figure 5.3a Cycle of Credibility (adapted from Latour, 1979)

This understanding from the cycle of credibility was used in combination with the workflow theme (as discussed in section 4.1.5) to build up a further level of understanding of scientific practice. My own 'workflow' theme was emerging from my data as I sought to explain the steps scientist take when working with image acquisition and image analysis process. The cycle of credibility could situate the scientific working process on a more generic scale, as the cycle of credibility provides a wider over overview of the steps of the scientific process. From the start (funding) to the end (publishing), a scientist needs academic credibility. From my own analysis, although with a much smaller scale of data and linear set of actions, the 'workflow' theme could be positioned within the cycle of credibility as is shown in Figure 5.3b. The 'workflow' theme was formed/focused on the actions of scientists who work with a microscope. Thus, any further comparison to the wider context of the cycle of credibility has been limited.

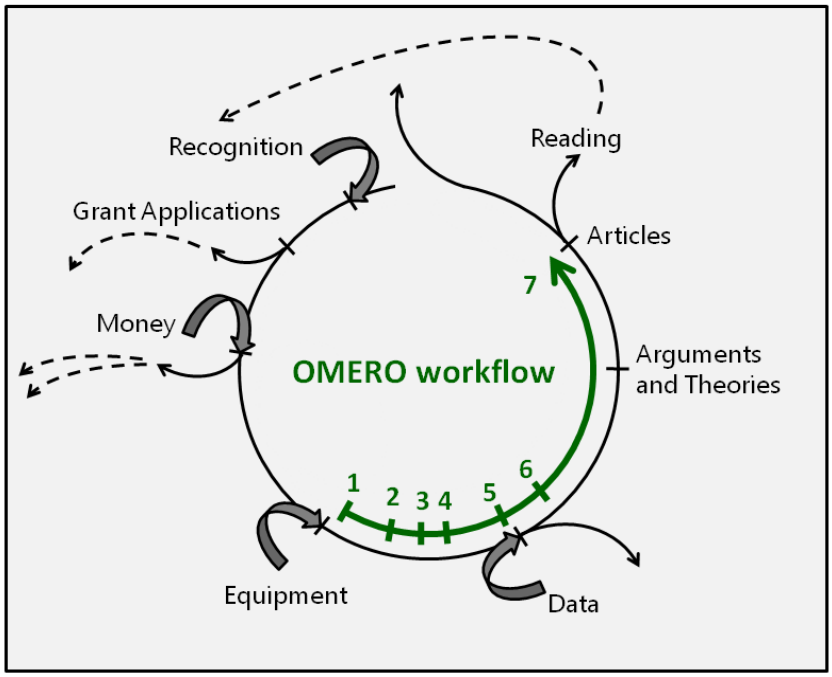

Figure 5.3b Cycle of Credibility with OMERO workflow

(Steps of the workflow: 1, Protocols; 2, Experiment; 3, Microscope; 4, Deconvolution; 5, Image viewing; 6, Image analysis; 7, Publication. Adapted from Latour, 1979)

More recent work by Shankar (2004; 2006) has examined an academic animal neuroscience laboratory. Shankar (2006) focuses on the requirement for re-thinking the area of scientific knowledge management. This work investigates the future development of technologies for scientific work involving the creation, sharing, and managing of scientific data so that it may be more purposeful. The ethnographic contribution of the work is in the awareness of record keeping for scientific work.

One particularly significant aspect described in Shankar's (2006) work was that she had previously studied and worked in a laboratory environment (Shankar, 2004) and that she came into the ethnographic work with an undergraduate degree in molecular biology along with the first part of a graduate degree in biophysics. Therefore, she explains that the scientists could explain certain concepts with her being able to understand without extensive explanation. This point was picked up on and seen to be relevant to my own analysis as the 'practice' theme identified aspects of the ethnography where reading and understanding the scientific terminology was challenging for me as a researcher without any scientific background. This point was significant for my research as my own analysis encountered the challenge of scientific terminology in the 'practice' code (see section 5.1.4). So contrasting my work with Shankar (2004), she had the vantage point of an understanding of the complex

terminology within the scientific domain. This allowed her to enter the field as a research closer to the position of a native scientist, whereas my own basic learning of the scientific terminology happened over a longer period of time. Therefore, the work by Shankar (2004) contrasts with my own research as the basics of the scientific domain and terminology were acquired through time in the OMERO project.

Star and Ruhleder's (1996) work studies a dispersed virtual laboratory system, the Worm Community System (WCS), to link the work of over 1400 biologists. It was both unique and complex because the WCS was able to function in three different ways for different groups: 1) as a set of digital publishing tools; 2) as a tool for problem solving and information sharing; and 3) as already established infrastructural laboratory tools. The work by Star and Ruhleder (1996) was important for my own research as it covered the permutations of how the scientific software functioned in several different ways for the three groups. The relevance for my research is described in Chapter 6 section – where the OMERO scientific software development moves between the development focus of functionality and infrastructure of the OMERO software. The similarities made have been in how a scientific software development team may manage moving between different areas and groups for the scientific software development process.

5.3 Supplementary fieldwork

Along with my analysis of the fieldwork, my work and experiences within the OMERO project and the Usable Image project also provided additional insights for the research. The supplementary information has been categorised into two aspects: 1) the co-location benefits examines the advantages of working alongside the OMERO SSD team; 2) the implementation of UCD in OMERO project via the Usable Image project.

1. Co-location benefits
2. Contribution of UCD in the Usable Image project

5.3.1 Co-location benefits

In the OMERO team, several of the Dundee-based software developers have been working in a scientific context from the very beginning of the project. I questioned this specific issue of SSD embedded working in the OMERO project because of the

previously examined work by De Roure and Goble (2009), and the six principles of designing for adoption which had emerged from the observation of their own team for the myExperiment project (See Chapter 2 section 2.7). The project run by De Roure and Goble (2009) operated with two core developers, with a larger team around them providing occasional support for specific research communities.

The co-location for the OMERO project gave the OMERO developers an insight into the microscopy and scientific work. This developed over a long period throughout the OMERO project. It has also helped to support the relationship between the OMERO software developers and the scientists. The co-location was brought about because of the OMERO scientific software developers being in the same building and, in some instances, the same office as the scientists. The co-location has also brought about a social aspect that has been discussed in Chapter 3, which included coffee breaks and attendance at science talks and outreach events. This has ultimately meant that the OMERO development team were not cut off from the end users of the software.

There was, over time, variation in the developer co-location with the scientists in the Wellcome Trust building. This was a reflection of the OMERO project's growth and, also, the acknowledgement of the requirement to accommodate further disciplines within a scientific institution. Figure 5.4 below highlights the levels of co-location over time that the OMERO development team have been exposed to. The front view of the Wellcome Trust building highlights the scientists' offices in orange and the software developers' offices in green. The left panel of the figure shows the initial office co-localisation (the offices having both scientists and developers are orange with a surrounding green box), and later on, the building co-localisation (right panel).

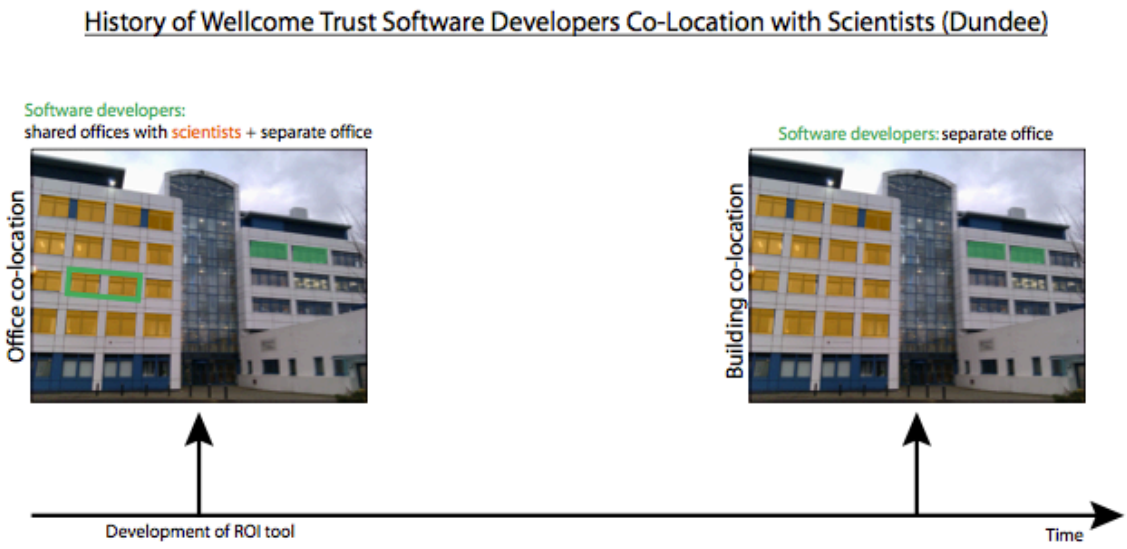

Figure 5.4 Timeline of Co-location through OMERO

Co-location can have a significant benefit for the work between the scientific software developer and scientist. One of the developers (Levi) was based in an office with three other scientists to conduct the early development work of the OMERO analysis tool.

However, in discussions with the OMERO team, it emerged that a significant scale and period of the software developers being directly embedded may bring unwanted consequences to the software project itself. Although there are potential immediate starting benefits of beginning the development process embedded with the user community, the OMERO team highlighted that, with a period of sustained co-location, the software development can go astray with the requests of the local scientists and so lose sight of the wider scientific community. A consequence from this observation is the proposal of simply moving the developers to be embedded with the scientist in the way highlighted by De Roure and Goble (2009), who propose embedding developers with users and users with developers side by side for long periods of time. Otherwise, there could be negative side effects from this, which could cause a level of divergence within an SSD project. In extract 25, Yvan confirmed that:

> "*The early development of the feature was benefiting from the fact that the scientists and developer were co-located. However, over a longer period, it did*

begin to have a negative effect on the development of the feature. In part this was due to some of the technical foundations as well as the influence of the co-located scientists on the developer and the ability to request improvements only targeting their needs. This can be noticed in the current status of the feature and is part of the underlying requirement and need to re-develop the feature."

Extract 25 - Yvan 2010

The value of having the developers co-located in the same building and having access to the scientists was observed through fieldwork. A further advantage for developer-scientist interactions is the immediate feedback on small bugs/requirements. The extract below discusses suggestions from Callum, a scientist, about his problems with the OMERO measurement tool. Extract 26 deals with how he expects to view the image in OMERO, which is as they were captured from the microscope. Callum emphasises his point by his final statement: *'Stop fiddling with images before we have a chance to see them!'* In Extract 27, Steve and Yvan had direct communication with Becky (another scientist). Like Callum, she tries to explain to the developers that the settings used on the microscopy to capture the images should not be changed when being imported to OMERO. It is this direct feedback from working in a co-located environment that has exposed the OMERO developers to the scientific views and challenges of the users of the OMERO software. This would reflect the advantage described by De Roure and Goble (2009) of the benefits of the first-hand experience of working in the scientific work environment.

Callum requested to stop the default scaling. When images are imported, Yvan put on a default scaling intensity, 'which is far too high'. Both thought 'it should come as it is', and 'images should not come as upside down' (flipped on the horizontal axis). Images should be the way that they were captured in the DV. This principle should apply to flipping images and intensity. Images should be imported as they were in DV. 'Stop fiddling with images before we have a chance to see them!'

Extract 26 – 070605-1UserObs notes

Scientists want their raw images in the original formats. The things that Steve and Yvan apply might make sense to them, but not to scientists. They mentioned that they told Miles about this before. Becky argued that they had a reason to

make the setting on the microscopy to capture the images and they shouldn't be messed around after being imported to OMERO. The fiddling decision should be in the hands of scientists. **Extract 27 - 070605-1UserObs notes**

The interactions between scientists and software developers may only be a small portion of the potential scientists' user base for the OMERO software, but the interaction of the OMERO software development team has supported the understanding of the development of the OMERO software. Through the teams interactions and communication with scientists, the developers have been provided with real-life work practice examples to draw on. The physical co-location between the scientists and the scientific software developers gave instantaneous feedback to the scientific software developers and it helped them to gain direct insight into the scientists' perspective. Through the range of channels for user feedback, either face to face or the UCD methods of the UI project, the OMERO team could understand why the software did not fit the workflow or function as expected. Extract 28 below is an example of this where Levi and Yvan are working with Becky, a scientist who has lost information in the measurement tool provided by the OMERO software. She explains the situation to those who are trying to help diagnose her problem.

Becky lost her lines while fiddling with things. Levi suggested to close and reopen measurement tools and the lines didn't come back. Becky defended that this couldn't be her machine as she shut almost everything from the screen and she admitted she was responsible in the past as sometimes she had 100 windows open. Levi and Yvan reckoned this had something to do with 'zoom to fit the window'.

Extract 28 - 070605-1UserObs notes

A further example of the advantage of co-location and working collaboration was in the additional expertise and help a developer provided. Extract 29 below explains the context of how Levi the OMERO developer ended up working alongside Callum – the scientist in the same office. This extract was taken from my analysis under the code of 'scientist developer co-location'.

There was a reason for Levi, one of the developers, to sit with Callum, a scientist, apart from the limited space within the two small developers' offices

upstairs. When Anthony joined the team to take over some of Yvan's duties, it seemed a good idea for him and Yvan to sit together and Levi to come to the lab.
Extract 29 - 070316EthnoObs_Callum

The observation goes on to explain that Levi was also helping Callum in his work in terms of using the existing image analysis tool called CellProfiler. This is described in Extract 30. It was this expertise regarding the CellProfiler tool that allowed Callum to do simple programming.

Levi had been helping Callum with imaging measurement using the software package 'CellProfiler'. Levi's role in the team was developing the analysis (measurement) ability of OMERO. With Levi's help, Callum was able to have a go with 'CellProfiler' – the parts that were useful for him. He could do some simple programming.
Extract 30 - 070316EthnoObs_Callum

The benefits of co-location and working collaboration for UCD have been the positive feedback that the user observations have provided for the UI design. The outcomes of UCD aid the continued evolution of the design, the development of the user interface and the establishment of a clear workflow through the system.

However, along with the advantages of being co-located with the scientists, the co-location can also bring unfavourable consequences for the context of the SSD environment. With co-location comes raised expectations of local users – as highlighted by De Roure and Goble (2009) (see Chapter 2 section 2.7) the provision of "favours will favour you" may help the project development team but also bring about compromises for the development of the software. This recommendation is met with some trepidation because of the nature of SSD projects and the scope of permutations this has on how the users may or may not be managed. The design principle of managing favours through the time and growth of a project can pose more questions if not carefully managed against the evolving size and scale of the project.

As previously outlined by Hine (2006) in Chapter 2, the success of the software can be made more complex in the deployment of the software, as a successful deployment can depend on an individual scientist, a laboratory, or a project. It is this 'tragedy of the

commons' as referred to by Segal (2009) that impacts on the requirements of the software both in agreements and priorities. Because of these implications of the deployment and uptake of the software, the value of being co-located for the research offered a unique insight into the scope of the issues such as in the additional expertise and help a scientific software developer may provide and benefit for scientists. In contrast to this, there is a recognised disadvantage, as the user community will not have a single voice, so this sets out further complications for an academic SSD project (i.e. one laboratory may want features A, B, and C, yet scientists in another laboratory may request features X, Y, and Z and complain the most about directing a project towards their own scientific agenda).

In creating a co-located environment for software developers, extensive consideration of the local influences that may direct the project team must be given. There is a fundamental requirement to balance the information gained from the local context with the information of the wider community. This was demonstrated through a prolonged co-location between the scientists and software developer within the office level, as previously shown in Figure 5.4. The development of the analysis features for the OMERO software throughout this period highlighted the wider implications of this. Through the co-location of the software developer and scientists in an office within a laboratory, the scientific software developer benefited from easy access from the initial proof of concept phase of development. In my discussions with the OMERO software developers directly involved in the development process, it became clear that the development work and process in question was built upon a proof of concept that grew without a suitable technical foundation. This was, due to the co-location of the developers and the scientists. However as depicted in Extract 25, problems were also experienced by the local users, who themselves became 'local community champions' (a term coined by De Roure and Goble (2009)) of the OMERO software. These are the local advocates and day-to-day users of the software. Through the prolonged exposure of the OMERO software and involvement in UCD activities scientists can begin to expect the software to be of benefit to their own scientific practice. With this understanding, it reinforces the requirement for understanding users when making software changes, as highlighted by Kensing and Blomberg (1998).

This work practice of scientists may be referenced back to the concept of the 'black box' working as defined by Latour (1987) and as highlighted earlier in this Chapter

section 5.2. This is not a personal criticism of scientists as the majority appreciate scientific software developers' time constraints. However, scientists' concerns do not always consider the bigger picture of the software development project or the priorities and deadlines; instead, they focus on their own scientific work and deadlines.

The potential problem this presents for an SSD project is an expectation to always help fix scientists' problems with the software. This may then translate to scientists' refusal to participate in any potential further involvement in the software development project. This issue is also significant when the UCD-led work is separate from the software development team. The direct contact with the scientists, through the range of usability studies I as the researcher have been involved in, provides an ideal opportunity to communicate on behalf of and for the benefit of the software developer.

The two-way relationship between the project team and its scientific users requires a high level of trust to be built and continually consolidated for the project's growth. This, in turn, can affect and provide a misrepresentation of the SSD project as a whole, which has to consider and satisfy both local users and the wider scientific community. Further longer-term questions are considered in relation to the way incremental development and incremental content provide an immediate return for those providing the feedback. The approach and practices of incremental development in the SSD context has previously been discussed by De Roure and Goble (2009) in Chapter 2 section 2.8. One of the practices they outline promotes incremental development to establish incremental content. This practice was so that the scientists get core functionality and the software has quicker user uptake of the software. Further work by De Roure *et al.,* (2010) describes their type of scientific software development to be for an Agile web site so the project may afford this type of software development approach.

The longer-term evolution and sustainability of a type of project like OMERO (an enterprise level, academic SSD project) may be more problematic. It would need to be able to meet the demands of the software development team regarding the sustainability for this type of process. A specific concern that has arisen from the fieldwork in the OMERO project is the distinction in the development between the software features and infrastructure. The OMERO software is dependent on a client-server setup, and, consequently, a sufficient level of development work must cover enough for the technical infrastructure. This can present a major challenge in presenting periods of

time where end users of the software can be at a lower level of priority in the software development process. The nature of incremental development and incremental content does not lend itself to infrastructure development as the end users are not seeing an obvious benefit or return on the work for themselves. Because of this the software development process requires a level of flexibility to allow for shifts in priorities, when necessary. The questions being asked of the UCD process in this context require a similar degree of flexibility and adaptability to the questions that may arise from the needs of the software development process. There exists a key balance between keeping local users contented yet ensuring that they do not take away the project's focus for the community goals and the uptake of the software. This can be critical for the success of an academic SSD project. This process of working between the software features and infrastructure has been identified to be dependent upon managing the constraints through the uptake of the software with local users, against the management for the wider uptake of the software in the community. The central problem that was appreciated through this work was to ensure that the scientific software does not become a best spoke solution for the local users.

5.3.2 The contribution of UCD to the Usable Image project

Applying UCD to the scientific context has demonstrated immediate and short-term problems that scientists encounter with their workflow when using the OMERO software. The effects and value of user involvement in this work correspond to other work by De Roure and Goble (2009), Thew *et al.,* (2009), Letondal (2005) (See Chapter 2 section 2.9 for the wider discussion of these). The following extract, taken from a user observations conducted within the Usable Image project, demonstrates the type of insight gained from the UCD work. Extract 31 discusses the feedback from Jody, a scientist working with OMERO, and it shows the frustration she has when the OMERO login screens covered up her screen space.

> *Jody was annoyed at the modality of the Importer and Insight login and splash screen windows, which stayed on top of all other windows*.
> **Extract 31 -** OMERO USETEST 160507

The OMERO usability testing (as shown in Chapter 4 Figure 4.1) created a feedback

loop for the OMERO development team in the form of the Usable Image weekly evaluation cycle steps of: scientists' observations of the software, and a short turn around on issues/feedback gained from scientists' use of the software. This has demonstrated isolated usability problems at the Mesoscopic level. But as argued by Spinuzzi (2003), when design focuses on problems on a particular level, the problems on the outstanding levels of Macroscopic and Microscope are related to the layer under investigation. Thus, if the focus for UCD falls on a level that does not take into account the outstanding levels, the design cannot appreciate the full scale of the problem. This issue also relates with the previously examined work in Chapter 2 by Battle (2005) and the pattern he describes of a 'foot in the door for an external usability group'. The pattern describes how UCD is introduced into a software development group via an external UCD consultant role. His work recognises the limitation of this pattern because UCD is only used in the latter phase of the software development cycle.

A brief assessment of the UCD-led methods applied within the context of the Usable Image project is presented in Table 5.1. The range of methods used in the ethnographic work has mostly helped to inform the research work and to raise user awareness about OMERO. However, as highlighted in my embedded work within the Usable Image project, one of the central problems with Leo's ethnographic observations was with the translation of the ethnographic observations into the OMERO development process. This problem of the implications of design for ethnographic work has already been cited as a challenge of ethnographic work in systems design (Hughes *et al.,* 1995).

Table 5.1 Summary of Use of Usable Image Methods

Name of UCD Method/Technique	**Experience Practical Positive Use**	**Experience Practical Drawbacks**
Ethnographic Observations/User Observation	Insights and understanding were gained from the everyday practice of the scientists.	Bridging the findings into the software development process was a fundamental challenge.
Usability Test	Helped to support the release testing, helped to remove any major recognised usability issues.	Was unable to directly affect usability early on in the design process.
Group Taster Session	Helped to expose a larger group of users to the software and helped to encourage questions.	Required a substantial amount of organisation.
Individual Getting Started Session for a Scientist	Provided a further opportunity to obtain individual insight into how scientists used the software.	Highlighted the demand on resources away from UCD-led work and on promotional-led activities.
Focus Group	Helped to provide a diverse set of	Required a substantial amount

	feedback in one session about OMERO and produced a lot of information to analyse.	of organisation with scientists.
Design Workshop	Provided a platform to generate different and other possible ideas for the design and development process from within the Usable Image Project.	Practically it was not easy to make the transition of information from the design workshop into the development process.
Requirement Gathering Meeting	Helped to form and gain a stronger insight into the project goals.	Highlighted the gaps between the technical context of the SSD project and UCD.
Other Discussion with Scientists	Helped in gaining insights from outside of the local immediate community.	Information and findings from such discussions were not always applicable to the context of development for the project.
Discussion with Developers	Allowed direct communication with the OMERO developers.	Presence of an underlying language barrier, which was overcome over time.
Survey	Helped to reach a wider audience and to provide insight into different scientific institutions.	Required intensive planning with the available resources.
Interface Review	Helped to identify major issues before the software was put in front of scientists.	Was not able to directly affect the design process early on.

The methods of the Group Taster Session and Individual Getting Started Session for a Scientist demonstrated the value and necessity of promoting the uptake of the software. The work by Sloan *et al.,* (2009) documents the Usable Image survey as a method for promoting the OMERO software. The survey provided a way of forming potential contacts to introduce OMERO to new research institutions. This was critical to the Usable Image project as the promotion of the software is a key element for further uptake of it. For the Usable Image project, this was significant in terms of how the academic context recognises the requirement for promoting the academic software. The recently formed group The Software Sustainability Institute (Software Sustainability Institute 2011 is such an example of this. This group provides information on the technicalities of scientific software development and the role of managing open source software.

5.4 Summary

This section will review the insights from the analysis of the ethnographic work and from the supplementary fieldwork in the Usable Image project. From my own analysis, the three key aspects to be taken from the work are as follows:

1) The construction of a 'workflow' theme
2) The emergence of an understanding and empathy for a scientist's research work
3) The influence of my analysis on the system design process

The first point relates to the significance of the 'workflow theme'. This was uncovered in the work when compared with existing scientific ethnographic work and is discussed in conjunction with what was learnt from my own analysis. The second point from the analysis concerns the role of empathy towards the scientists. As discussed by Wright and McCarthy (2008), ethnographic studies have shown the emergence of empathy towards those observed in the ethnographic work. A couple of themes helped to form this empathic view: 'working life' and 'microscopy'. These two themes complemented each other as on the one hand they uncovered how the scientific culture demands that scientists work long hours and are dedicated to their work (see section 5.1.1 for the codes and extracts that cover this). On the other hand, it also revealed the complex nature of the scientific work, particularly how working with such technology as microscopes did not always go according to plan and that the complexities of the microscopes can make scientists' work complex (see section 5.1.2 for the code and extracts that cover this). The third point was that the analysis was influenced by the goals through the system design process for the Usable Image project. This was because I was so embedded within the Usable Image project, my analysis was influenced by the implications of design in the software. Such an example where this was present in my analysis was in the emergence of the themes 'tools' and 'workflow', and the particular frequency of the code 'file management', which was used 20 times (see sections 5.1.3, 5.1.5, and 5.1.4 respectively). This code was relevant for the OMERO software as it is an image file management tool (please refer back to extracts 13 and 14 for an explanation of the code). The emergence of the 'workflow' theme served as a direct way for me, the researcher, to understand where OMERO fitted into a scientist's workflow. When examining my analysis against the existing ethnographic studies, the work by Latour (1979) and his cycle of credibility provided the strongest guidance. The

cycle of credibility provided an overview of a general model of the scientific process, from the steps of scientific funding to the publication of the scientific research, where I was able to position the 7 steps my own 'workflow' theme (see figure 5.3b).

The fieldwork analysis further benefited from the supplementary fieldwork contribution of in the Usable Image Project. The observation of the co-location between the scientists and OMERO software developers created insight into the context. This ranged from developing an understanding of the use of the OMERO software and understanding where the UCD problems lie. In terms of the research, this helped to elevate the understanding further to hold a level of empathy with the scientists and also increase the awareness of a UCD process. It was the combination of these aspects and asking then what was missing from the analysis of the Usable Image fieldwork that led me to identify that the missing point of my analysis was the working practices of the OMERO software developers.

The next step for my research involves taking an embedded role within the OMERO team. This requires the research to shift perspectives from a sole UCD perspective within the Usable Image Team to the scientific software development team of OMERO. This is to obtain a more holistic understanding of the challenges that UCD is associated with. Such a transition of work between UCD teams to software development teams has not been widely seen in the existing literature. The research acknowledges the existing work by Boivie *et al.,* (2006), which evaluates the single role of a usability designer in two professional systems development organisations. The similarity observed was their project aim of bringing improvements to the systems development process with an increase focus of UCD. However, the research by Boivie *et al*.,(2006) as a direct comparison to this research does not examine the type of transition of the usability designer from within a usability team to the inside of a software-development team.

To conclude, the fieldwork and analysis from Chapter 5 has raised further questions regarding the contribution and insight that may be gained from the scientific software development process. Chapter 6 now moves on to discuss this shifting account from the perspective of an embedded member of the OMERO software development team and it analyses the OMERO development fieldwork data.

Chapter 6: Scientific Software Fieldwork

As mentioned in Chapter 4, this research adopted different perspectives: Chapter 5 presented the UI perspective and Chapter 6 will now present the OMERO perspective. Chapter 6 covers five parts. The first part explores the repositioned role of the research. The second part gives the background of a typical OMERO week of work, while the third part discusses the analysis of the OMERO fieldwork. The fourth part then discusses the findings from the analysis of the OMERO fieldwork. Finally, the fifth part concludes with a discussion of the proposal for a change and shifts for integrating UCD into the SSD process.

6.1 Repositioned role of the research

In the exposure to and the deconstruction of the scientific practice, this research work has begun to reposition the role of UCD in the software development process. Figure 6.1 presents my move to obtain a more holistic view of the research.

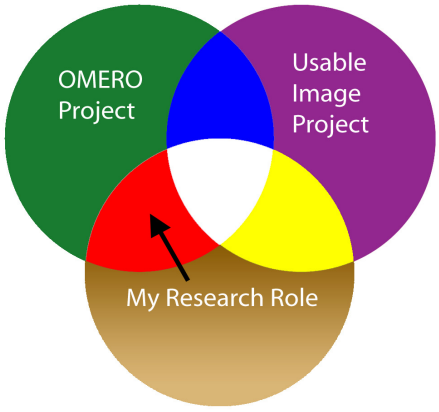

Figure 6.1: The move to observe the OMERO team

The insights from the analysis of the Usable Image ethnography work have highlighted two key issues. First, within the context of the OMERO project, the Usable Image UCD work questions the work culture of scientific practice. It also creates a frame for a level of empathy with and rapport for the scientists' work. This, in turn, for the Usable Image research, opened a channel to support specific system requirements between the scientists and the OMERO developers, which is something that is further explored in the work by Macaulay *et al.,* (2009).

Secondly, it has helped to provide an understanding of why the sole perspective of UCD only completes part of the picture for the development process. Early on I acknowledged the established problems of integrating ethnographic work into the systems development process (Schmidt, 2000, p. 141). This also applies to the previously established list of integration challenges discussed in Chapter 2 section 2.6, but all these have helped me to interrogate my own research questions because of how they have exposed the core and established challenges of integrating UCD and software development. This revealed that I had little understanding of the actual SSD process in relation to the UCD work that was being carried out in the scientific software development process. This led me to identify the area that is not questioned in my research work: the examination of how SSD is undertaken in academic contexts. Figure 6.2 illustrates this, identifying the shift of focus to the new perspective from the Usable Image project. Based on this gap of understanding of the SSD process, for the next phase of the research, the focus shall turn to the SSD practices of the OMERO team. I shall use the opportunity of being embedded within the software development team to understand the functioning practices of a SSD team. The following section now analyses this second perspective of a SSD team and its development process.

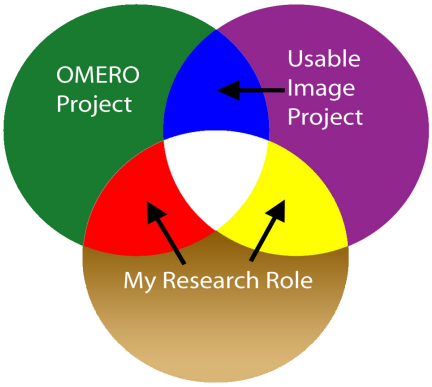

Figure 6.2: Perspectives through the Projects

6.2 The OMERO working week

The OMERO team's week traditionally begins with coffee and cakes on Monday morning at 10:30am. This event was originally organised by Miles through the GRE Wellcome Trust and brings together nine of the laboratories within the Wellcome Trust

building. PhD students, postdoctoral researchers, and PIs from the laboratories all attend when possible. The OMERO team chats informally while drinking coffee, and its member's generally find out what each other have been up to over the weekend. This can be anything from Levi's long walks through the Scottish countryside to the rugby matches Yvan has refereed over the weekend. The OMERO team interacts with many of the scientists that attend; the team is particularly friendly and know many of the scientists from Miles laboratory. The OMERO team therefore chats about diverse subjects with the scientists; these range from new movie releases to computer games.

Outside the Monday coffee-morning meetings, the OMERO team has coffee breaks regularly at around 10:30am and 3:30pm during the week. Jack, Steve, Levi, Cathy, Kurt, and Bob all regularly go up to the Wellcome Trust cafeteria with several of the scientists such as Sasha, Callum, and Finn from Miles laboratory. There is always lots of laughter and humour among them during these breaks. One topic getting a regular laugh is the range of cultural differences between the team members, from how words are being pronounced by Steve in his Canadian accent, to how Sasha understands English with her French background.

The day-to-day working practice of the OMERO team is dominated by the use of the Instant Messenger (IM) dev chat tool. I was able to experience and truly become part of the OMERO team through the use of this tool. Socially, the IM dev chat tool has been used to alert the team about one of the team's members becoming a father and of children being unwell, and it is used for sharing jokes. The IM dev chat tool, on reflection, has played a critical role of information that is exchanged within the OMERO team, but also in helping me to become an integrated member of the OMERO team. Having the opportunity to use the IM tool opened up a further channel for support and insight for me from the OMERO developers. It helped to create a bigger picture of the what, why, and how of the UCD process in relation to the scientific software development work. This was made possible through the supporting information that is available – through such things as the IM tool or Skype. Table 6.1 summarises the range of tools that I used with the OMERO team on a daily and weekly basis.

Table 6.1: Summary of the software tools used by the OMERO team

Tool	Purpose /Activities
Devteam (IM Chat)	Used on a daily basis for conversation between members of the development team.
Email List	
Private list	Used to message all of the involved members on the OMERO project.
Public list	Used to provide support to the OMERO community, to answer technical questions and notify users of updates and releases.
Forum	Created to support the public list for OMERO support.
Teamspeak	Used to allow for remote group conversation.
Skype	Used to allow for remote group conversation.
Trac	Software bug tracking system.
Hudson	Extensible continuous integration server.
Plone	Project web site.
OmniGraffle	Visual diagram tool used for visualising development sprints.

A more recently adopted tool is OmniGraffle. This tool was used to manage the weekly cycles of development work on the project. The tool itself is an ad-hoc Gantt chart displaying each phase of the development process for the project (shown in Figure 6.3). This tool was the responsibility of Yvan who ensured that each member of the team had made his or her updates in the progress of work on a week-to-week basis. The Gantt chart would then be reviewed at every Friday meeting where more technical issues of the project were discussed within the team and consequently addressed.

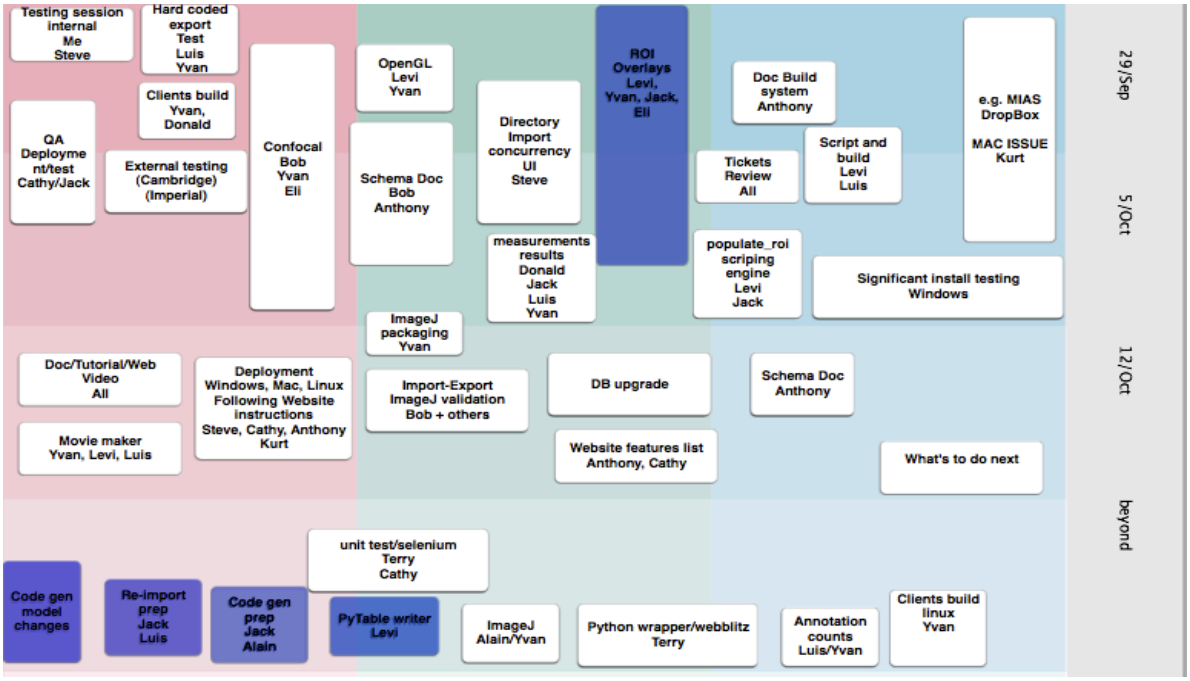

Figure 6.3: Example screenshot of a Gantt chart (Burel, unpublished)

A key part of the OMERO development week is the meetings that take place on Tuesday and Friday. The Tuesday meetings are focused on issues between the UI team and the OMERO team, and the Friday meetings are focused on internal issues of the OMERO project with just the OMERO team members in attendance. The general structure of the Tuesday meeting follows the presentation and/or discussion of the usability-led work conducted over the week. In the meeting, the results must be distilled down to the key points that may be relevant for the development work. The presentation and the discussion of the usability work helps to draw out more specific questions about Yvan's software development work with the OMERO.client, which was the central user interface of the OMERO software. The meeting will then address the general issues of the OMERO team. These issues will reflect the phase of the development process that the team is going through. If the OMERO team becomes involved in an in-depth technical discussion, the meeting can go on for a much longer period. Any outstanding issues, such as any promotional-led activities, visits to new potential scientists within Dundee, possible visits to external institutions, or funding-led activities in which the project is involved, will conclude the meeting.

The general structure of the Friday meeting follows the agenda set by the team during the week and is emailed out to everyone by Anthony. The typical structure of the OMERO team meetings, of course, can vary depending on the nature of the issues that need to be discussed. Unlike the Tuesday meetings, the general focus of Friday meetings is on the in-house workings of the OMERO team. The OMERO meetings can, on occasion, become a quite heated debate, this from the understanding gained from the embedded work within the OMERO team. This is more a reflection of the multiple sets of challenges and the potential perspectives to the solutions that the OMERO team have to deal with. But, of course, each member of the team does like to ensure that they express their own view. This, in part, is also part of the OMERO project challenge to meet and deal with the wider issues of the scientific community, alongside the technical development challenges. Because of this, OMERO meetings have a tendency to overrun. Yvan does take the responsibility for alerting us about this so he raises the point when meetings have gone over the one mark hour (the meetings could go on for some time if they are not managed appropriately). It has become a recognised accomplishment for the OMERO team and by Yvan particularly if the meetings are under the hour mark. Because of the work within the OMERO meetings and through the

time spent in the OMERO development office during visits, I could observe the involvement and contributions of Miles, the PI, to the project. Miles' scientific insight, in many ways, acts as a scientific portal for the OMERO team. During the meetings, Miles brought information of new and on-going developments from the scientific community. In many ways, he is the eyes and ears of the project in terms of understanding how and where future scientific developments lie scientifically. This was vital for the OMERO development team. Specifically, when Miles returns from conferences and events, he spends time discussing the key points that might be relevant for the team by sitting in the developer office while having a coffee and sharing his thoughts, experiences, learning etc. Bob, with his biology background, is always particularly keen to hear the details from Miles. In his contribution to the OMERO team, Miles does utilise the expertise available in the OMERO team to support the scientific work in his own laboratory, when necessary. The expertise of the OMERO team, ranging from mathematics, image analysis, and technical expertise, has been utilised and valued by Miles for each person's ability to make a scientific contribution.

In addition to the day-to-day practices, there are various external activities throughout the year that the software developers are involved in, such as promoting and demonstrating the software to both scientists and fellow scientific software developers.

- OMERO Software Demonstrations at various events within Dundee. This includes a lab and institutional retreats, and various poster sessions throughout the year.
- OMERO Software Demonstrations at various academic and commercial events. This presents the OMERO team with multiple opportunities to present and meet various types of existing user types and potential new users.

This has helped to expose the OMERO team and allowed them to receive instant feedback on the software as well as gaining an outside perspective. This aspect of the work strongly reflects the academic context in which the project finds itself, as it is critical for the project to promote its use through appropriate academic activities. The subsequent section now analyses the OMERO meeting notes (study 2) that I attended between the period of 27/10/2009 and 23/03/2010.The full details of my analysis are described in Chapter 4.

6.3 Ethnographic analysis

The full details of and background to the method used in the fieldwork are described in Chapter 4. All the fieldwork presented in Chapter 6 is available in Appendix 8. The analysis of the OMERO fieldwork revealed nine themes (see below). How they were constructed from the analysis is shown in the visual diagrams, which are available in Appendix 7. The nine themes are as follows:

- UI development
- New techniques
- OMERO components
- Community
- Current working practice
- Development tools
- Administration
- Project management
- Software collaboration

Similarly to the UI fieldwork analysis (see Chapter 5), I have chosen to explain the first six themes as these were the major ones that have influenced the analysis. This point is expanded in the summary of the OMERO fieldwork analysis in section 6.5.

6.3.1 UI development

The first key theme is 'UI development'. This theme was comprised of the following codes: 'own influence' (13), 'UI review' (3), and 'UI feedback' (3). All three codes of the theme are explored below. Importantly, for the perspective of my fieldwork, this theme gave a view of the UCD work that emerged from the OMERO meeting notes.

The code of 'own influence' was used to cover my impact in the OMERO meetings. Such an example is shown in extracts 1 and 2. Extract 1 describes the action assigned to me to observe the uptake of a new software project management tool called Agilo. Extract 2 then presents the feedback from Yvan after a one-month trial of Agilo. The

main benefit Yvan noticed was that the tool helped to replace the OmniGraffle tool (see Figure 6.3) and the visual representation of the information in Agilo was much easier to monitor and update.

Scott will try to monitor everyone's usage of Agilo
** Likes/dislikes*
Extract 1 – 2010-02-23 Tuesday meeting

Scott: any problems with Agilo?
** Yvan: easy to see what other people have planned*
Extract 2 – 2010-03-23 Tuesday meeting

The 'UI review' code aims to cover the discussions of the OMERO software interface in the OMERO meetings. An example is shown in Extract 3. Extract 3 is a discussion of how the OMERO.web client has been changed to provide a more consistent interface between the OMERO.web client and the OMERO.insight client. The meeting provided an opportunity to review the new three-panel layout with the entire team and for all of the OMERO team to discuss the details of the features that are covered in this change.

Web presentation
** now more insight-like with 3 panels*
** two different views of the middle data panel (thumbnail and table view)*
** default view: 2 panels (left+center) till dataset clicked (then 3 panels)*
** metadata panel shows: full image preview, comments, global metadata, acquisition data, tags, annotations (each on separate tabs)*
Extract 3 – 2010-03-23 Tuesday meeting

The three instances of the 'user feedback' code were made up of two types of user feedback that can be distinguished. The first type is shown in Extract 4, where user feedback has been brought back into the OMERO meeting. This extract itself is commenting on the use of the ImageJ software that is compatible with the OMERO.insight software. Because of a lack of documentation, this ImageJ software is very difficult to use. This conclusion was discussed in the meeting, and to avoid the same type of criticism for the OMERO software, the necessary action was taken to improve the information in the OMERO documentation.

Scott

** user comment: documentation on ImageJ install (postdoc)*

** first time ImageJ user*

** docs are poor; assume previous knowledge*

** movie & feature list? (one for usage but not installation)*

Extract 4 - 2009-10-29 Thursday meeting

Extract 5 covers the second type of user feedback where the OMERO team requested user feedback.

We need to go out to the community and find out what their use cases are.

Extract 5 – 2010-02-23 Tuesday Meeting

The OMERO developers requested user feedback from the community as described in Extract 5 because they more information on how OMERO can implement structured annotation and OMERO.tables. Based on their previous experience, the OMERO developers know that a "one fit for all" solution will not always work, so user feedback is critical for further implementation.

On reflection both types of the 'user feedback' code supported the view of the project. The first type of user feedback shown in Extract 4 gave me instances of use of the OMERO software and feedback to the OMERO developers, with the advantage that the entire team may hear the problem. In some way, I also felt that this was similar to the role of Miles when he gave information back to the team from when he had visited various scientific conferences, whereas the second type of user feedback supported a different view where more technical information was needed for the development of the software.

6.3.2 New techniques

A second theme to emerge was 'new techniques'. This theme was meaningful as it captured the growth of scientific development against the progress of how the OMERO software could or would adapt to these new methods. The theme was made up of the following codes: 'SPW' (8), 'NDIM' (8), 'HCS' (3), 'EM' (2), and 'OMX' (1). The

three most frequent codes of the theme are explored below.

The two codes of 'HCS' and 'SPW' will be discussed together as they are connected. The 'HCS' code represents the high content screening (HCS)[4] technique that the OMERO software supports. The related code of 'SPW', that means screen plate well (SPW)[5], represents a level of the image processing for the HCS technique. Extract 6 is a simple example of the 'HCS' code where the necessary bug fixing for the HCS data is explained for an OMERO release.

** 4.1 bug fixing*
** 4.1.1 release*
** insertions of HCS data has been reworked (collapsing settings, logical channels, etc.)*
Extract 6 - 2009-11-12 Thursday meeting

Extract 7 documents the actual initial implementation of HCS in OMERO. The term then used to discuss the details of the HCS technique is the code 'SPW' because this best described the setup of viewing the images in OMERO.

** SPW in insight*
** looks great: field view, heatmap thing, strip of fields, ...*
** see email from Yvan*
Extract 7 - 2009-10-29 Thursday meeting

The conclusion was that although in some instances the terminology was used interchangeably, I could make the distinction between the reference to HCS as an

[4] High content screening (HCS) is a method that is used in biology and drug discovery to identify substances that alter the phenotype of a cell in a desired manner. The influence of the tested substances is measured by automated image analysis. This type of experiment generates a massive amount of data that OMERO needs to be able to deal with (for display as thumbnails for example).

[5] The Screen Plate Well (SPW) model is designed to support High Content Screening. It is aimed at providing a flexible framework to organise the images that result from such a screen and link to external systems that contain full information about the components and products used. More information on this model can be found at http://www.openmicroscopy.org/site/support/file-formats/working-with-ome-xml/screen-plate-well/screen-plate-well-2010-04.

imaging method and the actual details for the imaging process the code ‘SPW’ represented.

The ‘NDIM’ code represents an area of work for expanding the amount of dimensional information stored in an image. The standard dimensions are spatial (XYZ), temporal (T), and for the different channels (C). This ‘NDIM’ code itself represents the effort of the OMERO team to support the new microscopy techniques that utilise more image information (e.g. angle, tile, and phases that can be important in some experiments). Extract 8 documents the steady progression of the decisions that need to be made and the need for user data.

ndim

** getting the timing right with the rest of the software community (ImageJ, ...)*

** our experiences with SPW, doing things slowly and right, and the half-way houses*

** several uses cases, real data.*

** will keep thinking about it.*

** keeping legacy API in place*

Extract 8 - 2009-11-12 Thursday Meeting

The central point taken from the ‘new techniques’ theme has helped to provide an insight into the growth of scientific methods from the perspective of the SSD process. This was evident from the codes in the theme. The following section now goes on to examine the range of parts that make up the areas of development in the OMERO team.

6.3.3 OMERO components

A third key theme to emerge was ‘OMERO components’. This theme was meaningful to my analysis as it represented the areas of development in the OMERO team. The theme was comprised of the following codes: ‘FS’ (10), ‘ROI’ (7), ‘Bio-Formats’ (6), ‘Data Model’ (6), ‘Web’ (3), ‘File formats’ (2), ‘Insight’ (2), and ‘Dropbox’ (1). The three most frequent codes of the theme are explored below. I have also chosen to discuss the ‘file formats’ code as it specifically relates to a previously discussed issue in Chapter 5 section 5.1.2.

The ‘FS’ code represents a specific area of development for the OMERO project called ‘file system’. Through the meetings I participated in, I learned that this was a change to the way the OMERO software imports the image data to improve efficiency, especially with larger image data sets. This code is in that sense linked to the ‘HCS’ code, as this type of experiment can generate large data sets. Given the prominence of the development of the feature while I was with the OMERO team, the ‘FS’ code has been used to signify the discussion around this feature. Extract 9 is an example of this work that is being carried out.

** FS*
** now working on thumbnailing, annotating, ... (standard container)*
** Kurt: commit today a first thumbnailing service (trying it out)*

Extract 9 - 2010-02-11 Thursday meeting

The code of ‘ROI’ has allowed me to have a technical background to the Region Of Interest function in OMERO. The ROI feature allows a scientist to measure the pixel intensity of their scientific image. Extract 10 covered a discussion on the future developments of storing the ROI information. The further point that can be made is that because of the complexity of forming a solution, the OMERO team proposed an investigation. With my involvement in the OMERO meeting, it provided a direct link to having a wider scope of the development process and its implications. Both the ‘FS’ and ‘ROI’ code are examples of this. However, in the ‘ROI’ code, there is an additional benefit of having further information about the ROI feature that may be used when discussing the feature with scientists.

** ROI evolution*
** primary problem/discussion, still on-going: storage & measurements*
** currently: some in DB, some in HDF*
** now need investigation*
** it adds another bullet point on roadmap with 10–20 hours*
** it's a blocker*
** will need to re-prioritize other tasks*
** Jack: just letting people do the investigations*
** Levi: also getting a list of suggestions for other investigation*

Extract 10 - 2010-01-28 Thursday Meeting

The ‘Bio-Formats’ code was especially significant for my background understanding of the other key aspects that work with the OMERO software. This was the principle reason why it was established under the OMERO components theme. Extract 11 documents regular updates on the development of Bio-Formats. This was important for the OMERO team because it ensures that the updates for the OMERO software and Bio-Formats can be released at the same time.

** Bio-Formats*
** tests for read/write of IO system*
** better testing (parallel)*
** integration with build system now*

Extract 11 - 2010-01-28 Thursday Meeting

The ‘Bio-Formats’ code was associated with the ‘file formats’ code as the Bio-Formats development provided new image file formats. Extract 12 describes three more file formats of incell, lei, and slidebook (three image file formats from three different microscopy companies), which were covered by the ‘file formats’ code. This was particularly relevant to my previous analysis in study 1 and the principles behind the DV file format (see Chapter 5 section 5.1.2)

** file format issues*
** incell fixed potentially*
** lei unfixed*
** slidebook unfixed (need files)*
** ...missed...*
** filing tickets for each*

Extract 12 - 2010-01-28 Thursday Meeting

6.3.4 Community

The fourth theme was ‘Community’. This theme was useful for my analysis as it captured the wider issues of both the development and user community of the OME project. The theme was made up of the following codes: ‘Community support’ (19),

‘Growing community’ (2), ‘External expectations’ (2), ‘Community feedback’ (2), and ‘Being open’ (1). The four most frequent codes of the theme are explored below.

The ‘Community support’ code captured the OMERO team’s responsibilities in providing setup support for the OMERO software. Extract 13 is an example where in each meeting the emails that have been sent to the OME email list and forum questions are ensured to be answered by a member of the OMERO team.

Emails / forums
** install issues seem to be up-to-date*
** ome-xml validation: needs to be re-written to understand 2009-09*
** in general, need to discuss support for latest schema in OMERO*

Extract 13 - 2009-10-29 Thursday Meeting

An additional aspect of the code involved an extension of this work, where a member of the team may take the necessary action to respond to the question. An example of this is shown in Extract 14.

Yvan :
** nd2 issues: Eden emailed, waiting for reply*

Extract 14 – 2010-02-11 Thursday Meeting

The ‘Growing community’ code describes the challenges being faced by the team regarding how to meet the growing uptake of the software when there are already many challenges to overcome with the software. This is demonstrated in Extract 15 with the remark by Miles that the user “wants it all”.

** Miles: User wants it all*
** easy, movable, see what I want*
** balancing act: when is an image in OMERO and when isn't it?*
** overriding issue: we've worked through server-fs visible, but data duplication isn't solved*

Extract 15 – 2010-03-09 Tuesday Meeting

The ‘Community feedback’ code acknowledges the OMERO team’s recognition that they require wider community feedback to develop the software and make the difficult decisions that are required. Extract 16 was a primary example of this. During a presentation by Miles, a discussion questioning how analytic results should be stored in OMERO started and the team realised they needed more community feedback.

> ** We need to go out to the community and find out what their use cases are.*
> **Extract 16 –** 2010-02-23 Tuesday Meeting

From the perspective of my analysis, the code of ‘External expectations’ was closely connected to the code ‘Growing community’. Given the growing uptake of OMERO, the speed of the image data must be imported quickly and efficiently as described in Extract 17. The implication of the ‘External expectations’ code gave me an insight into the OMERO development team’s expectations of the way they perceive the level of use of the software.

> ** Miles: screens have to go in quick*
> ** have to get people past idea that a single application for all their data*
> ** should only be high quality which goes in*
> ** managing expectations*
> ** Miles:"give me everything I want with complete flexibility and super-fast"*
> ** penalties for certain usages*
> **Extract 17 –** 2009-12-15 Tuesday Meeting

The understanding gained from the fourth theme of ‘Community’ demonstrated the demands placed on the OMERO development team due to the community and their related commitments. This was a reflection of the Community theme of that it was not directly connected to the scientific software development process but had significant implications for the team. Such examples covered were instances where the OMERO team had responsibilities for providing setup support for the OMERO software to the challenges being faced by the team regarding how to meet the growing uptake of the software.

6.3.5 Current working practice

The fifth theme to emerge was ‘Current working practice’. This theme documented the wider elements observed from working within the OMERO SSD team. It was made up of the following codes: ‘Technical solutions’ (8), ‘Agile’ (7), ‘Release’ (4), ‘Short term solution’ (4), ‘Bugs’ (3), ‘Communication’ (3), ‘Demo account’ (3), ‘Nightshade’ (3), ‘QA’ (3), ‘Bottlenecks’ (2), ‘Documentation’ (2), ‘Mini group meetings’ (2), ‘Sprint’ (2), ‘Technical constraints’ (2), ‘Example driven approach’ (1), ‘Investigations’ (1), ‘Software functionality’ (1), and ‘Tickets’ (1). The four most frequent codes of the theme are explored below.

The ‘Technical solutions’ code reports on the possible solutions to problems in the development of the OMERO software. The ‘Technical solutions’ code served to underline the complexity of the development process and the scientific environment. Two examples are shown in extracts 18 and 19. Extract 18 describes a discussion on how the information from the ROI solutions could be stored. Extract 19 covers a further presentation on the NDIM problem and notes the five solutions that have been proposed for this.

** mongodb & document db is Levi's favourite*
** bigtables-cousins would be great for purely measurements*
** but we want to do more relational work*
** normalization*
** do we actually use it as a relational db (RDB)?*
** at the import stage: objective settings, SPW, ...*
** making our life difficult, since we don't make any use of RDB*

Extract 18 – 2010-02-16 Tuesday Meeting

** NDIM*
** 5 solutions*
** adding D won't be used if another NDim solution used*
** re-using C is a proof-of-concept*
** stitching images together at a higher level*
** ndim & subdimensions: adding dimensions at the top or bottom*
** need to define the cost of doing this to know when/if to do it*

** NDim seems to be the only viable solution (stitching possibly then on top of that)*

Extract 19 – 2009-11-10 Tuesday Meeting

The code of 'Agile' describes the on-going transition to a more Agile software development process for the team. Extract 20 captures the initial discussion where the particular point is that the team is not fully Agile. This would previously re-enforce the existing work by Segal and Morris (2011) that SSD is selective in using the Agile methodology according to the book. Extract 21 attempts to explain this point further as it was discussed by the OMERO team that the Agile process was easier in the past because the size of the team was smaller.

** we're not Agile*
** Agile is a principle*
** process is different types of systems around Agile*
** centred around short iterations*

Extract 20 - 2010-01-19 Tuesday meeting

** was previously more Agile (as we were smaller)*

Extract 21 - 2010-01-19 Tuesday meeting

Extract 22 highlights the other parts of the Agile scrum process that the OMERO team is not adopting. Again, this would re-enforce the principle that Agile development methodology is not being used by the book.

** In some ways we are using a "Broken" scrum*
** No product owner*
** No stand-ups*
** No scrum master*

Extract 22 - 2010-02-25 Thursday Meeting

The 'Release' code describes the parts of the meetings where there is a planned release of the software. Extract 23 explains this process for the OMERO 4.1.1 version release of the software. The range of points cover software bugs that have already been fixed, changes to the OMERO.importer that have already been accepted for the release,

outstanding minor software bugs to be fixed, and an overview of the change made to HCS data.

** 4.1.1 release*
** nominally by end of tomorrow (mergings done)*
** Friday/Monday was spent cleaning up some RE mess from release*
** also fixes channel concurrency issues*
** insertions of HCS data has been reworked (collapsing settings, logical channels, etc.)*
** Steve merged importer changes into 4.1*
** importer changes: sending log file, setting logging, cli attachments*

Extract 23 – 2009-11-12

The 'Short term solution' code examines the OMERO team's proposal to offer a stopgap to a problem. This code provided an insight into understanding what is technically feasible for the next release. Extract 24 covers the upcoming release of version 4.2 and what work should be done for the ROI data storage. The implication of the code for my own analysis concerns the reasoning in the development decisions that were made for the software. This was ultimately about recognising the project constraints for the current release and beyond.

** transitional phase is an issue*
** do we want to work toward a single storage mechanism?*
** do we use the best storage mechanism for a particular type?*
** what's the scope? (4.2?)*
** which project would show the utility?*
** Jack:*
** don't currently utilise the ROI object in db correctly*
** InCell you'd tag every ROI with correct tag (cell/nucleus)*
** ROI as an object is un-utilised*
** getting us (and our users) out of a situation.*
** Levi: currently blocked. Make this decision sooner rather than later.*

Extract 24 – 2010-02-16 Tuesday Meeting

6.3.6 Development tools

The sixth theme to materialise was ‘Development tools’. This allowed me to account for and question the range of software tools that were used by the OMERO team. The theme was made up of the following codes: ‘Graffle’ (18), ‘Trac’ (12), ‘Wiki’ (3), and ‘Hudson’ (1). The three most frequent codes of the theme are explored below. This theme developed from refining the ‘current working practice’ theme.

The ‘Graffle’ code captures the use of the OmniGraffle software to construct a Gantt chart. An example of this was shown previously in this chapter in Figure 6.3. The information captured in the code covers the use of the OmniGraffle software. Extract 25 notes that the current tasks on the Graffle diagram will be moved because of the current work on the release of the software. The code has also shown the transition to abandon the use of the tool. Extract 26 shows a point made by Anthony about the Graffle tool and why it was being phased out.

> ** Graffle: most tasks pushed back for work on 4.1.1 (QA started this week)*
> **Extract 25 –** 2009-11-19 Thursday Meeting

> ** Anthony: Graffle doesn't show dependencies, useful in Gantt*
> **Extract 26 –** 2009-10-27 Tuesday Meeting

The ‘Trac’ code captures the use of the project management and bug tracking system to manage the software process and the instances where it was discussed in the meetings. The ‘Trac’ code was significant for my own analysis as I used it during my stay with the OMERO team so I did have some first-hand experience of its use. This did influence my own impressions of the use of the tool as although it provided the project management and bug tracking for the software project, it was flawed because of the volume of information held about the project. The code that best describes the use of ‘Trac’ in the project is captured in Extract 27. The significant aspects discussed through this extract describe how there is a large amount of unscheduled tickets (a ticket is a actionable task that can carried out by a OMERO software developer). This means that there is a ‘sea of tickets’ as explained in the meeting that can subsequently lead to missed tickets and difficulties in managing and visualising the ticket information.

** trac*

** dead pool of tickets sitting in "unscheduled"*

** everything in trac is a sea of tickets*

** lots of tickets we missed*

** no way to visualize all the tickets*

** no scheduling, strictly milestone based*

** Jack: communication component of trac*

** "here Levi, please ..."*

** Yvan: cF. QA also reporting to the user*

** Dev2Dev communication and to user*

Extract 27 - 2010-01-19 Tuesday Meeting

The 'Wiki' code describes the conversation in the meetings on how the wiki pages for the project can be utilised for the benefit of project organisation. The wiki was integrated with the trac project management and bug tracking system to provide the necessary links. Extract 28 discusses how the proposed new layout and format for the wiki pages should be created. The particular key point not discussed in the extract is that a wiki page is being constructed for communicating the software development process to members of the OMERO community. Extract 28 explains that to do this the aim of a wiki page is to show how complete a development task is, so that an external developer/user knows when they can download an update.

** wiki pages*

** took 1 hour to create wiki pages for all roadmap bullets*

** don't need tickets on roadmap page then*

** description at top*

** usage at the top (screenshots at bottom?)*

** looks better from external point of view*

** too many things on a single page?*

**Luis: if I have to create a wiki page per ticket, I won't*

** if something has only one ticket?...*

** what's the purpose of the wiki pages?*

** showing external how done it is*

** knowing when they can download*

* * status
* * no tables in formatting
* * must-have/good-to-have?
* * lots of discussion
* * Yvan: "good-to-have" has no choice of being done
* * Anthony: related work, good to have it together
* * should only be tickets
* * per milestone? Yes
* * dependencies?
* * estimates?
* * ... we just spent an hour building a tool
* * just treat the must haves?*

Extract 28 – 2009-10-21 Thursday Meeting

6.4 Additional activities: Community use

Along with my analysis of the OMERO fieldwork, my further experiences within the OMERO project also provided additional insights for the research. The supplementary information has been labelled as 'Community use'.

There is an immense pressure to present and promote the uptake of the OMERO software, which involves a range of community led activities. This is primarily dictated by the requirement for continued funding, so the importance of demonstrating current use and the uptake of the software is significant, as are demonstrations related to grant renewal or new funding opportunities. This further re-enforces the reputation of how scientific software operates within a reputation-based economy that values scientific publications (Howison & Herbsleb, 2011). This information was significant in understanding the evolution of the community term in this research. Another benefit was in understanding the significance for the community within the context of SSD. Figure 6.4 presents an adaption of the work by Gabriel and Goldman (2002) to show the communities in OMERO. Unlike the original illustration by Gabriel and Goldman (2002), there is no natural progression from the scientists to a developer of the software code. However, this is a feasible model, as previously discussed by Segal (2007) and the role of scientists as end-user developers could be possible, although this role is not

applicable to the context of the OMERO project. For the adapted Figure 6.4, the community is centred on the circle labelled code and the role of the core OMERO software development team is to deal directly with the scientific software code. Figure 6.4 also highlights the distinction between UCD and the role of the core software development team.

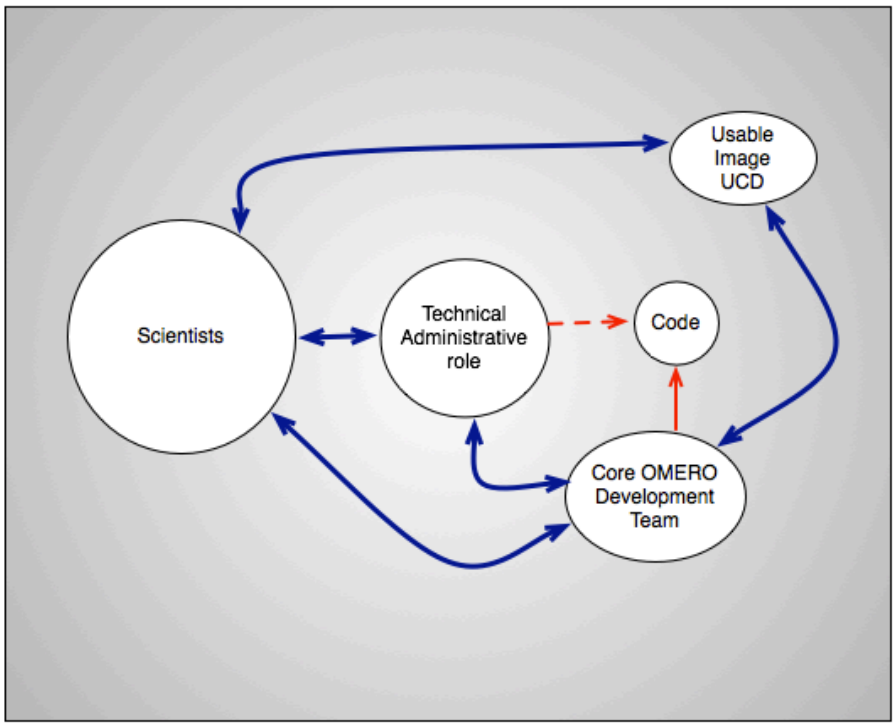

Figure 6.4: Communities built around common interests (Adapted from Gabriel & Goldman 2002)

The promotion of the OMERO software for the project is performed through the project promotion material such as posters and project presentation. The OMERO project website plays an important role as a first point of contact for community use. The website serves as the point for downloading the OMERO software and as the source of information about the project and the software. The website brings a high level of accountability for the OMERO team in terms of maintaining and managing the information.

An eternal question raised through the release schedule for the OMERO team is how up to date to keep the documentation, so various techniques have been used to document the system. This is key, not just because of its importance but also because of the time investment required by the team to produce documentation and the limited resources

available to the team so it is important to produce effective results within these constraints. One way the team has tried to remedy this is by creating video tutorials to provide information on the various features and how to use the system. The concern regarding documentation is also related to the wider responsibility of the OMERO project for serving the entire community. In the context of the OMERO application, this covers the scientists who use the software, the system/technical administrators who install and run the system, the bio-informaticians, and scientific software developer's roles. Therefore, the range of information for OMERO must address the full range of these users and potential further variations in users.

The small cross section of issues covered through the promotion of community use also presents a key challenge for the OMERO project. This may be traced back to the technical nature of the project and its position as an enterprise-level scientific management software tool. Because of this, the software tool has to be able to consider and take on a wide scope of users from the front-end scientists that use it to manage and analyse their image data, to the systems administrators and/or scientific developers that are required to deploy, modify, or collaborate with the project. This forms a critical perspective for SSD projects, with the key for their sustainability being to promote use of the software throughout the scientific community. For such a project this means earning the scientific academic credibility that the scientific community requires, in addition to promoting collaborations for software growth into new scientific communities.

6.5 Summary of the OMERO fieldwork analysis

This section will review the insights gained from the analysis of the OMERO ethnographic work. From my own analysis, the two key aspects to be taken from the work are as follows:

1) The combination of the 'OMERO components', 'New techniques', and 'Community'.
2) The 'Development tools'.

The benefits of the ethnographic method with being deeply embedded in the OMERO

project has been clearly seen and captured in the themes of ‘OMERO components’, ‘New techniques’, and ‘Community’. The ‘OMERO components’ theme gave an insight into the core aspects of the project, from the ‘Bio-Formats’ code that supports the variety of image file formats the OMERO software supports to the more specific aspect of new development of the OMERO infrastructure and function that the ‘FS’ code highlighted. The theme of ‘New techniques’ identified a connection between the science and the SSD process. The ‘NDIM’ code was a primary example of this with the necessary proposals being made by the OMERO team to handle the development of new imaging techniques. Finally, the ‘Community’ theme helped to develop a sense of commitment to provide the necessary software support at a technical level with the code of ‘Community support’ demonstrating the need to provide on-going maintenance to the community. In addition to this the aspect of ‘Community feedback’ played a key role in supporting the difficult decisions the team made. The ‘External expectations’ code brought a heightened awareness of the wider community prospects and of the broader requirements that are made by the community.

In addition to what these themes brought individually was an overall understanding of my own position in the project. This is discussed in the following section (6.6). The significance of the ‘Development tools’ theme for the work was in how it was first deducted from the ‘Current work practice’ theme and the problematic codes it presented such as ‘bugs’, ‘bottlenecks’, and ‘technical constraints’. The problematic aspects of the development tools were picked up in the ‘Graffle’ and ‘Trac’ codes. The ‘Graffle’ code was probed as the software tool was being withdrawn from the OMERO team because of its inability to effectively link information for the benefit of the team. This point is explained in Extract 26 where Anthony points out that the Graffle code did not show the dependencies that a typical Gantt chat would provide. The specific ‘Wiki’ code in Extract 28 explored more closely the changes for documenting the development process, with the aim to improve the management of the development process. It was the combination of these themes and the return to my own research questions that led me to question the roles of the development tools in terms of facilitating the scientific software developer practices.

6.6 Discussion

In this chapter the analysed data from study 1 (the Usable Image fieldwork analysis) and study 2 (the OMERO fieldwork analysis) were revisited and the findings were compared. The first similarity observed was that the ethnographic method yielded a level of empathy for those involved in both studies. In study 1, it helped to form empathy towards the scientists as described in Chapter 5 section 5.4. When comparing this data in study 2, a similar insight and empathy towards the SSD process has been made about the OMERO scientific software developers and the challenges they face in the SSD process. Such an example code that demonstrated this in study 2 was 'Growing community', which highlighted the challenges the team faced in terms of how to meet the growing uptake of the software. The code of 'NDIM' presented a different perspective of the SSD challenge with the demands and on-going developments of the scientific environment. Also, the 'Graffle' and 'Wiki' codes demonstrated the challenges with software tools and the 'Technical solutions' presented an insight into the complex decision making made by the OMERO team for the development of the OMERO software. These codes are the examples of how the empathy for the OMERO scientific software developers was established based on the SSD work and challenges that they are involved in.

In further examining study 2, the two themes that could best describe the general observations issues of the SSD project were the 'Current work practice', which related to the software development process, and the 'Community' theme, which related to the wider issues of the OME project community. The 'Current work practice' theme covered such codes as 'Agile', 'Technical solutions', and the 'Development 'tools'. It encapsulated the many principles of software engineering, which is the focus of the project. The second theme, 'Community', covered the codes of dealing with the 'Growing community' and 'Community support'. The reason this theme was selected is because it is central to the continued use and uptake of the software. This is because of the need to support the community in their use of the software and the need to gain community feedback. The theme 'New techniques' holds a close relation to the support for the community because it highlighted the new and on-going developments in the scientific community. This was exemplified in the way the OMERO team actively planned to support advancements in microscope techniques in the 'NDIM' code (see section 6.3.2). These were the major aspects to be taken forward for the purpose of my final research question. However, what was still missing from these points was the role of UCD. After iterating between studies 1 and 2 further, the one aspect I had overlooked

was my own influence in the analysis particularly in study two. The code 'own influence' existed in study two and along with the presence of the 'OMERO components' theme, the combination gave me an insight into the wider operations of the OME project and OMERO software as a whole. As I did not only carry an analysis of the single part of software relevant to the UCD covered in the code 'insight' but also with the 'OMERO components' theme, it allowed me to form a more global view of the related components of the software codes. The 'Bio-Formats' and the 'data model' codes were examples of this. While it was not critical for the role of UCD to fully comprehend the full technical details, the view did help to situate my own view between both projects, which were arguably both a UCD view and an SSD view.

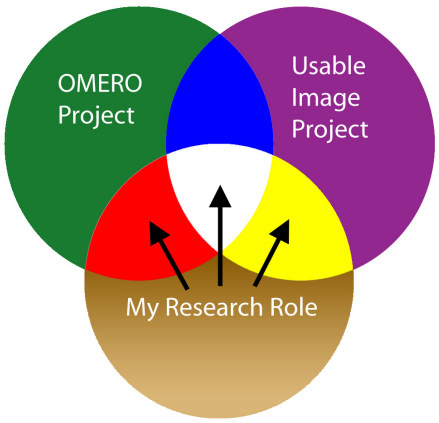

Figure 6.5: The culmination of perspectives gained in the research

The combination of the insight from the analysis of studies 1 and 2 helped to form a dual perspective, which were both the scientists and scientific software developers. Because of this, I re-examined the challenges previously examined by Seffah and Metzker (2008) for integrating UCD and software engineering (See Chapter 2 section 2.7). In my research, obtaining a double perspective between study 1 and 2 incorporated and supported the particular challenge of the cultural gap between UCD roles and engineers, as previously described by Seffah and Metzker (2008). The cultural gap between UCD roles and software engineers is described to be due to the different culture. The misconception between the two disciplines covers different terminology and a technology centric view in software engineering to the contrast of a user focused and way of working in UCD. It also specifies that UCD specialists must understand how and why the technical choices influence the end design. Through my research I

have gained an understanding to the how and why of the technical choices of the OMERO project. The findings from the analysis indicate that with the recognition of the 'OMERO components' theme gave way to an understanding of how and why the technical choices can influence the end design of the OMERO software. This was the critical path to working between the cultural gap between UCD Usable Image project and the SSD project of OMERO scientific software developers. This is not to say that the misconception between the two disciplines of UCD and SSD can disappear completely, given the differences in the terminology and where the focus of each lies. But in returning to my own research question for answering how the uptake of UCD philosophies, methods, and thinking in the application of academic SSD can be improved. I required a way to frame the perspectives of the research of the SSD and UCD without neglecting the wider community that the SSD project is situated. This led me to examine a way to form a framework that may embrace and build on these principles.

Chapter 7: Forming a Framework

7.1 The move towards the Project Community Framework

This chapter discusses a proposed framework that has emerged from the experiences and observations in the Usable Image fieldwork and the OMERO fieldwork (see Chapters 5 and 6 respectively). In reflection of these experiences the work has devised a framework that conveys the requirement for the re-thinking of academic SSD. Chapter 7 now covers my own personal response to the fieldwork and subsequent rational that has emerged from my position in the field.

The research was conducted with a wide range of design constraints operating in the environment within an SSD project. The observations from the Usable Image fieldwork analysis uncovered the main question for investigating the SSD context. The subsequent analysis of the fieldwork has cited the three aspects of SSD, UCD, and community as being significant to forming a more holistic view of the SSD process – so not only benefiting the integration of UCD with SSD but also for the position of the SSD project.

What has been learnt through the research is that a lack of UCD in SSD can be caused by the model of scientific end-user development, as the scientific software is small and can be disregarded once the scientific answer is obtained. As previously explained by Segal and Morris (2011) with their scientific end-user software model, both software design and usability become a smaller issue (see chapter 2 section 2.9) when the end-user is the developer of the software and the software can have a limited exposure to a wider audience. The research has also emphasised how several funding bodies have acknowledged a lack of UCD practice in SSD (see Chapter 2 section 2.7.2), which has meant a limited application of UCD practice in SSD. The research has also reviewed how the practices of scientific software developers can cause conflicts with interacting with scientists, as scientific software developers can follow conventional software engineering practices such as getting the software requirements up front (Segal and Morris 2008; Segal, 2008). Combining the already recognised practices of SSD from the existing research with the fieldwork analysis of the SSD practices of the OMERO project and questioning how SSD is undertaken in academic contexts led me to identify key factors of the scientific software developers' practices, but it also made me aware of the importance of the community.

Based on these findings, this research proposes that a more context-specific proposition for SSD is made. Although several existing usability engineering methodologies that have been reviewed in Chapter 2 section 2.4.1 have demonstrated the management of integrating UCD and software engineering, I would argue that my research does echo the earlier point made by Aikio (2006) who states that trying to find a generic solution for the integration between UCD and software engineering is difficult because of the variable factors for any given integration case. Such an example is the observation of the cultural difference of science and the scientific software development process. The scientific process uses a trial and error approach to discover and eliminate ideas among a spectrum of thoughts to explore (Kane *et al.,* 2006). The work by Segal and Morris (2008) and Segal (2008), has studied this aspect of working with scientists to be particularly challenging for scientific software developers who are following traditional software engineering practices and want the software requirements up-front; however, to scientists this is a very difficult and unfamiliar practice because they are more familiar with a trial and error approach. So to answer my final research question of how to improve the uptake of UCD philosophies, methods, and thinking in the application of an academic SSD, a more individual approach is required. Because of this what follows concerns the process of how to answer the question for my research, as the research addresses a different and complex model of scientific software development in comparison to the professional end user model of development described by Segal (2004) (See Chapter 2 section 2.8 for a full description of the professional end user). The higher complexity of the type of scientific software OMERO is requires a specialised software development team needing both expert scientific software developers and UCD expertise. Therefore, both the gap and consequent integration between UCD and SSD is a prominent and significant question for the scientific software development process.

In making these conclusions, this research is calling for a new philosophy for UCD in SSD that positions UCD within the complex environment of SSD. The step towards this proposal is a framework that acknowledges and understands the specific design constraints of the academic OS context of SSD. The purpose of the framework is to draw out and communicate the new thinking for UCD in SSD, to enable and encourage the SSD team to use the SSD, UCD, and the SSD community information held internally and externally to the advantage of the project. The research creates a framework so that the principles of

scientific software development, UCD, and community may be considered in unity and not just in isolation. This framework aims to explore the insights of this new philosophy. It has been created as a way not to predict the gaps of SSD but instead to use the information that is currently available to inform the decision-making process of academic SSD.

The proposal of the use of a framework for the research has drawn upon the process of frame analysis (Goffman, 1974). This involves the study of the organisation of social experiences and may be used to analyse how people understand situations and activities. The direct relationship I see of this with my own research is with the scientific software development team and their role for understanding the situations and activities of SSD. The metaphor Goffman uses to illustrate frame analysis is how people structure (frame) the content (picture) of what they are experiencing (observing) (Treviño, 2003). While frame analysis is valuable for the purpose of this research, Goffman (1974) notes that perspective is situational and can be experienced with other individuals, and it may even be distributed because "*retrospective characterisation of the same event of social occasion may differ very widely*" (Goffman, 1974, p.9).

The final question of this research concerns examining how SSD is undertaken in academic contexts and how the uptake of UCD philosophies, methods, and thinking in the application of academic SSD can be improved. Based on this question and through the fieldwork analysis of studies 1 and 2, the perspectives of both the UCD and the SSD have been investigated and they demonstrate that in the development of scientific software there is a multiplicity of perspectives (frames) around the process. The two frames of scientific software development and UCD were presented in Chapter 4 Figure 4.4, which were adapted from Seffah *et al.,* (2005). The specific applications of the two system perspectives to the OMERO project were also discussed in Chapter 4 section 4.2. As already mentioned in Chapter 4, the work by Seffah *et al.,* (2005) highlights how the perspectives differ. Briefly, the software engineering perspective of System 1 is driven by specifications that are provided for defining the application, including the interface. The user interface has to meet the functional and usability requirements, but the requirements are tied to the system, which corresponds to the application itself. The focus is on the software application and the interface is one of many components that have to meet certain requirements. The perspective of System 2 incorporates UCD, with the focus on the priority to ensure that

each user may perform the required set of tasks within the application.

The fieldwork of the Usable Image project and the OMERO project led me to agree with this perspective of Seffah *et al.,* (2005) and the System 2 perspective and the framing of the challenge for two reasons. The first reason is that the analysis of the Usable Image fieldwork provided an understanding of the use of the OMERO software and an understanding of where the UCD problems lie. In terms of the research, this helped to further hold a level of empathy with the scientists and also to increase the awareness of a UCD process. The second reason is that from the analysis of the OMERO project fieldwork, there was an increased awareness of the range of challenges and problems an SSD team may face, which covered instances of their current working practice of the scientific software development process and the responsibility of the OME project to the current and on-going issues of the OME project community. This perspective illustrated aspects of the System 2 view but not entirely. According to Seffah *et al.,* (2005), when an application's degree of interactivity and interface complexity is high, then there is a recognised an imbalance between the UCD and the SSD and the System 2 perspective should prevail. This point was being put into effect with the presence of the UI project, although the question of the SSD and its community remained.

To this end, the research work now puts forward the idea that for complex SSDs, a System 3 perspective should evolve. The evolution of the System 3 perspective is illustrated in Figure 7.1, where it shows how the frame and perspective of the community span the work of both the scientific software developers and the UCD roles in a project, and it accounts for both the SSD and UCD perspective view of the community. The third perspective of community has been added for the context of the SSD project, as the priority is to ensure that there is also a higher level of focus for the SSD and UCD process. This is to underline the responsibility to the community by the SSD project team for the software users and for the potential future users.

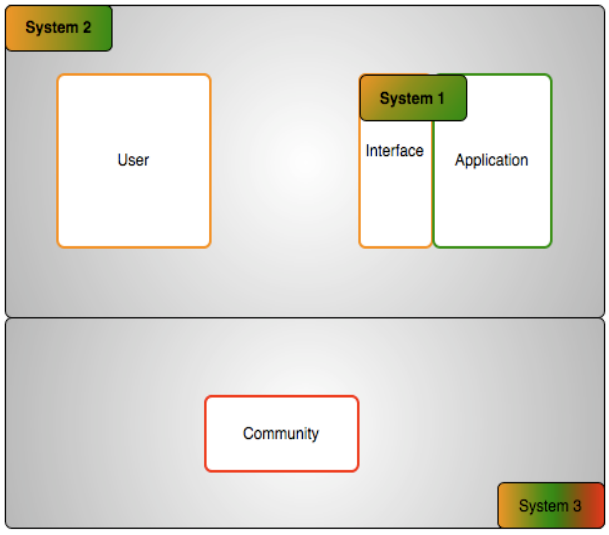

Figure 7.1: A System 3 perspective

For the framework the whole SSD process may then share three natural levels of emphasis on the following:

- The continued development and evolution of the software.
- The process of UCD and its relation to the development of the software.
- The community of the software and project – to maintain sustainability and growth.

It is important to note that UCD and SSD are considered with the same frame with the aim of working closer together for an SSD project.

The selection of the framework approach has also been created with an awareness of avoiding the concept of the "ontological drift". This term, as discussed by Robinson and Bannon (1991), occurs when a design passes between many different professional groups, each with their own worldview and specialised language. These professional groups are termed ‘semantic communities’. When a design is passed within and between the semantic communities, some things are lost and gained; consequently, the design work cannot be measured in an equal fashion. The significance of this for the research lies in the previously identified point of both the educational gap and the cultural gap between software engineering and UCD (see Chapter 2 section 2.6). Both these gaps can result in communication difficulties between software engineers and those in UCD roles. Within the context of SSD, the direction of communication can be identified to move between at least

three different professional groups (the scientific software developers, UCD roles, and the scientists), each with their own worldview and specialised language, so any measure that may help to support this was considered. The research work has therefore actively chosen not to adopt a model-based approach, because of the very mixed and interdisciplinary context of academic SSD projects.

7.2 How to build the Project Community Framework

There are four steps required to capture the key elements defining the Project Community Framework (PCF):

I. The capture and characterisation of the Project Community.
II. The storage of the Project Community information.
III. The process of UCD.
IV. The Project Community action and reflection.

The framework has been created with the purpose of accommodating the design and development of scientific software. The steps provide a platform to frame, store, and iterate through the information of a project as well as, explore, collaborate, through the evolution of the community surrounding the project. The steps of the framework do not require a comprehensive level of information to be added each time or an extensive amount of time spent at each step. The purpose is to use the steps as a way to create a collection of sketches of the Project Team and Community in order to build a detailed picture of the Project Team and Community over a certain period. Box 7.1 defines the key roles of the PCF that shall be used throughout this chapter and the remaining research.

Project Team: The Project Team is the people whose main activity is the academic SSD project. The Project Team comprises the Project Community Mediating members and any further members that have more specialised functions within the project (e.g. programming). The Project Community Mediating role(s) is to act as a catalyst for the progression of the work of the Project Team as a whole.

Project Community: The Project Community is the people outside the Project Team whose main activity is not the academic SSD project itself. Though indirectly involved or connected to the project, the Project Community can be used to identify roles that are not contributing to the Project Team, yet hold interest in the output of the software.

Project Community Mediating Role(s): This role is the key role for adopting, running, and promoting the Project Community Framework within the Project Team. The responsibility for this role is for the cross-pollination of ideas between the disciplines in the Project Team. The occupier of this role may come from many different backgrounds, but the key for it is in the promotion of interdisciplinary understanding within the Project Team.

Project Leader: The Project Leader is the principal investigator (PI) who directs and manages the project.

Funding Bodies: The funding bodies have an authoritative role in the Project Community. They are not part of the Project Community but they do overlook the academic SSD project and influence the composition, growth, and fate of the Project Community.

Collaborative Contribution: The term used to define the growth of the Project Team.

Box 7.1: Project Community Terminology

It is acknowledged that when using the PCF, one should keep in mind that it is one's responsibility to translate the PCF to one's own unique circumstances. This is a quality of the PCF – to remain open to the interpretation of the numerous variations of what may make up a Project Team. The critical contribution of the PCF to the process is to promote and raise awareness of the integration between the SSD, UCD, and the community of the SSD project. In recognition of this, the following information in Table 7.1 shows the various roles that may exist in a Project Team and the steps that may be applicable to them.

Table 7.1: Early roles identified for the potential application of the Project Community

Role	Steps – Applicable for Possible Use
Software Developer	Steps I, II, III, IV
Bio-Informatician	Steps I, II, III, IV
User Centred Design Role	Steps I, III, IV
PI of SSD Project	Steps I, III, IV
Scientists End User (Scientists developer)	Steps I, II, III, IV

The key role for the use of the PCF lies with the Project Community Mediating role(s); this role(s) can be comprised by more than one individual who can come from various types of expertise evident in different background areas of a project (a selection of these is shown in Table 7.1). A Project Community Mediating role(s) takes on the responsibility of advocating their own role (e.g. UCD) but critically must be able to draw together and promote awareness between the different perspectives of the project (the SSD, the UCD, and the community of the software). This is critically with the intent of integrating the mix of perspectives. In the next sections, the PCF steps will be detailed.

7.3 Step I: The capture and characterisation of the Project Community

Step I defines the context of the PCF. The motivation for completing Step I is to outline the people and tools for the project. It is divided into the following:

- Defining the Project Team and Project Community.
- Recommending the project tools.
- Raising points of awareness for specific Project Community Mediating roles within the project.

The last point is optional but can be valuable for some Project Community Mediating roles within the Project Team. The construction of the Project Community involves the identification of the Project Team and, through this process, the identification of the wider project context. The purpose of an initial view of the Project Team and Community is to begin establishing the project team's context.

The aim is not to draw an exhaustive picture at the start of the framework, but to make one that begins to identify who is in some way connected to the project. This, by its very nature, can be a subjective process, but the intention is that it may continue to be re-evaluated over time. Step IV describes the iteration of this Step (see section 7.6). The aim is to know and to recognise who is involved, and what roles are required, which in turn may help to identify what the missing elements are (e.g. more specific people) (see Box 7.1).

7.3.1 Defining the Project Team

The Project Team is the people whose main activity is the project itself (see Figure 7.2 and Box 7.1). Figure 7.2 provides an example of the four types of roles in an SSD project. The figure also illustrates the range of cross-over roles that can occur, such as a system administrator having a scientific software developer role and a UCD role work between a scientific one. Figure 7.3 provides an example of a team member role and its related surrounding role duties. For the academic SSD project, this may range from small-scale setups, that is 3–4 people covering and working in many different roles (system administrator, developer, UCD, etc.), to larger teams of 10–12 people that has the ability to cover a selection of more specialised roles such as bio-informatician, scientists end user, etc. Smaller projects require members of the team to work between multiple job roles (i.e. a person will occupy multiple positions within the project).

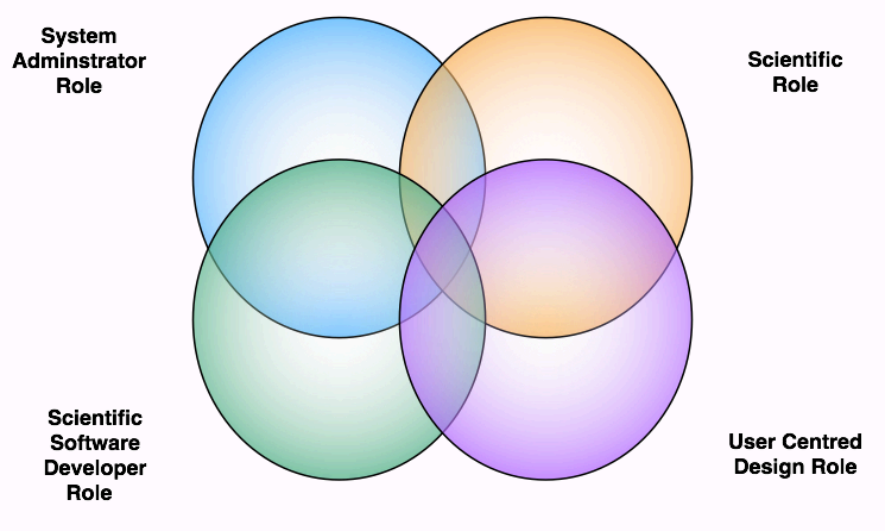

Figure 7.2: The scope of domains of a Project Team

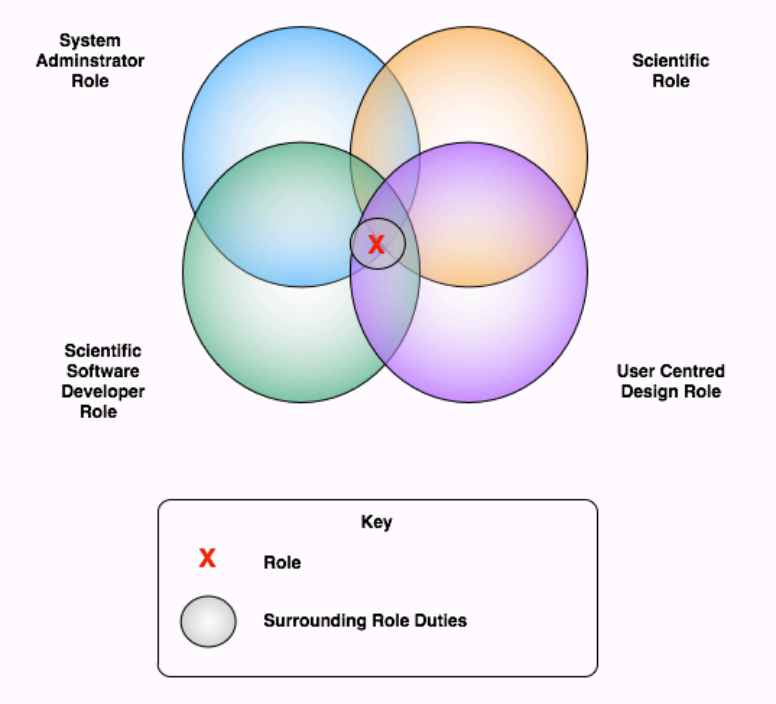

Figure 7.3: The scope of a role in a Project Team

For the Project Team, there are two possibilities for team members as the project grows:

1. Team members may continually reconfigure their activities in response to the changes that the task demands as the Project Team adapts to take on multiple roles. This is the antithesis for the position of a Project Community Mediating role.

2. Team members can bring their individual perspective of the SSD project to help in the information for coordinating and the organisation of information of the SSD process with the Project Team.

A Project Team must work to have both types of members. This is illustrated in Figure 7.4. Point one on Figure 7.4 relates to the issue above regarding the Project Community Mediating role. It shows how a scientific software developer role has drawn on further roles of a UCD role, a scientific role, and a system administrator role. Point two on Figure 7.4 encircles the original role and shows the growth of the role as a dotted line. With a limited

size team, which is ‘typical’ of academic science projects because of funding constraints, people are required to take on multiple tasks for the Project Team to operate, which supports the decision to have a software engineer within an SSD project to take on tasks outside their job description. This is an assumed and understood part of the job for academic based research.

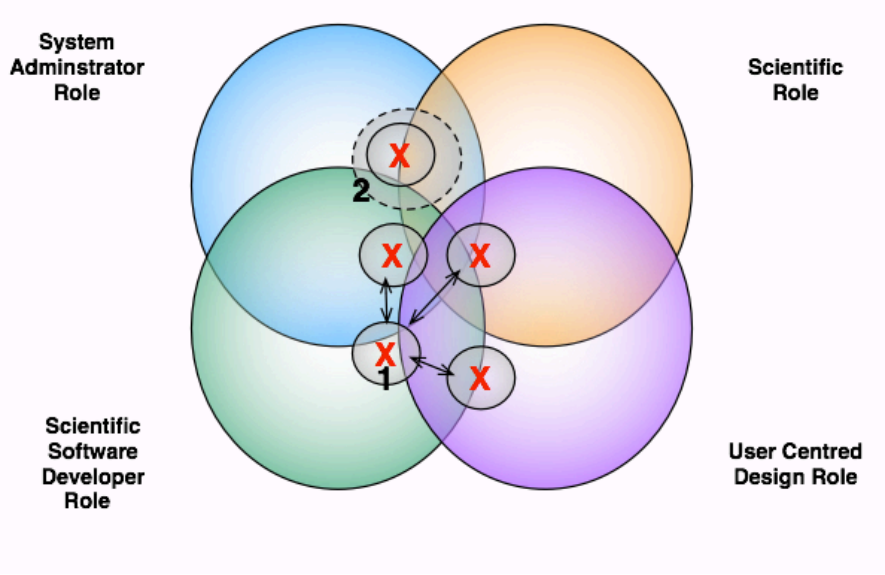

Figure 7.4: Role duties

A key responsibility of the Project Team is to ensure that the members are capable of and willing to accept their roles within the Project Team. This particular point again helps to emphasise the requirement for an expert UCD role in SSD projects because of the need to utilise the expertise of UCD role. The PCF has paid particular attention to this aspect as a core previously acknowledged challenge was with software engineering lacking the expertise of UCD. This underpins a central element of professionalism for all roles in the Project Team.

7.3.2 Project Team Example

A Project Team has funding for five years for seven team members; in the second year the project has an additional set of funding for 3 more people for 3 years. The team comprises 2 UCD roles and 8 scientific software developers.

7.3.3 Project Team Variable Conditions

The project team variable condition is the funding of the project, which is the central factor that can determine project sustainability. It is recognised that funding may come from alternative sources outside of the general funding councils such as commercial sponsored grants. This impact and effect it may have on the Project are yet unknown, as they have not been explored in this research. Because the Project Team is defined as the people whose main activity is the project itself, the purpose of recording the goals of the Project Team over time is to record its evolution, so that it may be used to inform current and future design and development both the Project Team and the Project Community. The goals must incorporate not only the SSD development goals, but also the UCD goals, and the goals for supporting the community.

How the Project Team may evolve over time depends on the success of the project's funding renewal. The Team may also expand through collaborative work between similar projects. This requires a Project Team to invest a certain amount of time with any collaborating project and so can potentially take out the SSD Project Team members from their core areas of work.

7.3.4 Project Team Outcomes

The outcome of defining the Project Team is the identification of roles and matching these with people, which is done by drawing attention to certain aspects such as possible issues when working as a team, missing roles, people. For the project leader, defining the Project Team aims to promote and support ways to think about funding/grants in a more strategic way. The identified benefits of the Project Team are as follows:

- To recognise where additional support of people/roles is required (e.g. support for the system administrator).
- To recognise the need for new roles for the Project Team in the categories of SSD, UCD, or the uptake of the software in the community (e.g. new technical support).

Defining the Project Team may provide awareness of the role of the Project Leader, and highlight and question the contributions of supporting roles outside the formally defined funded model of the project.

7.3.5 Defining the Project Community

The Project Community can also be defined as the people outside the Project Team – the people whose main activity is not the SSD project itself, but are indirectly involved or connected to the project. The Project Community can identify roles that are not contributing to the Project Team, yet have benefits of its use through the output of the software.

The Project Community can be defined through the work of Communities of Practice. The Project Community is formed by the people who engage in a process of collective learning in a shared domain of human endeavour. There are three crucial elements in distinguishing a community of practice from other groups and communities (Wenger, 2007).

- The Domain.

It has an identity defined by a shared domain of interest. Membership therefore implies a commitment to the domain, and therefore a shared competence that distinguishes members from other people.

- The Community

In pursuing their interest in their domain, members engage in joint activities and discussions, help each other, and share information. They build relationships that enable them to learn from each other.

- The Practice

Members of a community of practice are practitioners. They develop a shared repertoire of resources: experiences, stories, tools, and ways of addressing recurring problems — in short, a shared practice. This takes time and sustained interaction.

7.3.6 The Project Community Example

A simplified example of the communities within the PCF is demonstrated in Figure 7.5. This figure shows three example types of communities with which an SSD project may interact. Community A is a scientific software engineering community, community B is an example of a collaborating project, and communities C and D show two different scientist end-user communities. These communities might be distinguished by their different use of the software. In identifying the different roles and communities (e.g. Community A of scientific software engineering group would use the software differently in comparison to community B of a collaborating project), the PCF has acknowledged how the theory of Community of Practice (Wenger *et al.,* 2009) deals with structured organisations. The Community of Practice recognises that within an organisation the value does not inevitably lie with the individual members of a community of practice, it also recognises that benefits to an organisation such as an area of problem solving, an improvement to the quality of decisions, and more perspectives of a problem that can be accumulated for the organisation itself. The work by Lesser and Storck (2001) further discusses how communities are a successful means of managing unstructured problems. They are also an effective way to share knowledge external to the traditional structural boundaries, and they provide a channel for developing and maintaining long-term organisational memory. The PCF intends to attach itself to this concept set out by the Communities of Practice; hence, it utilises the benefits to aid in the development and evolution of the Project Team, the software, and the Community. This action is to provide the Project Team with the necessary feedback in the PCF. This problem is further taken into account in Step IV of the PCF, where the reflection of the process is reviewed from more than one perspective (See section 7.6).

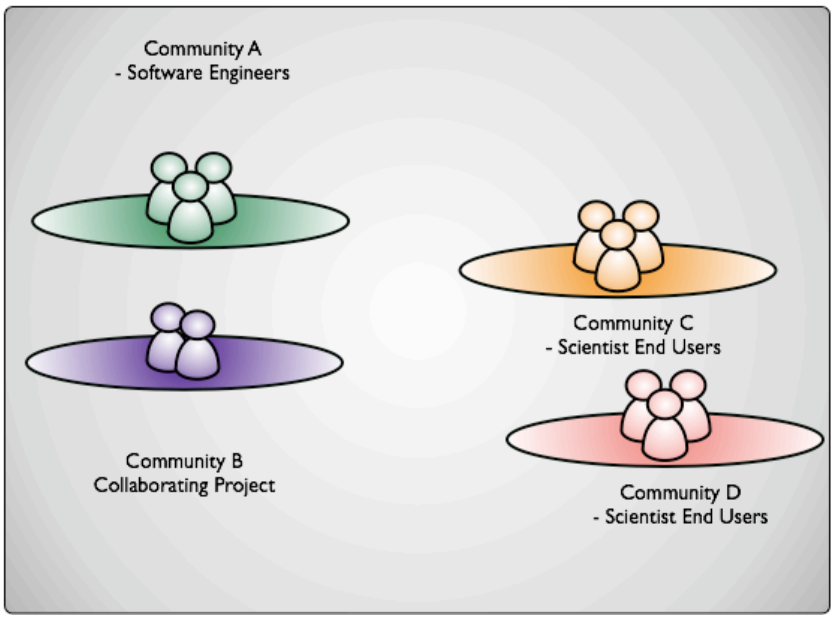

Figure 7.5: Separate communities (Adapted from Fong *et al.,* 2007)

In defining the Project Community, the depth of the roles continues to be re-evaluated through the iteration of the Project Community framework over time. An initial suggested depth of 2–3 levels would be a recommended starting point; this may be altered according to the use of the framework and requirements. Figure 7.6 illustrates the potential depth of defining the Project Community and community involvement. The centre of the model represents the project while each layer outside of this represents a level away from the active participation in the project. The example of this would be an active member who may regularly provide feedback or contribute to the project, or a peripheral member who rarely uses the software or provides feedback to the Project Team. The figure has been adapted for the purpose of the PCF to place the Project Team at its core. This is so the principle for understanding the depth of the role can be adjusted accordingly.

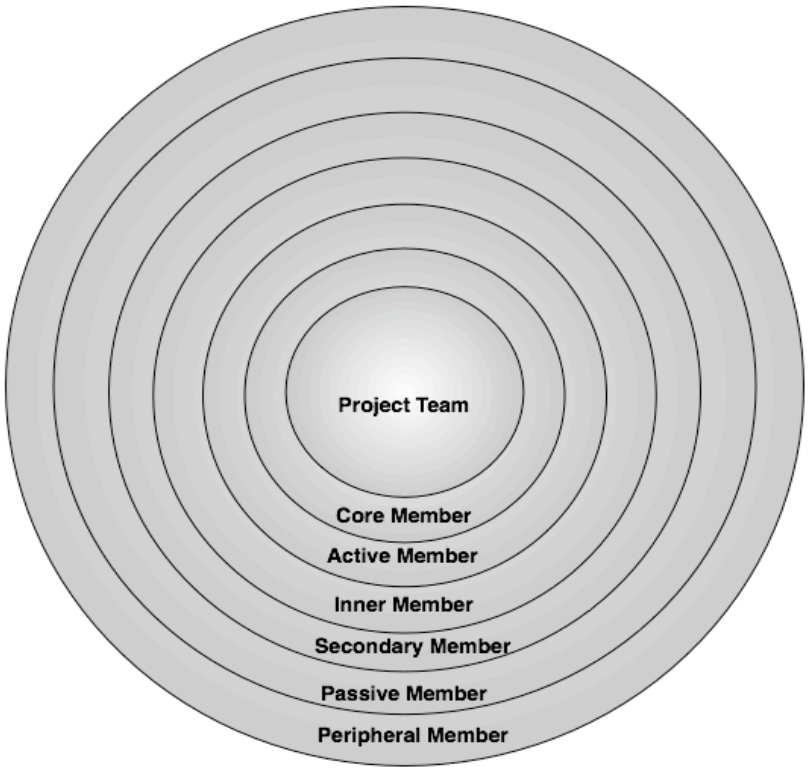

Figure 7.6: The Onion Model of the Project Community

(Adapted from Antikainen *et al.,* 2007)

The purpose of new community growth is to identify the growth and the evolution of the community. Figure 7.7 identifies the three major phases of a project, which have applied three principles from the work by Schön (1973), in which he discusses the role of the development of innovation. The same principle has consequently been applied to the information of a Project Team and how it is diffused in the community. The three points below describe this:

- The information and evolution of a Project Team remain central to the project. This occurs in the initial phase of the SSD project work. This initial phase is shown in Figure 7.7.A.
- The second phase for the information and evolution of a Project Team is where there is a movement of information from the Project Team to its users in the community. This can be via various points of contact (e.g. the project web site, documentation). This second phase is shown in Figure 7.7.B.
- The third step shows how there is widely distributed dissemination of information via a centrally managed process for the Project Team (this identifies the long-term

goal and purpose of the PCF). The range of development may cover new sites of development of SSD, UCD, or community uptake programmes such as new training sites for the scientific software. This phase is shown in Figure 7.7.C.

The final Figure 7.7.C, as discussed by Schön (1973), retains the centre of what translates and is represented in this research as the Project Team but adds the role of secondary centres that are actively engaged in the diffusion of information. These secondary centres represent the flow of information from the Project Team into the wider community.

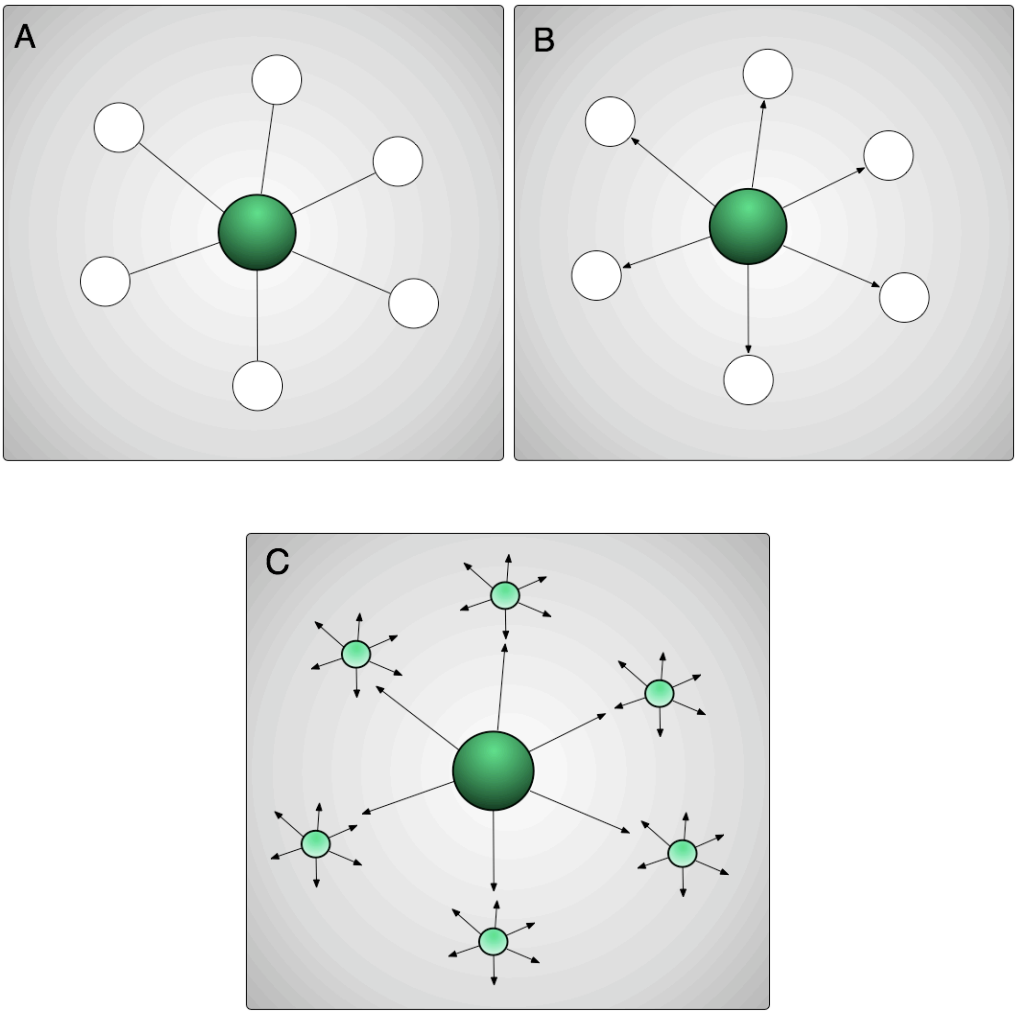

Figure 7.7: New community growth (Adapted from Schön, 1973)

7.3.7 The Project Community variable conditions

The Project Community can evolve over time, depending on the success of the project and the sustainability of project funding. The growth of the community, in turn, is central to uptake of the software.

- How is the Project Community contributing to the scientific community (application of software to science)?
- Do new communities emerge from the collaborations of the Project Team?

7.3.8 Project Community Mediating Role

The Project Community Mediating role is important for forming a community and championing the PCF. This requires careful consideration of all the perspectives of the SSD, the role of UCD, and the uptake of the software and the SSD project. Therefore, this role is critical to the PCF linking the principles of SSD, UCD, and the SSD community. The support for these perspectives may come collectively from different types of members and roles from within the Project Team. Based on this, the PCF has defined the following set of preliminary questions for a mediating role to aid in supporting the principles. The preliminary scope of questions to ask is:

- Which users should be targeted first? (This target may be aligned with the Project goals)
- Is the software for the whole laboratory, collaborating laboratories, or a full university science division?
- Is it the software use forced or optional?
- What are the barriers to entry in the laboratory, collaborating laboratories, or a full university science division?
- How can the software be improved to make the early adopters of the software happy?

Figure 7.8 illustrates the nature of the Project Community Mediating role in integrating the Project Team and the surrounding communities and how it may connect the external users with the Project Team. The example in Figure 7.8 shows that for each community labelled A to D, there exists a Project Community Mediator, which is colour-coded to that particular community in Figure 7.8. In addition to this, there also exists a more central mediating role (coloured in blue in the diagram) to whom the other Project Community mediators can provide information. A Project Community Mediator needs to cultivate, develop, and maintain an environment in which the components of the system can develop, grow, and evolve. The purpose of working within the PCF is to promote awareness, best practices, and information from the different perspectives of SSD, UCD, and community, to help to inform the systems design process. Finally, Figure 7.8 also aims to illustrate that there is no expectation that the Project Community Mediating role may lie with a single individual; the role is open to combination of supporting individuals within a Project Team. This is illustrated in Figure 7.8 where the expertise of the Project Community Mediator has been spread between four roles that directly interact with the communities of scientist end users, software engineers, and a collaborating project.

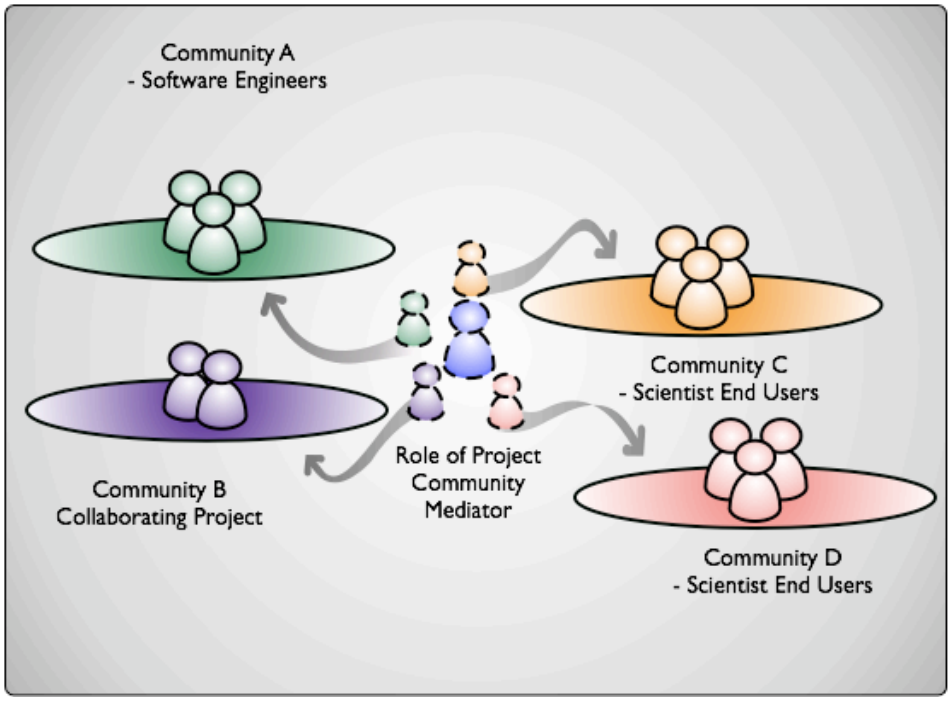

Figure 7.8: The Role of the Project Community Mediator

7.3.9 The Project Team and Project Community Limitations

The limitations on the Project Team and Project Community have been purposely categorised together because they share the common limitation of funding. Because of the instability of the funding cycle that the Project Team is dependent on, there are questions concerning how to learn, stabilise, and bring funding to the Project Team continually. Identified potential avenues to explore for this include joining forces with academic funding, although this brings unknown dependencies from the perspective of this research for a Project Team. Alternatively, another possibility involves finding other sources of non-academic funding.

7.3.10 The Project Community Outcomes

In defining the Project Community, the intention is to assess how the Project Team should reach out to the community. It is critical to state that the PCF is, firstly, a new way of thinking about and a philosophy for the creation and development of SSD. This has notably drawn on the agile philosophy, as well as on the openness that is UCD. Secondly, the motivation for the implementation of this philosophy is that it allows the PCF to be applicable to various SSD contexts. This is a key element in providing the desired flexibility of the framework for those who use it in various SSD project contexts.

In the process of researching out to the community to promote the uptake of the software, it is also required to take into account the limitation of resources within a project, recognising that the number of promotional led roles in scientific software projects is negligible, and recognising that such a promotional role would be considered a luxury in many SSD projects. Nevertheless, some individuals within a SSD project must take on such promotional work. This is an example of the responsibilities of software engineers have taken on board, as they must work beyond their own job description. A suggested list of practical techniques to promote the uptake of the software through the Project Community is as follows:

- Through web 2.0 tools
- Community events (e.g. conferences)
- Internal visits
- Developer scientist demo days

7.3.11 Tools of the PCF

The aim of setting out project management and software development tools is to provide a SSD project with the necessary means for managing the Project information. The tools can also be used for the wider level of information presented to the members of the Project Community e.g. the project web site. In the context of the Project Community, the combination of these tools serves the purpose of coupling with the central knowledge repository identified and discussed in Step II. The initial core tools outlined for this process are described in Table 7.2:

Table 7.2: Core tools

Tool	**Description**
A version control system	Management of changes to documents/source code
A bug tracking system	Management of bugs across the source code
A developer mailing list/forum	Management of community questions/support
A project web site	Management of community access

7.3.12 PCF Tools Variable Conditions

The variable conditions that can shape the selection of tools are again related to project funding. In an academic funding project, tool selection can be constrained by costs. Some tools cover all of the above (shown in table 7.2) but also provide additional features and functionality, yet they come at a financial cost, which can be very expensive. However such tools can be beyond the budget for an academic SSD project. Therefore, OS tools or tools allowing free use for OS projects are recommended. If more funding is available for an SSD project, tool selection can be less constrained by costs so there can be more to choose from and more benefits gained from the tools. In such a scenario, one institute's technical

department may provide and support alternative commercial tools so can offer a more complete tool solution. Alternative commercial tools would come at an additional cost for the SSD project, but can provide software support and predefined software development workflow processes to integrate with other project management and software development tools.

7.3.13 PCF Tools' Limitations

The selection of the tools must be able to fit the workflow for the project. This may mean a trial of a range of possible tools. It must be stressed that this is vital for a Project Team. To take the first set of tools without the members of the Project Team having used and experienced it adequately is ill-advised. A tools' use must be expected to expire through the growth of the project because it may not be able to cope with increasing demands (i.e. a tool that manages the SSD of a small group of developers will, over time, acquire more data as the size of the software grows and if more roles are added to the Project Team). The selection of the tools must also be made in relation to the resources of the project. A drawback of taking a set of tools is that a person in the Project Team is required to be responsible for the management and upkeep of the set of tools.

7.4 Step II: The storage of the Project Community information

The aim of Step II is the creation of a centralised repository for the storage of the Project Community information but it also aims to provide a way to couple and connect information about the Project Community. The importance of Step II for the PCF is in sharing the knowledge acquired throughout the Project Team and to share this information to help support the wider Project Community.

The proposal for a centralised repository tool is to support the co-ordination of and actions within the Project Team, and to ensure that the information and knowledge are not isolated inside various individuals. The knowledge is inter-subjectively shared amongst the members of the Project Team and, can be where necessary, the wider Project Community.

The efficiency of a repository tool is therefore represented by its ability to provide information for the individuals of the Project Team in order to coordinate activities.

7.4.1 Step II Example

An example tool that may be used to support the needs of an SSD is a Wiki, which will allow the Project Team to form interconnected pages. The advantage of the chosen tool is that it works in unison with the set of tools set out in Step I and integrates all of them.

7.4.2 Step II Variable Conditions

The variable conditions of the selection of the software tools of Step I are applicable to the selection of the tool for Step II.

7.4.3 Step II Limitations

Again, the selection of the repository tool is based on a trial of the range of tools that offers all the members of the Project Team the opportunity to use and experience the tool that is ultimately selected. The selection of a centralised repository tool must be able to work in the workflow of the project, and it must be able to complement the existing set of four core tools: a version control system, a bug tracking system, a developer mailing list/forum, and a project web site described for Step I in Table 7.2.

A key issues with any centralised repository tool is that it must be used effectively to manage the information of the SSD Project, so that the Project Team benefits from having the information about and knowledge of the Project stored in such a way. This means that the chosen tool must be correctly managed and supported by the entire Project Team. A tool requires the full commitment and participation of all of those in the Project Team as well as a team member to administer the use of the tool.

TIP: The use of the repository tool may expire through the growth of the project. The selection will also be made in relation to the project resources. The drawback of taking a tool is that each tool means that a role in the Project Team must be allocated for someone to be responsible for the management and upkeep of the tool.

7.5 Step III: The process of UCD

Step III of the PCF returns to the Project Community Mediating role(s) and specifically the goals and agenda for UCD. The Project Community needs team member(s) who specialise in UCD. This is a critical component for any step towards implementing UCD in SSD practices. While no explicit software development methods are specified in this framework, no specific UCD methods are specified either as it is the responsibility of the UCD expert(s) working within SSD projects to make the appropriate decisions on such methods appropriate to their context and resources. This also underlines the principle described by Mayhew (1999) on the selection of a UCD approach that can be adopted based on project constraints. It is the intentional openness of the PCF at this Step with the selection of methods to allow for the flexibility of the PCF. This decision has also been informed on the experience from the research with the recognition that the feasibility of using ethnography due the large investment in time. It is clear to see not all SSD projects will have the resources to employ such a technique.

Step III provides the information for UCD in SSD to be integrated into the SSD process and to be applied in the PCF. This step acknowledges the existing role of usability engineering and the scope of UCD methods available to UCD expert(s), which allow them to make a selection appropriate to the SSD project in which they are involved.

7.5.1 UCD for SSD

This section is pivotal to this research. The insights obtained from both the fieldworks and the existing literature underline the perspective that SSD has a limited application of UCD in scientific software. This is not a criticism of the scientific software developers but rather an acknowledgement to why Step III for the PCF sets the agenda that scientific software projects must always work with a specialised UCD role(s) in the project. In an SSD project, it is recommended by this research that UCD is applied from the very start of the project and that this is enforced by the SSD project funding bodies. The inclusion of this is to support the UCD Project Community mediating role. This point has purposefully drawn from my own experiences in the fieldwork and presence of working alongside the OMERO

team, and the existing literature by Mayhew (1999) and Battle (2005) and the description of the role of a 'change agent'. In reaction to this recognised problem in how UCD has to be adopted throughout the organisation (described in Chapter 2 section 2.7), the PCF recommends that a UCD champion role is employed from the very start to provide a constant level of commitment to UCD, to be a source of UCD information, and to create and utilise the opportunities for process change within the PCF. Any UCD champion role would work closely with the Project Community Mediating role.

7.5.2 Variable Conditions of UCD

A variable condition of the application of UCD methods is with the level of experience in using and applying the range of them. The range of UCD methods used relies on the appropriation of how best to integrate the phases through the workflow of the project. A further variation recognised for the PCF is how the process of UCD can be introduced after the start of the SSD project. In this scenario, the move to working as close as possible to the software design decision becomes the central priority for the role of UCD. The UCD process may then have a stronger influence on and a closer integration with the SSD process rather than remaining in an external position to the development of the software.

7.5.3 Limitations of UCD

The range of UCD methods can require a UCD specialist to maximise the information that can be collected and processed for the development process. This highlights the demands on the project resources that such an approach can make so that it may or may not be feasible.

7.6 Step IV: The Project Community action and reflection

Step IV for the PCF is an essential step to allow the evolution of the Project Community over time. The basis of this step is separated into two phases: the first is Reflecting-in-Action and the second is Reflecting-on-Action (see Box 7.2).

Reflection-in-Action

Reflection-in-action is 'thinking on our feet'. It involves looking at the experiences gained throughout the project and testing out ideas in use. This demands the continual building of new understandings to inform the actions in the situations that are unfolding through the Project Team and the Project Community. In order to move towards improvements in understanding about problems that occur in the Project, Reflection-in-Action calls for testing out 'leading ideas' to allow for the development of further responses and moves. Significantly, this does not demand strict adherence to established ideas and techniques. The Project Community recognises that issues with SSD have to be thought through as each case has its own characteristics.

Reflection-on-Action

This is the outcome of Reflection-in-Action after the event; it is the act of actively thinking back on what has been done in order to develop insights and discover how Reflection-in-Action has contributed to the positive or negative outcome (Schön, 1983).

Box 7.2: Defining reflection practice

Step IV for the PCF aims to maximise and integrate the knowledge from all levels and fields through the Project Community, with the goal of drawing together an awareness for the SSD, UCD, and community led challenges. The step concludes with the Project Team setting out a strategy that may continue to evolve through a learning process involving skills, experience, and insights gained through the dynamic interplay between formulation, implementation, and critical reflection. The strategy is based on learning from the context, conversing from within the Project Team, the Project Community, and being influenced by the reflections on the actions taken in the situation. The details of Step IV are now explained in the following sections.

7.6.1 Phase One: Reflection-in-Action

The first phase of this step uses the process of Reflection-in-Action (See Box 7.2). The act of Reflecting-in-Action enables the member(s) of the Project Team to spend time on understanding the actions of the Project Team by exploring these as they occur in the SSD project. This process is highlighted in Figure 7.9.

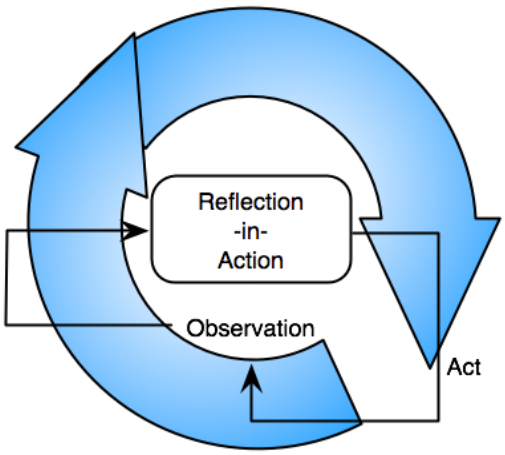

Figure 7.9: The Reflection-in-Action cycle within the Project Team

The Reflection-in-Action process needs to be able to accommodate and to be communicated between the areas of SSD, the development of the UCD, and the development of the community for the software.

In order to gain a wider perspective of the experiences gained through the project, the Reflection-in-Action phase promotes the use of collaborative roles through SSD. This phase aims to build new understandings and give insights to inform the Project Team about its goals.

7.6.2 Phase Two: Reflection-on-Action

The Reflection-on-Action phase is central to the efforts in this area for all involved within the Project Team (see Figure 7.10). This second phase builds on the outcome of the

Reflection-in-Action phase of building new understandings and giving insights to inform the goals of the Project Team, and it aims to enable the members of the Project Team to act on the information.

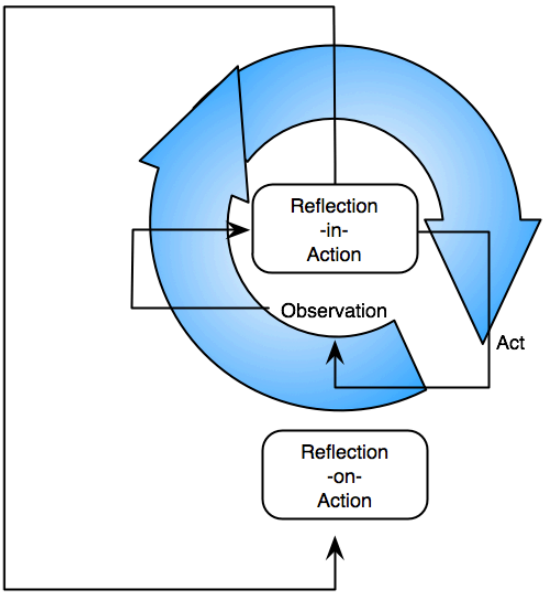

Figure 7.10 The Reflection-on-Action cycle within the Project Team

Both the individuals and the Project Team can form a collective learning process from the context through "conversing" with it and being influenced by their reflections on the actions taken in the situation (Schön, 1983). A view of this process is presented in Figure 7.10. Enabling an environment for reflective practice in the Project Team can provide a platform for the creation of a strategy. Conversing and reflecting on the actions taken can help to form the construction and connection of the strategy. This allows a strategy to be formed on the range of problems from the aspects of identified related to SSD, UCD, and the community within the SSD project and the combination of holding a holistic perspective with these elements.

The underlying aim of Reflection-on-Action distinguishes itself from reflection on the areas of the SSD, UCD, and the community in regards to the design, development, and uptake of the software. If the Project Team does not have insights into any of these areas, then for the purpose of the Reflection-on-Action process it may be valuable for the Project Team to widen its perspective through this process to allow for the broader scope of discussion and feedback. For example, a UCD role may be adopting a different UCD method to gain a

broader or more specific type of feedback.

Table 7.3 identifies the initial scope of techniques that may be used for the Reflection-on-Action phase. Further techniques and methods may be identified and applied to the context of individual projects.

Table 7.3 Scope of Techniques to Enable Reflection-on-Action

SSD Techniques	UCD	Community Techniques
Development process	Active user involvement	User-Facing Improvement Initiatives
Requirements	Usability champion	User service and support
Design	Explicit and conscious design activities	Marketing Strategy
Re-factoring to improve design and code	Evolutionary systems development	Meeting and evolving to support new scientific techniques and communities
Testing		
Deployment		
Constant Communication between Project Core		

7.6.3 Step IV Example

An instance where Reflection-in-Action is applied through the software development process may be in the development of a new software feature. This is where the new feature scope for the software is relatively unknown for the areas of the SDD, UCD, and the community implementation. The Reflection-in-Action phase, as explained in Box 7.2, is the Project Team's involvement at looking at the experiences gained throughout the project and the testing of ideas in use to manage this process.

The Reflection-on-Action phase is then used at the end of the development or release cycle, where the Project Team may actively discuss the implications of the cycle. The focus covers the interaction between the SDD, UCD, and the community. The core outcome for the reflection at the development level and through the SSD is consciously undertaken to learn about the actions taken within the SSD project. This is illustrated in Figure 7.11. Whereas previously the focus remained on the act and the observation in Reflection-In-

Action (Figure 7.10), the new wider focus in Figure 7.11 extends to a more holistic and reflective view of the actions and observations taken from the Reflection-In-Action. The goal for this is a continuing learning process from each aspect of the SSD, UCD, and the community of the SSD project.

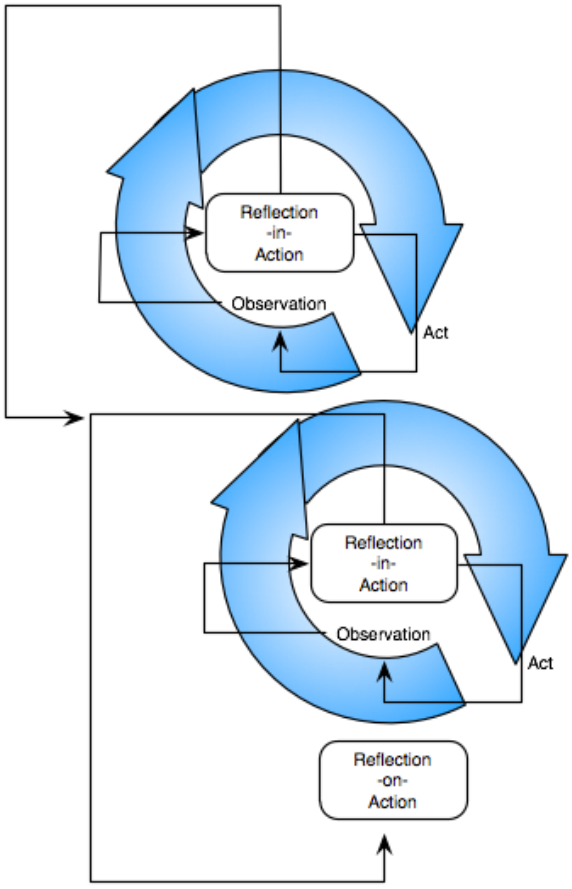

Figure 7.11: Reflection-in-Action as a continued learning process

Reflection-on-Action can also highlight many principles and good practices. Then, a Project Team can adopt and continue to evolve its priorities and strategies for the Project Team and Project Community based on its learning from the Reflection-on-Action process. The growth for the development of the strategic thinking that reflective practice presents is illustrated in the work by Dubberly *et al.,* (2009). This work outlines the interactions in a 1-to-1 balanced system, as shown in Figure 7.12. This highlights the stability between the linked entities. Examples could be an internal interaction of the Project Team or an external interaction between the Project Team and the Project Community.

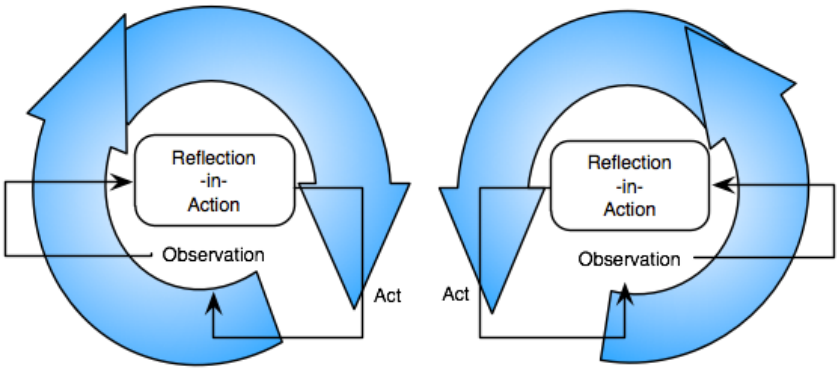

Figure 7.12: Reflection-in-Action as a Balanced Process

This next phase is further enhanced by the move to a fully conversing system, as shown in Figure 7.13. In this figure, the output of one learning system becomes the input for another one. This is in the context of either operating internally within the Project Team or in the interaction between the Project Team and the Project Community. Therefore, the Project Team continues to learn from the previous iteration of work, and additionally the Project Team can continue to learn from the feedback iterations from the Project Community. Each then has the choice to respond to the other or not.

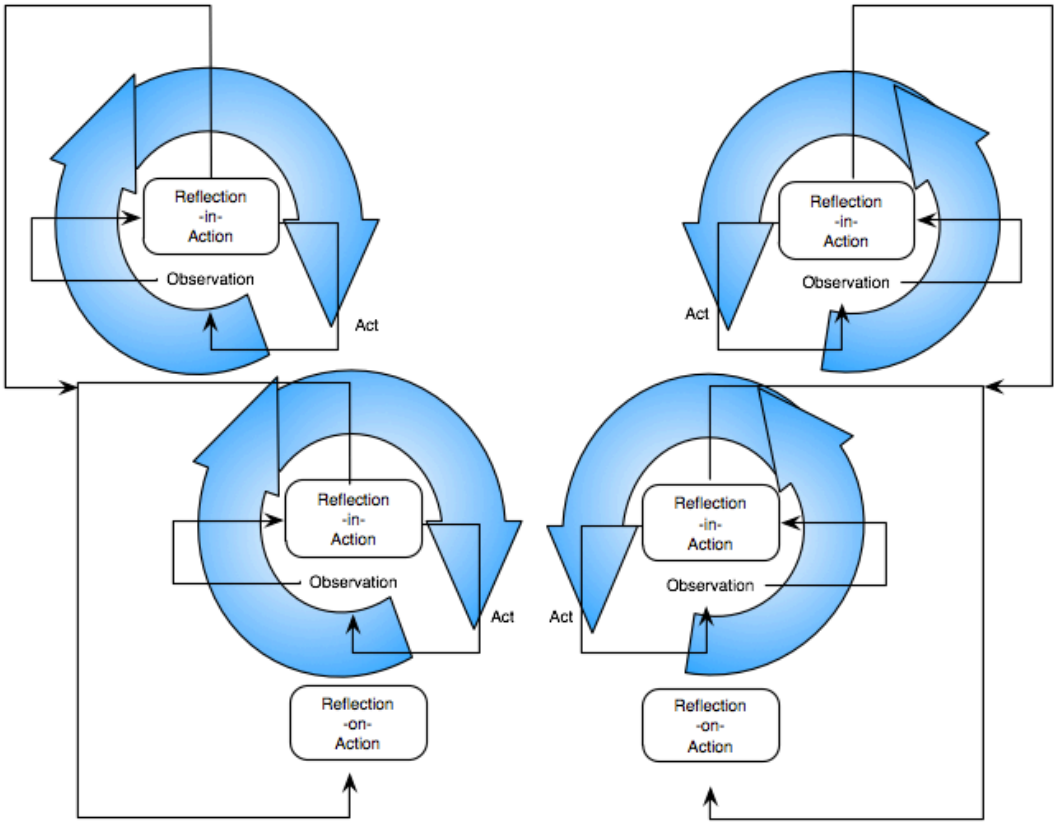

Figure 7.13: Reflection-in-Action as a conversing process

7.6.4 Step IV Variable Conditions

The process of Reflection-in-Action is sustained for a variable amount of time. This variable is based on the complexity of the development task in question. A recognised variable condition for the application and use of both Reflection-in-Action and Reflection-on-Action is the degree of how well the Project Team is working together. If the Project Team is relatively new, then it can take the team some time to establish a team's understanding of how other team members work can affect the Project Team and how it may react to creating and form new ideas. To compensate for this and allow a Project Team to become more established, the recommendation would be to maximise the use of the Reflection-on-Action process. This second phase should allow the Project Team to highlight and discuss what parts of the process of the project is working for them as a team or what parts of the process are not and need to be changed.

The variation for Reflection-on-Action is subjected to the time the Project Team invests in the process. The PCF acknowledges the time constraints of working within the SSD context, but the importance for working through the step of Reflection-on-Action and on the three key areas of the SSD, the UCD, and community of the SSD project are necessary for the sustainability of the SSD project.

7.6.5 Step IV Limitations

The limitation of Step IV is that not everyone in the Project Team will be at the same level of participation for reflection on the project work. The proposal is in that case to encourage the exploration of alternative techniques by a Project Team that facilitate both Reflection-in-Action and Reflection-on-Action. Such an example might be through using Project Team meetings to encourage wider team feedback on the Reflection-in-Action and Reflection-on-Action processes.

7.7 Summary

Chapter 7 has put forward the proposal and definition of the Project Community Framework (PCF). This is my own personal response to the fieldwork from the two pieces of fieldwork analysed in chapters 5 and 6. The PCF emerged to address the need for improving the uptake of UCD philosophies, methods, and thinking in SSD. To help this, the PCF has proposed and consequently put forward a new way of thinking about SSD.

The PCF has been set out with four steps (Steps I to IV), which were fully described in this chapter. To briefly summarise, these steps capture the key principles and aims of the PCF to accommodate for the software development tools used, the role of a UCD, and the continued reflection and action of information for a SSD project. The combination of these steps has been constructed to promote a new line of thinking for integrating SSD, UCD, and the development of the SSD Project Community. Chapter 8 now goes on to evaluate two existing features of scientific software development with the frame that the PCF has proposed.

Chapter 8: The Evaluation of the Project Community Framework

The Project Community Framework (PCF) was created from the analysis of two different fieldworks (studies 1 and 2, presented in Chapter 5 and Chapter 6 respectively). The PCF is a proposal that has put forward a new way of thinking about SSD. Therefore, because the PCF is a newly created framework, I will assess it against two evaluation reviews to understand the SSD practice that this research has been part of, so that the concept of Project Community can be evaluated against real SSD practice.

Review 1 is about HCS (the high content screening code mentioned in Chapter 6, which is a method that is used in biology and drug discovery to identify substances that alter the phenotype of a cell in a desired manner) and Review 2 covers the FLIM technique, which is a fluorescence microscopy technique used in imaging to map the spatial distribution of proteins lifetime and interactions. A summary of the evaluation approach used in the research is illustrated in Figure 8.1.

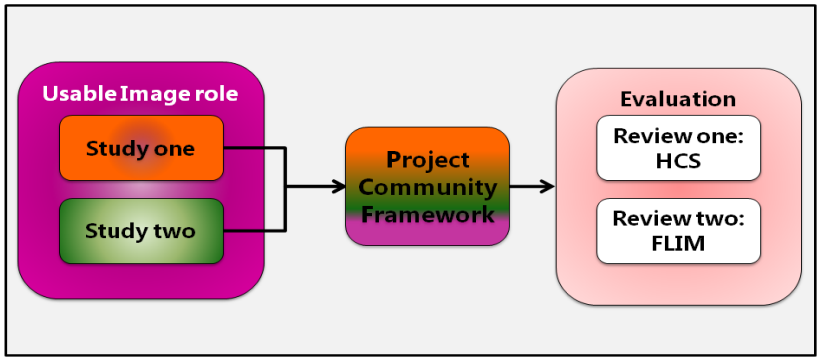

Figure 8.1: The approach for the PCF evaluation

For this research, because of the ethnographic method was already used (see chapter 4), I have also made the decision to evaluate the PCF within the ethnographic method. Thus, this research considers the four techniques discussed by Hughes *et al.*, (1994) for using ethnography in systems design. These four techniques are quick and dirty ethnography, concurrent ethnography, evaluative ethnography, and the re-examination of previous

studies. These are summarised briefly in the next section (8.1) with the selection for the evaluation of the PCF also explained in the next section. Section 8.2 of this Chapter accounts for the evaluation process of the PCF and the two evaluation reviews conducted for this research. The two reviews of HCS and FLIM for the evaluation are presented in sections 8.3 and 8.4 respectively. The chapter concludes with a wider discussion of the PCF and draws out the philosophy of this research in the PCF manifesto.

8.1 Evaluation strategy

The first technique, quick and dirty ethnography, is created to help designers generate a picture of the workplace over a short time frame. It aims to highlight the important factors of the workplace that are relevant to the design (Crabtree, 2003). The findings by Hughes *et al*. (1994) demonstrate that this is also suited to large-scale sites, based on their experience with an air traffic control system (Hughes *et al*., 1994). This quick and dirty approach may offer a restricted view but it helps in mapping out the interdependencies and activities of work, and it can be done in a short period of time, even though it is dependent on the size of the organisation (Crabtree, 2003).

The second technique, concurrent ethnography, is when ethnography takes place at the same time as systems development. In this instance, the designer and the ethnographer exchange the findings via regular communication at each phase of systems design. Hughes *et al*., (1994) describes this as a sequential process where the ethnography leads the design process. Crabtree (2003) also underlines how flexible this process is, as it iterates through the process of fieldwork > debriefing > prototype iteration > fieldwork, for as many times as required.

The third technique of evaluative ethnography involves a focused shorter period of time for the fieldwork. In using a focused period of time, it aims to gather the relevant information quickly rather than using alternative traditional long-term method. The general aim of the technique is to establish the practicability of a proposed design system and the process involves a sanity check of the design proposal. The evaluation is not designed to be

exhaustive, but serves to assess the feasibility of the proposal and to draw out any problematic issues (Crabtree, 2003).

The final technique, the re-examination of previous studies, is where existing studies are reassessed to inform the initial design thinking. Its goal is to gather existing sources of information that can be used to help to sensitise designers to the issues of the workplace (Crabtree, 2003). The work by Crabtree (2003) also recommends the re-examination of issues as they emerge, to further inform the design. He also states that although the technique is a satisfying activity in terms of examining existing work, oversights are made and parts have to be disregarded of the existing work as out of scope.

These four techniques are very much complementary and overlap given the requirements of the design process. As they do not require prolong periods of contact, these techniques are rather flexible. However, this conflicts with the ethnography core principle of prolonged contact, where the goal is to develop awareness in the design over time so it integrates social science methods in different ways (Crabtree, 2003). This change of principle regarding ethnography is therefore evident in my own research, where ethnography has been applied in traditional prolonged contact for the method documented in Chapter 4, yet for my evaluation this is not required as I am searching to question the practicality of the PCF in a SSD working context. Thus, there is no need for an extended period of contact in it.

Hughes *et al*., (1994) categorises six features of the different types of ethnography in design. These six factors are the detail of work, type of design information, duration of study, influence of field site, design/study relation, and form of study. For making my decision for my evaluation, the points accounted for are the type of design information, the design/study relation, and to a lesser extent the duration of the study. The selection and consideration of the factor of design information were focused on motivation and scope of design in relation to the PCF. The selection of the design/study relation was driven by the need to outline the design of the PCF in the evaluation review. Based on these two aspects, and considering that no further fieldwork will need to be performed so meant that there was no extension to the study, I have decided to use the ethnographic technique of re-

assessment of previous studies for the evaluation review, the existing studies which will be taken from a selection of collected OMERO meetings notes. I have made this decision based on the ease of access to this material and from my previous experience with the material in the research with analysis made in Chapter 6.

It is important to note that a wide range of notes were collected throughout this research, but this new selection of notes were collected before the notes that were used for the analysis PCF in Chapter 6. The meeting notes for the analysis covered a period from 04.08.2008 to 21.10.2009. I did purposely wish to examine the PCF against the historical working practices of the OMERO project to gain an insight into the feasibility of the PCF. Extract 14 (later in this Chapter) is the exception to this time period as additional work for Study 2 was observed during my embedded time in the OMERO project and I personally wished to explore this after the analysis presented in Chapter 6. Again, this evaluation approach aims to review the PCF and check how these meetings notes compare to it even though they have been carried out independently from the creation of the PCF. The evaluation work has been formed to review the SSD practice that this research has been part of, so that the concept of Project Community can be evaluated against real SSD practice. The following elaborates on the re-assessment of the previous studies for my own evaluation.

8.2 Re-examination of studies

This section explains how the two evaluation Reviews were selected and how the evaluation was carried out**.** My evaluation approach for the re-examination of studies has drawn on work by Hughes *et al*., (1993) and Hughes *et al*., (1994). This approach was chosen to evaluate the PCF against an applicable domain. As mentioned in section 8.1, I have selected a different set of OMERO meeting notes to use for my evaluation. This was because there was a limited amount of ethnographic material for scientific software development with UCD in the literature (See Chapter 2 section 2.8). So using the OMERO meeting notes it serves to provide a wider range of SSD information. However, I also faced the problem of drawing out the ‘implications for design’. Hughes *et al*., (1994), explains this problem in how that not all ethnographic studies offer clear design objectives. This

comes from the fact that an ethnographic researcher will have his/her own objectives for the study. A further problem described by Hughes *et al*., (1994) is that ethnography, as a social science method, fails to readily produce a body of findings to underpin any application of information. In accommodating these factors in my own research, I have noted that my own objectives would concern looking at the implications for design for the PCF based on the two topics of High Content Screening (HCS) and Fluorescence Lifetime Imaging Microscopy **(**FLIM). Because I could not be sure of the outcomes of the analysis of this new material, if I were to follow the steps of the analysis discussed in Chapter 4.

I have adapted my evaluation process from the method described in Chapter 4 in the following ways of the selection process and steps taken. These changes were made as my requirement for the evaluation of the PCF was in sensitising the PCF to a wider set of practical SSD information. The meeting notes were selected using the keyword terms of HCS and FLIM. This meant a total number of 14 meeting notes were reviewed. The selection of these terms is explained in the paragraph below and the entire meeting notes used in the evaluation are available in Appendix 12.

For each of the two evaluation reviews, I have then directly commented on the actions taken in the OMERO meeting and used this to expose the PCF against the actions being taken in the SSD meetings, using my own narrative and, when possible, further discussions with the OMERO developers to help this evaluation review. The two reviews chosen were HCS and FLIM. The first, HCS, was selected based on my own experience and on the output from the analysis in Chapter 6. The HCS code was part of the OMERO components theme, so it provided me with an area of specific OMERO functionality to take for the focus of the evaluation for Review one. Yvan, one of the OMERO developers, helped me with my decision for the second Review topic, which was based on my remaining codes in the OMERO components theme. The main code under discussion was the ‘FS’ (file system) code; however, Yvan suggested that given the HCS code and its focus on functionality throughout the OMERO software, the topic of FLIM would also maximise the evaluation. I do recognise that introducing the input from Yvan can bring an element of bias into the evaluation. The full implications of this shall be discussed in Chapter 9 along with the further scope for improving the evaluation of the PCF (Section 9.4). It is also

acknowledged that because of the use of Extract 14 there is conflict of previous analysis with the OMERO embedded work meetings that I analysed in Chapter 6. Again the implications of this shall be reviewed in Chapter 9 section 9.4.

8.3 Review 1: High Content Screening (HCS)

8.3.1 Background

HCS is an automated cell biology method drawing on various aspects of optics, chemistry, biology, and image analysis, allowing biological research and drug discovery. Further information on HCS and SPW are in the footnotes 4 and 5 in Chapter 6. The HCS technique was not originally planned in the original development of the OME data model. With the growth of the drug discovery process as highlighted by Abraham *et al.*, (2004), HCS has been an approach allowing for easing the bottleneck of drug discovery. However, the HCS process was unable to predict the challenge of managing the amount of data that was now possible to acquire with the HCS technique.

8.3.2 HCS fieldwork

The fieldwork has been conducted through the OMERO team meetings and OMERO conference calls, where the relevant transcripts to the HCS have been used as the main source of data. The researcher's own narrative and informal interviews and discussions with the OMERO developers support this when it is feasible.

8.3.3 The timeline of the HCS fieldwork

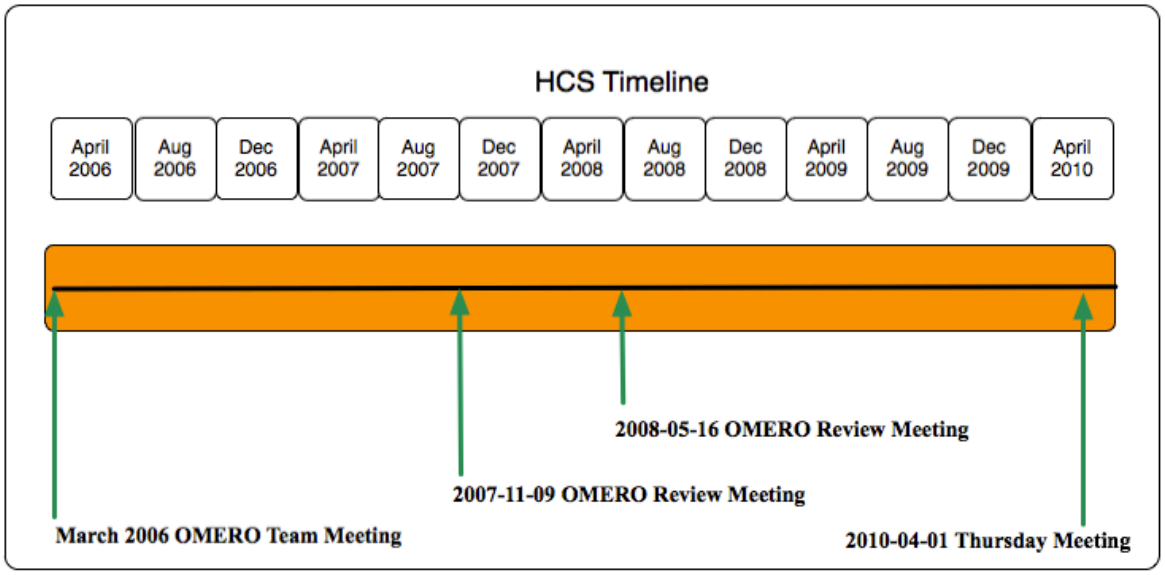

Figure 8.2: Key timeline of development events for HCS

The timeline for developing the HCS features support was created early on in the OMERO software development as a priority for the team to support the community (See Figure 8.2). A major dependency through the HCS work was interaction with the commercial entities for the HCS file format. This dependency had the ability to block the entire process. The following Extract 1 highlights the priority for the software development process of creating a test suite.

> *Importer*
> *Test suite*
> *This must be top priority!!*
> *Incell Screen-plate-well support*
> *Need to have the new XML model supported before we can get this done (this is coming from Terry)*
> *Eli can start looking at "the basics" fairly soon on this*
> *Miles: working on Evotech data and will have it delivered to Eli.*
> *Adding file formats (Ugh!) (We are agreed to slowing down this process, in favour of getting the testing framework implemented)*

> *Miles: HCS file formats (Should we start exercising parts of the SPW data model; we have an excellent use case here with the Incell 1000 files. Obviously, we won't support the SPW data in the client yet, but can consider this after the new year. Bio-Formats supports Evotech, shall we add Incell 1000.*
> *(Also note I have contacted Cellomics twice, no response.)*
> **Extract 1 -** 2007-11-09 OMERO Review Meeting (OMERO 2010)

The OMERO project is operating alongside the Bio-Formats project; the information exchanged between these creates dependencies throughout the development process. The OMERO project must have the technical ability to read the proprietary file formats, which is the main focus of the Bio-Formats project. The information on reading the proprietary file formats of HCS were critical to the development process. With the OMERO project and the Bio-Formats project being located in the UK and USA respectively, the OMERO development team uses a wide set of tools in order to maintain communication, such as instant messenger, email, Skype, and face-to-face group meetings four to five times a year. A further dependency that both the OMERO and Bio-Formats project rely on is shown in the communication with commercial entities, in regards to gathering the required information about a HCS proprietary file format.

This scenario raises potential dependencies for the OMERO project, as the community may ask to read the HCS format X that is owned by commercial company Y. Any delay in communication with company Y is of harm to the project and to the wider Project Community itself. If not properly managed, this ultimately could have a knock-on effect for the success of the OMERO project. This is a situation where the software development process must be able to manage and handle the spectrum of interaction and communication in which it is involved.

A further dependency created by the technical challenge of the HCS project was the size of the data. The typical file size was over 10 Gigabytes; a typical file size handled by OMERO until then was generally 1 to 2 Gigabytes. Such an increase made the technical developments of testing the data more difficult. Extract 2 demonstrates the nature of the technical challenge the OMERO team faced and extends further to the testing dependency.

The context in Extract 2 was at the end of the development cycle where the project is feature-frozen and both general testing and user testing have started with the OMERO team assessing the remaining technical challenge of testing large datasets.

Bio-Formats

HCS. Coming. More data?

Just SO big (+10Gig, one plate). Generate a smaller one?

Dan needs a solution, possible source of data.

Action: Jack sends an email.

Action: Steve gets more than one plate (3 plates?) from Finn and uploads it.

Notice: some of the datasets have 100K+ files, each one huge.

Lots of issues

-Tricky testing on real data

- Other format requests

- More time fielding emails

Extract 2 - 2008-05-16 OMERO Review Meeting (OMERO 2010)

Throughout the HCS development work, there has been on-going clarification and communication about HCS development. One channel of communication was through the weekly development meetings. The following Extract 3 is about the preparation for the Beta 3.1 release of the OMERO software. The release is one of many elements considered as the project moves towards the release date of the software

HCS format

Yvan is planning on getting started once Beta3 is out.

Status from Eli & Steve?

Hasn't been integrated with importer

Flex is currently the only format. INCELL 1000 is coming. And then of course more after that.

Extract 3 - 2008-05-30 OMERO Review Meeting (OMERO 2010)

Extract 3 is part of the roadmap of development for the Beta 3.1 release. This includes the other areas of development work of an import history, 're-import' feature, search and tag improvements, maximum intensity projections in one of the OMERO clients, and a new scripting engine framework. The scope of the software evolution through the software in Review one is evident in the technical challenge of dealing with such a large amount of HCS data.

The release cycles of the software have had to deal with this in increments, and to notify the community as to where the restrictions lie. This is demonstrated in Extract 4, which shows how the project team has acknowledged the limited size of data to be imported, which may restrict software use. The benefit of this is that it provides awareness of possible limitations for the HCS community. This problem was subsequently solved in the next release of the software and is documented in meetings that were not examined in this research.

> *PLEASE NOTE: In our testing with OMERO Beta3 and Beta3.1, we have seen that importing many thousands of image files will cause OMERO.server to slow down, and possibly need a restart. See wiki: OmeroThrottling for more info. For the moment, please limit imports to no more than 1000 image files at once.*
> **Extract 4 -** (OMERO MilestoneArchive 2010)

In examining this issue with the OMERO software developers involved in HCS, the problems were caused when the sample data files were insufficient or not representative of how the data files may be used in the system (e.g. a small image data file is provided when the typical size is much larger). This was in some cases down to a conflict of interest of submitting example image data, as all those users supplying the sample image data files did not understand that it would be for their own benefit and that of the scientific community. The supplier of the data holds his/her own agenda but resolving the problem is not always an easy task for the Project Team. In the informal discussions with the members of the development team directly involved in the HCS development, the developers informed me that they were required to communicate and interact with the company responsible for creating the HCS data model. In these instances, the communications were with commercial developers. This, in the context of the project, is significant as commercial

development comes with a very different perspective from that of academic development. The agenda discussed in the context of the project is disregarded because the commercial entity may not typically be concerned with serving the elements of the wider context of the academic community. In my discussion with Yvan, one of the OMERO developers working with HCS, the further perspective was added about how communication with Rafael, a commercially based developer, was very helpful on the OMERO development forums. The agenda and roles for interaction with commercial entities cannot always be assumed. The further HCS development objectives were highlighted during the March 2009 OMERO team meeting. In the development timeline of the OMERO project, this meeting came several months into the development of the HCS.

During the evaluation of the HCS Review one, the development process was reacting and adjusting to problems, as they became better understood. The OMERO project was using an Agile approach and working in this way meant the Project Team made estimations for the proposed HCS development work in terms of how long a development task will take to complete. However, this also meant that the estimations of the development work were subject to any delays in the feedback and communication process for the HCS process. One of the techniques the OMERO team explored was the use of a specific agile SCRUM method of planning poker (Cohn, 2005). This technique involves all the team members presenting individual stories. Each team member then selects a numbered card based on how much work is involved in his or her story that has been discussed. The overall conclusion about this technique was that it was of minimal benefit to the team and the OMERO team has not used it again since.

The March 2009 OMERO team meeting highlighted two types of development for HCS – functional and infrastructure development. The meeting itself was significant as the project team was defining the work for the rest of 2009 after the release of Beta4.0. The infrastructure development is concerned with the development OMERO server. The functional development is focused on the improvements to the user interface of the software. Infrastructure development is key for further developments in functionality. However, to improve the software and increase end users of the software, developing features is significant but has to be done with a balance between functional and

infrastructure development. It is critical for long-term project sustainability that both elements are maintained. This importantly highlights how the formation and use of strategic thinking is a primary need within SSD and can benefit from the reflection of three equal factors (SSD, UCD, and the community), which is underlined in the Project Community Framework.

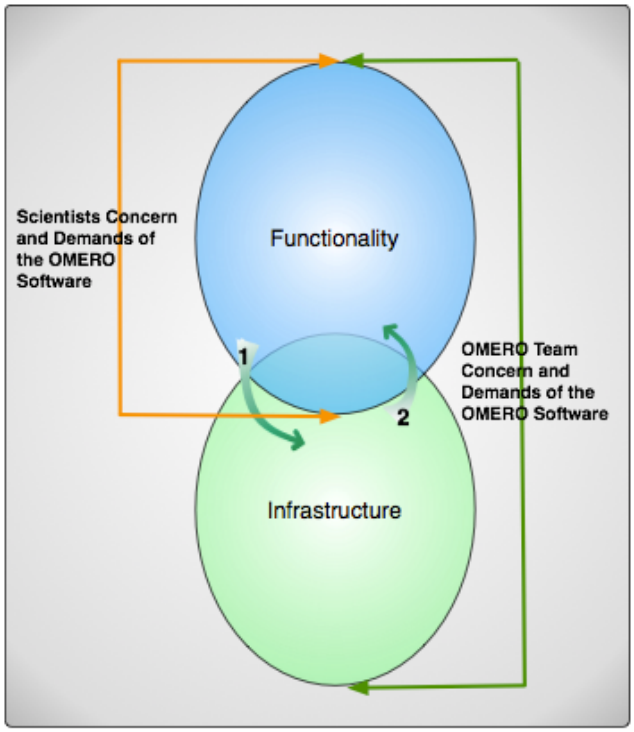

Figure 8.3: The infrastructure and functionality development conflict

Figure 8.3 is based on observed practice within the SSD environment, where there is a high demand to enhance functionality in order to stay updated with evolving scientific techniques. This is why Figure 8.3 represents the scientists' central concern with the functionality of the software. However, this is not necessarily always so, as scientists may primarily want the speed of the system to be improved at some point or deem data handling as being more representative of infrastructural development of the software. In the areas of development for HCS, the compromise between infrastructure and functionality was important. The concern was whether infrastructural development for HCS would allow more HCS file formats to be imported into the OMERO software, which would widen the scope of the HCS community and the use of the software. Even so, the HCS functionality

development provides the ability to view the large number of images associated with HCS data. The technical details of these are shown in Table 8.1.

Table 8.1 Infrastructure and functionality identified for HCS development (OMERO 2010)

Item	Functionality or Infrastructure	Known use cases	Comments
HCS viewer	Functionality	Database/Model; Multiple	Adjustment to data model and database to offer a useful plate viewer.
File formats	Infrastructure	Multiple	HCS file formats possibly.

8.3.4 The HCS developer review discussion

The HCS development process was evaluated with the software engineers involved; Figure 8.4 highlights particular aspects of the HCS process. The HCS process involves the external software calculating the result that the end users might wish to use in the OMERO software/platform, by storing the images and the information (metadata) associated.

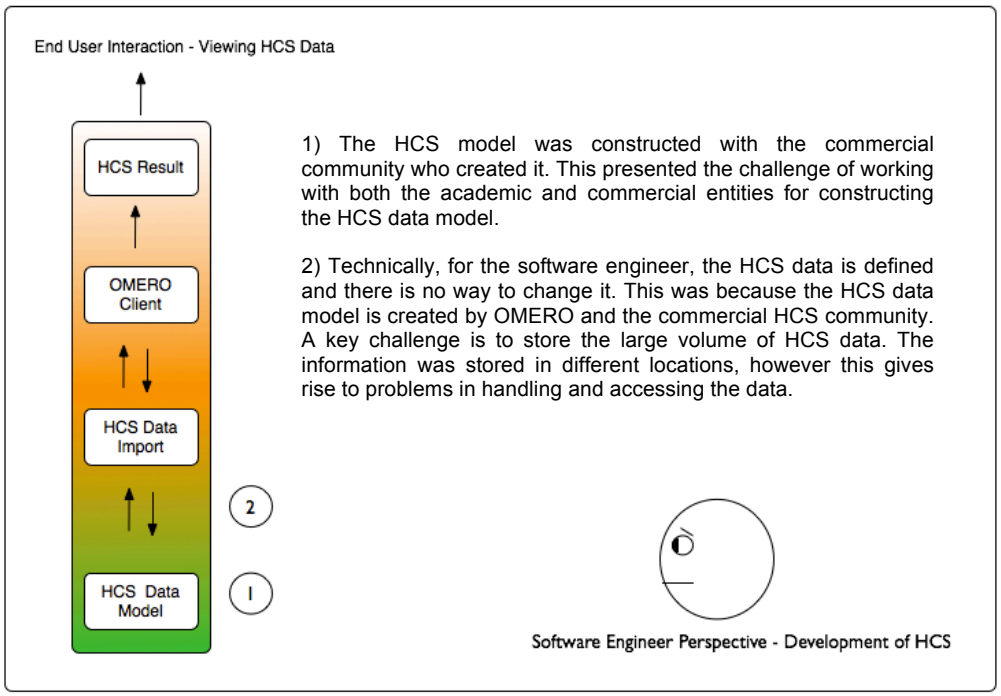

Figure 8.4: OMERO scientific software development view of HCS

The OMERO project was handling a requirement from the HCS community, as the community required a solution on how to handle the large amount of image HCS data. The OMERO project created the data model for the HCS data; the HCS community represented both academic and commercial entities. The summary of challenges is outlined in Figure 8.4.

Technically, for the software engineer, the HCS data is defined and there is no way to change the data. This was because the HCS data model was created by OMERO and with the commercial HCS community. A key challenge was to store the large volume of HCS information. Information, which was stored in different locations; however, this gave rise to problems in handing and accessing the data and data type.

The OMERO project had to redefine the HCS data model to meet the project's challenges (overcoming how the data was stored and the speed of accessing the data) and the HCS data model was adjusted to support these changes. The outcome of the HCS development process was the first release of the software that supported HCS in OMERO. Since this Review was documented for this research, further development has taken place to meet the growing HCS community that is now using OMERO.

8.3.5 Summary of HCS fieldwork

The process for developing and supporting the function of HCS process was problematic because of the need to provide a generic software solution. The project was striving to build and achieve a broad platform with community input, yet, the interaction with the community continued to prolong the development work, similar to the on-going technical challenges of the project.

The challenges occurred when the community was not always interested in what the project team was involved in or it may not have shared the short-term or long-term vision of the team. The term community becomes much more abstract for the OMERO team because of the conflicting interests between different entities making up a community. Through my interaction and discussions within the OMERO team, the conflict of interests of the

different types of people that make up the HCS community has been relevant and important for the development of HCS, as the process has spanned a wide range of end user types. So helping to provide a generic solution was beneficial for the project. The type of end users covers academic users of HCS, the commercial end users and creators of the HCS data model. The project's reaction to working with the spectrum of end users has been to adapt and adjust to the nature of the people involved and to communicate with them. However, this has highlighted how the Project Team manages and resolves potential conflict of the requirement requests that are made for HCS. The implication of this is on the Project team's time; consequently, the Project Team must make further decisions about the requirements and impact on connected areas of development.

8.3.6 Summary of the HCS fieldwork against the Project Community Framework

How Review 1 functions with the PCF is highlighted in the following discussion. Step I of the PCF combats a central challenge of HCS development: catering for the wide range of possible end user types, from academic users to commercial end users. Step I identifies the Project Community through the domain, community, and practice. This shows a framework to engage community members in activities and discussions and to share information. This also promotes relationship building so people from within the OMERO Project Team and Project Community may learn from each other. The variation of the type of users in the HCS community is positive for the PCF, as it is not restrictive in the identification of anyone who is in some way connected to the project. As well as this the Project Community also documents its expectations. However, as highlighted in Step I, in order to achieve this level of communication takes an investment in time and a level of sustained interaction with the Project Community from the Project Team. The tools for the PCF outlined in Step I of a version control system, bug tracking system, mailing list/forum, and project web site, the OMERO team already had an established set of these tools All the team used the tools, but Jack had the primary responsibility for administering and fixing of the tools if they happen to break down.

Step II of the PCF provides a central repository for the storage of the Project Community information and for the SSD process. In the HCS Review one, no one tool covers the

central purpose of having a centralised repository of information for the project. The tools that came closest to this was the bug tracking system, and the version control system that is used to record the development tasks. The version control system is judged to fit the way in OMERO project works, although, as highlighted throughout Step II, the success of a tool depends on its fit, adoption, and uptake by the Project Team.

Step III of the PCF supports the HCS data Review one. The HCS development process benefited from having access to valid sample files and this was a crucial element. This step may further support the example files by exposing the development process to a range of UCD techniques. The selection of the UCD techniques may be based on what is complementary to the existing development information.

The first phase of Step IV, Reflection-in-Action, has been limited because of a restricted amount information that Review 1 is able to provide. Therefore, the ability to gain sufficient insight through this was limited and no definitive perspective was gained from this step. The second phase of Step IV, Reflection-on-Action, draws on expanding and contributing to the challenges of working towards a generic platform. This second phase was in the way the HCS Review one had to redefine the HCS data model. So that it catered for the HCS community's challenges in dealing with HCS data. A general comment to come from this final step was the ability to identify the conflicts of interest between the entities of the PCF. This is for future development work of the Project Team. This, in turn, exposes the strategy that may be formed from the Reflection-on-Action process, which is key for building and evolving through a learning process, and connecting the skills, experience, and insights of the Project.

8.4 Review 2: New Community Fluorescence Lifetime Imaging Microscopy

8.4.1 Background

As already mentioned, FLIM is a fluorescence microscopy technique used in imaging to map the spatial distribution of proteins lifetime and interactions. As a new technique, FLIM

aims to provide scientists with the ability to analyse the images for experimental work such as Total Internal Reflection Fluorescence (TIRF) and Fluorescence Resonance Energy Transfer (FRET) analysis. The growth of the FLIM technique and its adoption into the OMERO project is also further evidence of the accommodating trends and new scientific techniques into the project.

8.4.2 Central elements

Again, the fieldwork analysis has been conducted through the OMERO team meetings and OMERO conference calls that focused on the FLIM development process, and these provided the main source of data. My own narrative, informal interviews, and discussions with the OMERO developers involved in the FLIM development process have supported this, when necessary when assessing the FLIM fieldwork meeting notes.

8.4.3 The timeline of the FLIM fieldwork

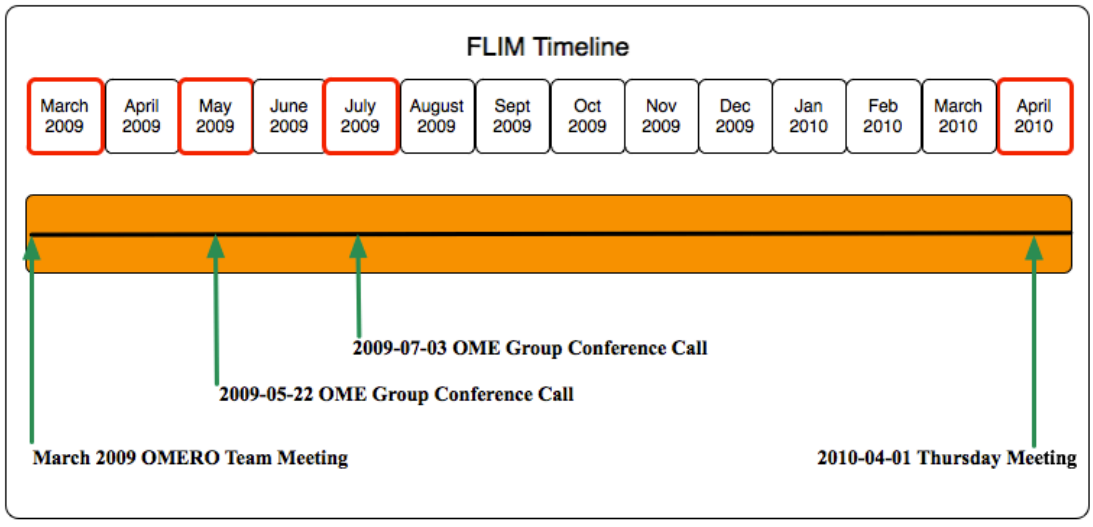

Figure 8.5 Key timeline of development events for FLIM

The FLIM development was documented from the March 2009 OMERO team Meeting (See Figure 8.5). As highlighted in the HCS Review one, this meeting aimed to address the

wider goals of the project for the next release of the software. Extract 5 highlights the focus for the project team, outlining the project team's work goals:

1. *To deliver these goals, we need to agree on a work plan. This will likely mean a very integrated work pattern, using the mini-group/iteration system*
2. *Most importantly, the natural tendency to focus on your own work, and hold functionality and commit later will have to be put aside*

Extract 5 - March 2009 OMERO Team Meeting (OMERO 2010)

The above extract identifies two points significant for the FLIM process that was developed from within the OMERO team. The first point concerns the team's development practice of implementing a mini-group system in the project team, which was a step away from the individual working practices. The mini-group process for the project team was organised into two groups: group A was responsible for the client side of the work and group B was responsible for the server side of the work. The Project Team, from working on this practice, also went on to adopt mini-group meetings in order to hold deeper detailed discussions about specific areas of development, with the developers more directly involved. The mini-group meeting allows further discussions outside the general meetings, so more time in the general meetings could then be spent on the wider issues relevant for the whole project team. The mini-groups and what the mini-group want have also been used to identify what they wish to get out of the community events.

The second point in extract 5 recognises the natural tendency of developers in the project to work independently. This is highlighted in the work by Weinberg (1971), where he discusses independent practice in development work. The discussion of independent working is significant for the OMERO team, which demonstrates its awareness of the potential drawbacks that can arise for the team from this. Dependency is the first element examined against the FLIM work; this is prominent throughout the FLIM development work. The dependencies within the project highlight the compromises throughout the development process. They are of two types: first, technical-led dependencies that occur within the project team, which are related to the software development; and secondly, the dependencies that extend to the interactions with the community.

The discussion in the FLIM meeting cited in Extract 6 was about the remaining work related to the region of interest (ROI). Every ROI needs a set of new analysis results. The dependency for the continuing development requires a more efficient scripting service and job submission system. The project team's previous experience with HCS played a part in the FLIM process by informing the type of technical dependencies in the FLIM development. The background experience of the Project Team in developing the HCS model served as a reference in the construction of the FLIM model. The FLIM data model could benefit from the experience of the HCS work, for example by reproducing the process for a proposal and the steps necessary for creating a data model for a community. Extract 6 shows this. The disparity between the HCS and the FLIM data models is in how the HCS scientific community identified the need for the data model.

> *Write proposal (as with HCS Screen Plate Well)*
> *Submit to several groups*
> *See what they need*
> *Repeat (as a process)*
> **Extract 6 -** 2009-07-03 OME Group Conference Call (OMERO 2010)

The need to create a FLIM data model led the Project Team to search for an external FLIM-based institution, as there was a limited amount of FLIM work within the Skye institution (The details of a visit to the institution outside of Skye are discussed later in section 8.4.5). Using two user FLIM groups was the main work practice of the Project Team throughout this phase of development and this was implemented in the March 2009 OMERO team meeting. Extract 7 below gives an example of a discussion during a meeting that leads to a further smaller meeting that is held outside of the entire OMERO project team.

> *Relation to spectral lifetime?*
> *currently only adding one dimension*
> *supporting N-dimensions? – Highlights wider issue for model*
> *enumeration is N! bigger discussion. off-meeting? 1400BST Monday*
> **Extract 7 -** 2009-07-03 OME Group Conference Call (OMERO 2010)

The dependencies identified in Extract 7 extended to the communication and interactions with the wider community. This was centred on the core development process, as Levi the OMERO developer was interacting with the local scientist Patrick working with FLIM (end user 1) and Carly – a bio-informatician (end user 2) who was working on algorithms related to the FLIM work. Figure 8.6 shows the interactions and lines of communications of the workflow and requirements for FLIM.

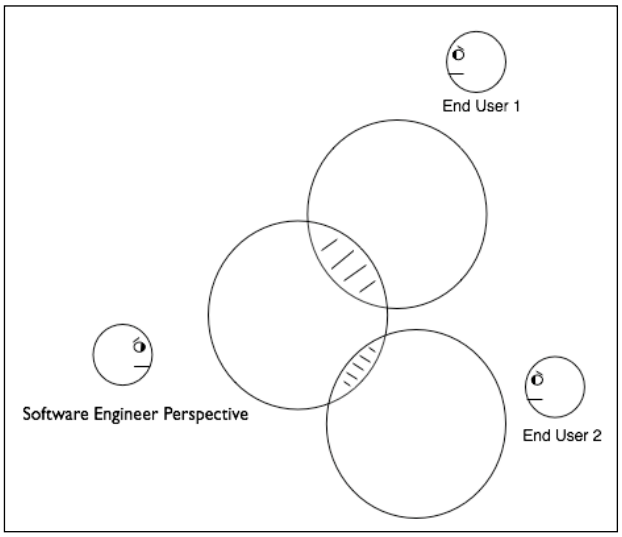

Figure 8.6: Roles in the FLIM work

The interactions of the software engineer with the scientists led to the proposal of the following FLIM workflow.

- *Tag the images with 'FRET', 'NoFRET' – Providing the ability to categorise the images into two sets*
- *Draw one ROI in image and one ROI in the background*
- *Assign to each ROI a keyword from FLIM workflow*
- *Run script on dataset*
- *Results k1, k2, a1, a2, chi attached to each image*[6]
- *Combined results of each cell attached to dataset*
  - *Generate histogram*
  - *Stats: average, std dev*
- *Heatmap for each image*

MacDonald Unpublished

[6] These are typical statistical results of a FLIM analysis.

Community collaboration and interactions have created further dependency for the project team, as the project team is dependent on receiving information to move forward with the on-going FLIM development (Extract 8).

> "*Yvan has contacted Carly (the bio-informatician) about her algorithm but has not heard back yet.*"
> **Extract 8 -** 2009-08-07 OME Group Conference Call (OMERO 2010)

The development meetings showed the communication and interaction with external end users and the dependency of information from them was necessary for SSD process. This involved, in part, identifying the spectrum of end users with whom the project team communicates. How to deal with and react to situations when communicating with end users was not something that was feasible to apply within the remit of the PCF. This question is discussed further in Chapter 9 section 9.4 about how further work may investigate a policy to help with exchange of team communication. Indeed, how the area of communication is generally handled within the team was a significant area of the evaluation for the work. This also involved thinking about the communications with people outside the project team in the context of academic SSD, where a project team is co-located and has access to information about its environment. For the FLIM development work, the communication had a dependency on how the development was going to be implemented with the OMERO team and how these decisions had to be communicated to the community so as to gain feedback.

Extract 9 below demonstrates the dependency that exists between the on-going technical developments of FLIM and the creation of an ideal workflow for the initial user. This involved analysing the on-going feedback from the scientist.

> *Miles: breaks downstream things that we were planning*
> ** sub-optimal: ok. But we know we're not giving Patrick everything*
> ** Does it give Patrick anything he can use?*
> ** We're already stretching his delivery requirements*

Should we get an independent assessment?

Patrick is the arbiter. Can we get feedback from him?

Extract 9 - 2010-04-01 Thursday Meeting (OMERO 2010)

There is an underlying element of dependency and co-operation for the project team, within the project. This phase of observed work for the FLIM development has demonstrated the compromise that exists between the infrastructure and functionality. These decisions between the infrastructure and functionality concern either the development of the infrastructure in the client-server software, or the development of the functionality of the OMERO project. The project depends on both aspects for the software to evolve. This has been significant for the OMERO Team in supporting the feature developments for the community. The OMERO Team may either be focused on the area of software infrastructure development or software functionality. When the focus of development is on the software infrastructure functionality, it can have the effect that the improvements to the software slow down to the users. It further points out the necessary communication between the three elements of the SSD process, UCD process, and then out to the community, so it does not have a negative effect on the uptake of the software. From the perspective of the Project Team and the PCF, how this could be accounted and managed for in the PCF is by the Project Community Mediating role. This again stresses the importance of the Project Community Mediating role to communicate the information between all necessary areas of the SSD project.

8.4.4 Clarification and communication within the Project Team

The aspects of the FLIM work that demonstrate the collaborative work of the software team are the continuing clarification for communication and the dependency for synergies. Several aspects in the planning and evolution of the software objectives over time show this. The following, Extract 10, highlights a discussion on the short-term goals for the project. The focus is on how the interaction within the OMERO team is being used to clarify and refine the software development activities for the upcoming community meeting.

This meeting will therefore define our activities for the next 6 weeks, in the lead-up to Paris meeting, and decide what infrastructure we will fix, and what functionality we will add in the coming weeks.
To deliver these goals, we need to agree on a work plan. This will likely mean a very integrated work pattern, using the mini-group/iteration system.
Extract 10 - March 2009 OMERO Team Meeting (OMERO 2010)

The team meetings have benefited from the social nature of the software development process with its on-going clarification and communication. Extract 11 below further demonstrates the OMERO team's planning of the next iteration and testing for FLIM. The discussion of the FLIM testing identified the need to purchase a new server for the FLIM testing because the testing required more intensive computation.

Yvan: *What' next in the iteration? Would be nice if we can decide this sooner rather than later. Levi's FLIM work should be ready to start testing in anger.*
Jack: Good time to buy a new server for FLIM.
***Action**: Everyone start planning the next iteration.*
Levi: Then we have a FLIM workflow that saves 10s of hours. Really just want to give folks the heatmaps.
Extract 11 - 2010-03-30 Tuesday Meeting (OMERO 2010)

The following extract (12) demonstrates the communication within the Project Team. It highlights Miles' point about the need to communicate the collaborations that will occur within the OMERO team for the upcoming FLIM work. This also demonstrates the core technical understanding the OMERO team has developed that is required for the FLIM analysis.

Miles: Obviously, there will be some synergies here
- Making simple ROI measurements and ROI support on server is an example.
Extract 12 - March 2009 OMERO Team Meeting (OMERO 2010)

Extract 13 further highlights the evolving nature of the science and technology. This is shown with the particular discussion with the team's support of N-dimensional object as more users (scientists) were coming with this type of files from a new scientific OMX microscope. The relevance of the issue, given the growing uptake of the new OMX microscope in the community, was significant for the team as more information about the microscope was required from the community. It is always important to understand this throughout the project, particularly when new microscopy techniques are being used.

FLIM
review of devteam discussion
using deltaT in planeInfo to differentiate (rasterizing)
supporting N-dimensional object. more uses are coming online (OMX,...)
trying to keep as much information as possible
structural illum. v. flim v. ... timelapse
supporting a wide-spectrum of use cases
discuss in Paris
Extract 13 - 2009-05-15 OME Group Conference Call (OMERO 2010)

Extract 14 demonstrates how the development team has formed an understanding of the scientists' work in FLIM. The FLIM analysis was significantly beneficial for the development process, as it taught the development team the value of providing such a tool in their software.

FLIM-analysis
Then there should be another dozen people wanting to do FLIM
Time? A week to have something to show people.
2 iterations (including holiday)
Has a month worth's of results, then he won't be doing it anymore.
Jack: is this FLIM or analysis?
Levi: focuses on FLIM but generally fixing namespace/keyword selection for ROI's
Jack: Bob's ticket 2036 could refactor the ROI workout
Jack: estimate of 2.5 for FLIM is not realistic for everything in ticket 1985

Levi: not sure how long it'll take

Levi: two steps that haven't been mentioned yet

** changing colours of ROIs and ...*

** not doing it now, but necessary for other people*

Action: Sit down tomorrow (Bob, Yvan, Levi) about the way to break it down

Not 100% sure of Patrick's deadline (started 2 weeks ago)

Action: Need feedback from Patrick (the scientist)

** Needs tagging/annotation of ROI selections (to distinguish ROIs from each other)*

Extract 14 - 2010-03-11 Thursday Meeting

8.4.5 Exploration of an external FLIM based Institution

In addition to the development work for FLIM in Skye, a visit was made to a second academic institution where several imaging staff and a PI were using the FLIM technique. The main purpose of the visit was to gather information about the FLIM data model, and to view the additional development work that the institution was involved in regarding the FLIM technique. The visit was arranged and planned through a contact of Miles at the selected institution. This demonstrated the benefit of community promotion that Miles is involved in (please refer back to Chapter 3 section 3.2 for information on the role of Miles).

Figure 8.7 provides an overview of the Mull institution, which was discussed during the software developers' 2009-07-03 OME Group Conference Call meeting. I participated in a visit to the Mull institution and was able to take new findings about the use of the software back to the OMERO development team. The visit to the Mull institution was an approach of understanding the differences from Skye as it provided a view of another scientific institution having the OMERO software setup. Figure 8.7 illustrates the range of departments within the faculty of Natural Sciences where OMERO was being used. The OMERO team used further this information to establish one of the key points of contact for the Mull institution.

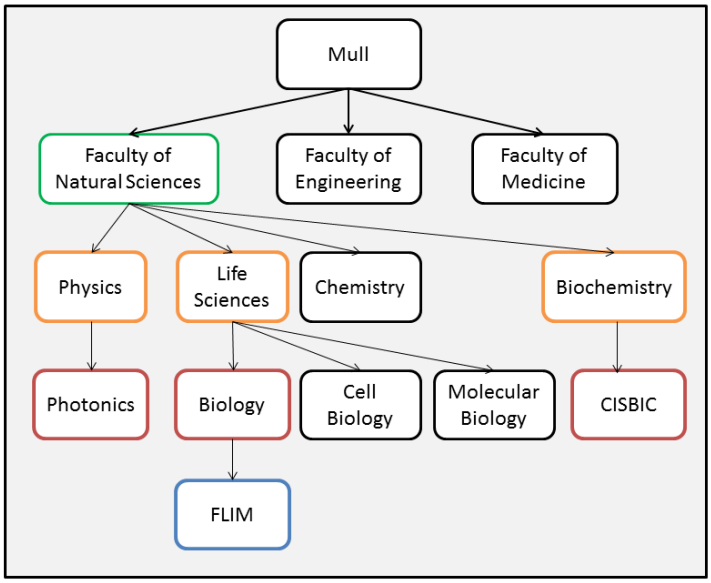

Figure 8.7 Overview of the Mull institution

The FLIM analysis was conducted in the biology department; this work was complemented by on-going research and development work within the photonics department. Some of the aspects covered within the visit to the Mull institution ranged from the file's formats, an overview of the FLIM data and the institutional workflow, the existing resources of the institution, and the specific background to the laboratory setup.

The FLIM findings gave to the development team various insights about the existing resources available in different work practices of another institution. For example, such an insight has provided information about the image file types (Leica and Zeiss) and the requirements based on this:

> *We need this reliably* (Leica and Zeiss files) *importing before we are able to fully get going.*
>
> **Extract 15 –** Loynton, unpublished

Further points have elaborated on the nature of the scientific work carried out at Mull institution. Extract 16 highlights the perspective gained into the scientific background at Mull institution.

The laboratory deals principally with quantitative microscopy. This means two possibilities for their work:

1) Cell biologist work at labelling protein interactions. (This is the same work that is done by Patrick in Skye)

2) Biologist viewing naturally occurring proteins i.e. these occur in skin. Johan, a researcher/lecturer based at Mull institution, is doing some preliminary work in this area.

Extract 16 – Loynton, unpublished

8.4.6 Information outcomes for the project visit

The outcomes of the external visit in relation to the development work and project are summarised in the 2009-07-03 OMERO group conference call notes. The key implications for the data model highlighted in Extract 17 are the two changes to allow storage for additional planes of data and values from microscope instruments in order to perform the analysis calculations.

Lifetime meeting at Mull
2 categories of changes
- Changes to allow storing additional planes of data in sensible way
- Values from instrument to do calculations: bin, width, etc.
How to find the proper level for attachment
Relation to spectral lifetime?
currently only adding one dimension
supporting N-dimensions? – Highlights wider issue for model
enumeration is N!
Bigger discussion. off-meeting? At 1400BST Monday

Extract 17 - 2009-07-03 OME Group Conference Call

The outcome from the meeting concerned the need to support the continuous technical challenge of N-dimensions images. This challenge affects the technical development of the OMERO project on an extensive scope, as shown at the end of Extract 17 where a further

meeting is scheduled. The Project Team had several discussions about which solution to take and the possible evolution of N-dimensions. Whether the solution would provide a subsequent future solution for N-dimensions or not generated many key questions about the development of the software. The OMERO team had to decide between the solutions and their effects in the implementation in the code. The schedule for the rollout of the solution was intended to be over a long period, again emphasising the multiple aspects of the N-dimensions problem.

The above instance for the development of N-Dimensional work presented a instance where a quickly put together solution may offer an immediate short-term benefit with the development of FLIM. However, this solution has subsequent dependencies in the code base in the Review of FLIM, so an incomplete solution could be detrimental to the wider SSD process. Applying this to a Project Team, a quickly put together solution would need the full agreement of the Project Team to avoid ill thought out solution that may affect dependencies in the code for others.

The interpersonal benefits for visiting the institution concern establishing lines of communication with the right people, which is beneficial as it makes gathering requirements for the Project Team easier for any future demands. In addition, a face-to-face communication with the institution helps to demonstrate that the Project Team wish to form or maintain collaboration.

8.4.7 FLIM developer discussion

For the FLIM work, I did have the opportunity to carry out informal questioning with the OMERO team. The information provided in the present section summarises the information from these conversations, as it has supported the FILM Review 2.

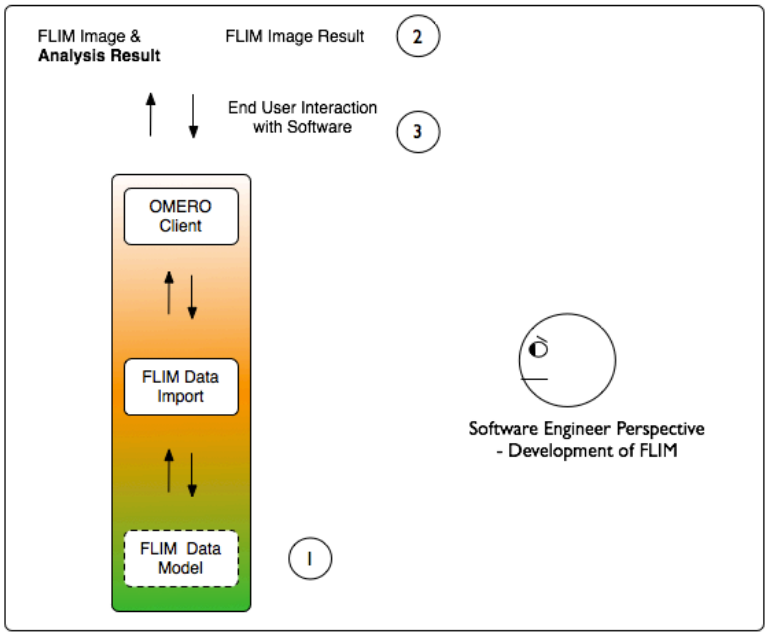

Figure 8.8: A software development view of FLIM

1) A key problem in the FLIM development workflow is that the OMERO software does not provide the actual FLIM result (See Figure 8.8). The implication of this for the software developer is the limited access to the image. Consequently, the development process is placed under technical restrictions with what can or cannot be done with the FLIM analysis. The development is restricted to performing the calculation on the image result only.

2) The FLIM work required an understanding of the scientific FLIM workflow to be established. The FLIM development work has followed an iterative process with feedback in regular meetings between the developer and scientists to confirm the changes to how the FLIM ROI is generated by the user and once created, the software then handles the information and processes the result.

3) The FLIM data presented a challenge because a Data Model had to be created from scratch. This required the developer to carry out exploratory work on the construction of the Data Model. In creating a FLIM Data Model, the development team had to make certain assumptions about channel and time-points; consequently, the information is not stored in an ideal way.

In discussing the development process with Anthony, the software engineer responsible for

the Data Model, the following points were made.

- The perspective of investigation in the Mull institution focused on the technology to develop for FLIM. This led to a more abstract view of how the Data Model was required to interact and develop with the FLIM requirements. This was valuable because it linked together the requirements from the real world, but it also generated points about a further context of how the software was used and where to develop the software. This information helps the OMERO project team to look at 'potential future developments' of the software.
- The discussions at the institute were held to create an abstract 'picture' for the preliminary FLIM work. However, the requirements and details for the implementation would require more detailed material as the collaboration evolved.
- Anthony felt that there was still little appreciation and insight into the software development work, and the type of work that was required for the process. In using the FLIM work as an example, Anthony explained how a scientist would only see the final image analysis as a result of the FLIM work. This would not reflect all the additional work that was required to be implemented by the OMERO team to get this result. A personal question raised by Anthony was if the scientific community appreciates the background work required to implement a solution and understand that it will take time.
- A further example of this and the difficulty that is encountered happened at the OMERO User Meeting. The project presented the feature x and now shall present feature y… This may not be the complete end result the scientists want, but it is difficult to clarify and clearly indicate all the work that has to take place to get the desired solution.
- The scale of the work required for the FLIM development and implementation may have been more difficult for the FLIM community to recognise, and, in contrast, the HCS community development work required a lot of image data to be handled and could be more easily acknowledged and appreciated in terms of the scale of the task. The challenge of developing the FLIM work, to some extent, remains hidden.

The second informal interview with Levi, a developer directly involved in the FLIM work, yielded further insights into Review two. The first issue was the co-location of the software

developer, which allowed him to work directly with and interact with Patrick (the scientist) and the local scientists based within the same building in Skye. This helped Levi to build up an understanding of the FLIM work practice carried out by Patrick, which was vital for the initial development work in terms of Patrick's work with FLIM.

This position for the software developer thus offered him direct exposure to the work practice of Patrick's FLIM work. Levi, in these meetings, discussed the FLIM workflow of the scientist and its implications for the development process. However, this required continual interactions for them both to understand the full details of the FLIM workflow. It thus demands frequent meetings for progress updates and what can be described as a feedback loop between the developer and scientist. This interaction between Levi and Patrick was problematic for Levi when continual interaction and communication with him obstructed the development work. Levi highlighted how the focus is on development so regular interactions can, on occasion, distract from the necessary software development work.

In reviewing the implementation of the FLIM development work from the meeting notes, the requirements gathering about the FLIM workflow required a consistent line of interaction and communication within the OMERO team to understand how it progresses and how it may impact on other areas of the OMERO project that FLIM is part of. Extract 18 documents an example of these interactions within the OMERO where both short-term issues of the need to define a ROI workflow and a standardized FLIM workflow are highlighted and longer-term issues such as when a new microscope will be online and available for use. The consequence of this is that the OMERO software must be ready to handle the new image data.

- *Need to use Polygon to define ROI.*
- *Current Polygon in measurement tool needs to be re−written.*
- *Need to define ROI in terms of workflow.*
- *Standard FLIM workflow + User defined objects*
- *Script takes one single image for noFRET and one for FRET need to convert to a batch*

- *N−Dimensional support.*
- *Time information (other metadata)*
- *Multiple channels (we're getting a new scope in June!!!)*

Still further key points for future

- *Deploy to Cluster*
- *Same methods can be adapted for FRAP*
- *Better Quantification*
- *Simple to add more statistics, LDA, Fisher scores between samples*

Extract 18 - MacDonald Unpublished

8.4.8 Summary of FLIM fieldwork

The communication and interaction during the software development process for the FLIM work occurred between the two roles of the scientist's and the bio-informatician as previously illustrated in Figure 8.6. The review demonstrated that this was undertaken for Levi to understand the scientist's FLIM workflow. The subsequent interview with Levi highlighted how his work with Patrick (the scientist) during the development of FLIM led him to take on the role of interacting with the end user of the OMERO software. This was feasible for Levi under the conditions of the FLIM development because of the co-located work environment (see Figure 5.4), as it offered ease of access to potential end users.

However, as Levi pointed out, working with end users brings additional responsibility for the software developer in terms of maintaining a consistent line of communication and clarifying the development process on a frequent basis. A software engineer is often focused on the development process so this is not always conducive for providing a consistent line of communication, even though this communication and information maybe in the interest for the Project Team in the long-term.

8.4.9 Summary of the FLIM review against the Project Community Framework

Review 2 has demonstrated the potential of the Steps of the PCF. The application of Step I and Step II of the PCF to Review 2 is similar to the possible application of Review 1. The actions of Step I may identify the context of the FLIM community and be used to understand the domain, the community, and the work practice. The supporting Project Community Mediating role, as defined in Step I, is significant for the FLIM Review 2 because it provides a role for the communication and interaction between the scientist and the developer and this role did not exist in Review 2. Subsequently, the developer was involved heavily in the communication, which was commented on by the developer that this activity was a distraction from the core development work.

Step II of the PCF examines where Review 2 may have benefited from a centralised repository of information again the development work for FLIM similar to the HCS formation of the Data Model. The use of a information data repository such as a project wiki used Review 2 may have benefited the team. It was discussed early on in the work carried in Review 2 that from lessons may be learned from previous experiences of managing data and so could be applied to the FLIM work. However, this could be problematic in how the OMERO team could effectively document the insights from Review 1 so that it would be applicable to other new areas of development.

The application of the Step III through the FLIM Review 2 involved continued interaction between Levi, Carly, and Patrick. The UCD led techniques applied by the developer Levi was interviewing the users. However, as highlighted in Review 2, this also allowed him to maintain the feedback loop between himself and the scientists. Based on this type of interaction, the research advocates the Project Community Mediating role to provide continued communication while the SSD developer focused on the development work and maintain a continual level of communication. The Review 2 example also demonstrates the lack of UCD expert involved in the process and how a SSD developer has taken up the responsibility of carrying out user feedback. This is not recommended in a SSD project. The evidence from Review 2 that would support the need for a specific UCD role was demonstrated by a comment by the developer was involved heavily in the communication was that this interaction was a distraction from the core development work.

Step IV (Phases 1 and 2) of the PCF is in some aspects demonstrated through the FLIM Review 2, or more specifically through the collaborative working and the mini-group meetings where the OMERO team further clarifies and refines the software development issues. This reflects aspects of Reflection-on-Action as it allows for actively thinking back on what has been done and employs elements of strategic planning for the Project Team in allowing insights to be developed from the Reflection-in-Action. What Review 2 does not demonstrate is the full reflection of the SSD, UCD, and the community uptake in relation to the software, where the central focus is on SSD issues. The Reflection-in-Action through Review 2 is only briefly demonstrated in the N-Dimensional work because of the connection with the FLIM work. The OMERO team required an element of planning with the long-term development of the N-Dimensional work alongside current work of the FLIM development.

8.5 Discussion of the reviews

The two evaluation reviews have aided in understanding of the PCF because they have reviewed actual SSD meetings and problems. This is explained below:

- Review 1: The focus of this evaluation was on the problem of image management within the HCS community, which required the OMERO project to form and develop HCS support in the OMERO software. The evaluation retrospectively analysed this development against the PCF.

- Review 2: The focus of this evaluation was on the formation and development of the FLIM technique within the OMERO system. The evaluation took place within the local community and drew on experiences from a second institution (Mull). The evaluation again used a retrospective approach against the concept of the PCF.

The two reviews have again shown many challenges during the SSD process. A key common element is the collaborative working, whether with fellow developers/software engineers or scientists themselves, because of communicating and translating the complex

nature of the scientific work into the software. Anthony, an OMERO developer associated with the FLIM work, particularly highlighted this point with how the complexity and challenges of the development work can be hidden from the scientists who only see the end result. The evaluation has examined the meeting notes with two separate expert software developers, who create scientific tools through interaction with scientists and feedback from the community. The reviews have demonstrated the expertise gained by the software developers communicating with the scientists; in both Reviews 1 and 2, the developers were co-located with scientists within the same institution – Skye.

The reviews highlight a key issue about the multi-disciplinary roles that can make up the SSD process. This has been covered in both evaluation reviews of the OMERO Team meeting notes, where experts on such things as data visualisation, image classification, and mathematics were used. Thus, these are core concepts that are transferable and benefit scientific inquiry. This itself returns to the point of defining the Project Team and the Project Community for two reasons. First, the Project Team and the Project Community Mediating role must comprise multiple individuals to support the multi-disciplinary contributions, one of which is UCD. Secondly, the interactions with multi-disciplinary roles outside the Project Team can mean working with role(s) whose main goal do not reflect the same goals of the Project Team and interests with the SSD software. Consequently, the communication and feedback can be affected, and in these circumstances the value of having a Project Community Mediating role(s) can aid to take the responsibility of the continual feedback and communication.

The following discusses the PCF in relation to two evaluation reviews that I have conducted. Table 8.2 identifies where the PCF has been applied through each review and to what extent each review has demonstrated the PCF steps, as covered in Chapter 7.

Table 8.2 Project Community Framework evaluation review

	Review 1	**Review 2**
Step I: Capture and characterisation of the Project Community	✓	✓
Step II: Storage of the Project Community information	✓	✓
Step III: Process of UCD	✓	✓
Step IV: Project Community action and reflection		
Phase 1 – Reflection-in-Action	X	✓
Phase 2 – Reflection-on-Action	✓	✓

In the above table, a green tick represents significant correlation of the corresponding PCF step while an amber tick represents some correlation to the PCF step with scope for further adoption. Finally, a red cross represents where the step has not been definitely validated in the review.

Table 8.2 shows how each step of the two reviews has been applicable in some degree. The positive learning experiences from the PCF as a method were that the steps were distinctly defined and so it seems there is no overlap in the instructions of the steps. This is perceived to be a positive factor given the range of users that the PCF has been formed for. A significant positive learning experience was how Step III may be applied. Review 2 showed how a scientific software developer was using interviews to gain feedback for the development process. However, given the demands of the interaction it was stated by the scientific software developer that this interaction became a distraction from his own coding process. Based on this, it would warrant the inclusion and requirement for a UCD role that may have taken on this process.

The amber tick and red cross in Table 8.2 indicate further scope for investigation regarding the PCF. For Step I, the further scope of questioning would investigate multiple SSD

projects and ask how robust capturing and characterising the Project Community would be in different situations. For Step II, it is identified that a significant amount of investigation would be required, as it would involve questioning the types of software development tools that could be used to support the PCF. For my own research, this also connected to the further ways I may evaluate the PCF. The full conclusion to this is explored in Chapter 9 section 9.4. The research now goes on to describe in the following section the further output of the research of the Project Community Framework Manifesto.

8.6 The Project Community Framework Manifesto

In part, the Project Community seeks to project the experiences of the research work by the philosophies and practices of UCD. A common way of promoting such practices has taken the form of a manifesto. A manifesto aims to encapsulate the strategy and position of work as it moves forward. In moving to use a manifesto for my own research work, it aims to capture and represent the core principles of the research of SSD, UCD, and the community of the project. The research findings have observed the significance of uniting the elements of SSD, UCD, and the community of the project. This manifesto has been set out to encapsulate the requirement and thinking for SSD projects to do this.

Examples of this in the current research include the software design manifesto (Kapor, 1996) and the Agile manifesto (Beck *et al*., 2001), which both respectively set out a call to action for evolving the thinking of the practices and processes of software development. The research work has particularly drawn on the experience from the software design manifesto and the principle stating that programming and design activities of a project must be closely interrelated.

So to present the qualities of the PCF and to stress its own UCD led philosophies for working in the academic SSD context, the direct findings from the research are listed below:

1. The identification of the Project Team but also its position and its relationship with the community.

2. The identification of the Community, its boundaries, and but also its relationship with the Project Team.
3. The value for continual communication both within the Project Team and with the community.
4. The need for continual reflection on the Project Team's creativity and the communication of this within the team.
5. The continual facilitation of communication both within the Project Team and community.
6. A requirement for listening to the community and the Project Team.
7. A SSD project must start equally with the consideration of the scientific software development, the scientists, and the scientific software project community.

Table 8.3 below shows the evidence of each of these seven points in the research.

Table 8.3: Manifesto summary

Point	Evidence in the research
1.	This point was linked to Review 1 through the action of defining the Project Team. The OMERO team then took into account of the wider community in the creation of the HCS data model, where a non-member of the Project Team was not neutral and was imposing a view without any consideration for the wider academic community.
2.	Recognising who is part of the Project Community was used in both reviews. This raised awareness about who is and who is not involved. For Review 1, those involved in the work extends to the Bio-Formats project and the interaction required for the Bio-Formats project to read proprietary HCS file formats. Review 2 is significant for the OMERO software developer's role of interaction with the FLIM scientist and bio-informatician. They are both collaborating members of the Project Community. Review 1 has also illustrated this in terms of dependency within the Project Community.
3.	The level of communication was most prominent in Review 1; this communication of the development process was evident in both the Bio-Formats project and external commercial entities.
4.	The importance of reflection was identified in both reviews. This concerned, in particular, reflection on the interaction in forming the HCS community and the reflection-on-Action with the interactions with HCS community to help to support the development of the FLIM community.
5.	The role of mediation by a software developer was demonstrated in Review 2. The mediation that took place within the Project Team was for the continued scientific software development throughout the user meetings for those involved

	in the FLIM work. The software developer in this instance was the mediating role for the communication of information within the Project Team. The further work in Review 2 at the external FLIM based institution provided wider feedback for the Project Team. This was documented in the outcomes from the visit and it involved both a developer role and my own role as a mediator of the information back into the Project Team. This was demonstrated in the project meetings that took place for the Project Team. The OMERO Community meeting was also further evidence of this. The importance for the mediation is in the exchange of information between disciplines.
6.	This point is central to the UCD perspective of the manifesto and is vital to ensure that the Project Team can form a wide and well-informed perspective. This was evident in Review 1 through the infrastructure and development conflict, where the users of the system are centrally concerned with seeing the development of the system and the functionality that comes with this. The software developers in the context of the client-server architecture are required to support the infrastructure to allow for the required development in functionality.
7.	The final point concludes with the fundamental philosophy and motivation for the Project Community as a new way of thinking. It draws on all three elements of a SSD project – the SSD, UCD, and community – and stresses the importance of all three for a comprehensible software experience. The SSD points are evident in Review 1, with the technical focus of managing the size of HCS image data and the requirements for database upgrades. The UCD points are apparent in the direct feedback from the FLIM scientist through Review 2. The community point is evident in Review 2 where the visit to the external institution was used to both gather more information about FLIM but also to understand how FLIM was being used to support the SSD process. The central message of the PCF is to ensure that each of these aspects of the SSD, UCD, and community is accounted for equally in an academic SSD project.

8.7 Summary

The main contribution of the PCF is in how it has demonstrated that the most valuable part of an SSD project is the people of the Project Team. This finding emerged despite the many challenges and conflicts within the Project Community. This is based on the conclusion and observation of how the PCF was responsive to certain PCF steps and the explanation that a Project Team all committed to the SSD tools, UCD process, and commitment to the Project Team community functions better than a Project Team that does not.

Because of this and expectation that a SSD project may or may not be responsive to certain PCF steps the following five questions have been constructed to support the Project Community Mediating role (see Chapter 7 section 7.3.8) and to help to uncover other challenges not anticipated within a SSD project and its community.

1. How do you decide which type of users/groups to target first?
2. Do we have access to an entire laboratory, collaborating laboratories, or scientific institutions for the initial software use?
3. Is the software use compulsory or optional?
4. What are the barriers to entry?
5. How can you make the early adopters happy?

The fieldwork process has formed a dual perspective with analysing scientific users and then the scientific software development team of OMERO. This has given the research the ability to look through the multiple perspectives of SSD, UCD, and the community. It is this fieldwork process that the PCF has emerged from. The research outcomes would have been significantly different without both the ethnographic information analysis and the co-working with the SSD Project Team. This consequently led to the formation of the PCF in such a way that it accounts for scientific software development tools, UCD, and the directives for an SSD Project Team for the awareness within its Project Community.

It is the PCF intention to be accessible and open to interpretations because of the many possible perspectives that may make up an SSD project. How successful this will be is a question for further work and is discussed in Chapter 9, but the implications this has had on the steps for the PCF is that they are set out so that a user of the framework may utilise its full aspects. The application of the steps of the PCF includes using software development tools, employing UCD, and the two iterative phases of Reflection-in-Action and Reflection-on-Action. Reflection-in-Action allows for the Project Team to spend time on understanding the actions of the Project Team by exploring these as they occur in the SSD project and Reflection-on-Action aids in new understandings and giving insights to inform the goals of the Project Team.

Also related to maximising the accessibility and use of the PCF is to promote its integration within the Project Team. The research acknowledges that such work by Demarco and Lister (1999) discusses integrating teams so that a team's work is flexible in itself to the situations that arise, and that the team members are working towards a common set of goals. In working towards this point, the research acknowledges the work by Demarco and Lister (1999) and their definition of professionalism. They highlight that professionalism in a healthy work culture is based on people to be knowledgeable and competent in what they do. The element of professionalism is a factor desired in the Project Team to create a sense of belonging centred on its goals. Team members are more effective because they're more directed. Therefore, goal alignment for the individual members of the Project Team and for the Project Team would be beneficial.

A further point established in the existing literature and discussed earlier in this research, was the seven challenges of UCD and software engineering integration (see Chapter 2 section 2.7). This research recognises that many of these problems cannot be easily solved in the short term (e.g. the need for UCD to be adopted throughout an organisation, a lack of support tools for the UCD process and the education gap a UCD students in software engineering and software engineering students in UCD). Therefore, the purpose of bringing professionalism into the Project Team is to aid this sense of belonging and aid working towards SSD project goals.

This research has established the need for a wider approach to integrating UCD into academic SSD rather than simply applying UCD methods without recognising the challenges of working with UCD in software development within the scientific context. It has explored how the work within a SSD project and its Project Team is vital to the project's success within the fast-moving, complex, and highly demanding world of the scientific community. The research has gone on to explore a practical way of drawing together the aspects of the scientific software development, UCD and the SSD project Community. This is provided by software development tools used, clear use of need for a UCD in all SSD projects, and the continued reflection and action of information for a SSD project. The research work has culminated and contributed to the creation of an initial framework – the Project Community Framework.

The research has not gone on as far as creating a fully operational community-orientated framework. This is central to what the evaluation has demonstrated. The research work acknowledges that this is an area for further development (See Chapter 9 section 9.4). What has been contributed and formed in this research though is a foundation so that a fully-fledged operational community-orientated framework that can be based on the core findings of this research. This core foundation has gone on to be evaluated through Reviews 1 and 2, where the topics of HCS and FLIM were explored. The extent of the further work for this research is now discussed in the concluding Chapter 9.

Chapter 9: Conclusion

9.1 Summary of the research

This research set out to investigate and analyse an academic SSD project where UCD was applied through the Usable Image project into the OME project. The research was carried out from within the Usable Image project – an EPSRC funded research project allied to a longer standing academic SSD project called OMERO. The Usable Image project set out to investigate the various challenges and constraints to attempt to extend and improve the adoption of UCD methods and principles in OMERO. The research presented in this thesis was a subset of that project specifically concerned with identifying barriers to UCD adoption in academic SSD, and proposing tools or methods to help to overcome these. The complexity of academic SSD in this work has been observed from two perspectives; that of the scientific and the SSD domains. The subsequent analysis of these two perspectives led to the formation of a proposal for a Project Community Framework (PCF). The implication this has had for the research has not been solely about the application of UCD for scientific software but also about the ability to integrate UCD with it and developing an understanding how academic SSD is conducted.

The fieldwork for this thesis began in traditional UCD fashion with a focus on the scientist end users. Through the fieldwork, the research started to comprehend the position of UCD in the context of the SSD and its interconnectedness with the wider elements of SSD. The second phase of the fieldwork examined the actual SSD and sought to understand the working practices of a real SSD process. Based on the analysis of these two perspectives, the manifesto for the PCF was formed.

The research fieldwork demonstrated that academic SSD projects must focus not only on SSD and UCD, but also on the community of which the SSD is part. The challenge for designers of SSD systems is in ensuring that SSD reflects, considers and responds to the needs of the wider view of the "user community" than is traditionally considered in UCD. The PCF encourages a more holistic understanding of the various players in the SSD, UCD, and the end user communities that comprise any SSD project. Furthermore the PCF

provides a means for bringing such an understanding to the fore, so that they may be accounted for by the SSD project.

The philosophy of this approach is provided in the form of the PCF manifesto (see section 8.6). As previously highlighted, the points of the manifesto are as follows:

1. The identification of the Project Team but also its position and its relationship with the community.
2. The identification of the Community, its boundaries, and but also its relationship with the Project Team.
3. The value for continual communication both within the Project Team and with the community.
4. The need of continual reflection on the Project Team's creativity and the communication of this within the team.
5. The continual facilitation of communication both within the Project Team and community.
6. A requirement for listening to the community and the Project Team.
7. A SSD project must start equally with the consideration of the scientific software development, the scientists, and the scientific software project community.

The work by the editors of the famous Science journal (2011) describes the emergence of the data role for scientists, as they frequently have to deal with larger and larger amounts of scientific data. The resulting implications are that many scientific data sets are becoming too large to download and work with. Even where the data might be accessible, it can be inefficiently organised to use for any scientific research work. The OMERO project - a scientific project that deals with the microscopy image data management - is trying to tackle this increasing hurdle. My own research integrates in this project by ensuring that such data management scientific software is developed not only to handle the data but also to integrate the principles of UCD and the consideration of the academic scientific community.

The research has directly observed through the ethnographic fieldwork in the Usable Image Project and then within the OMERO project (See Chapters 5 and 6). This provided multiple perspectives allowed the research work to form multiple standpoints.

This research investigation has outlined the PCF aims to accommodate for the software development tools used, the role of a UCD, and the continued reflection and action of information for a SSD project. It defines mediating roles in order to promote communication and information throughout a Project Team. This has led to the concrete proposal of the PCF manifesto that has been built around the three key elements for academic SSD: SSD, UCD, and the community.

9.2 Summary of contributions

This research has overlapped across the domains of science (specifically the field of image informatics) and SSD, with the overarching perspective of the research investigation being from UCD. The research has investigated the gap between UCD and SSD, which has consequently resulted in the construction of the PCF. This PCF has subsequently been used and evaluated in two reviews.

The first contribution has been towards the deeper understanding of the gap between UCD and SSD which lead to the proposal that SSD development requires a balance between the development of SSD, UCD, and the community. The research has recommended this balance to be consciously considered from the very beginning of a SSD project. The PCF has echoed this requirement by recommending a UCD professional(s) to be part of a SSD team and by defining a Project Community Mediating role, a role specifically formed to aid in the communication and feedback for a SSD project team.

The second contribution of the research is the project community manifesto. It proposes a richly ‘community focussed’ way of thinking about the development of academic scientific software, away from exclusively starting with the scientists as the way forward described in the work by Kalawsky *et al.,* (2006).

> "START WITH THE SCIENTISTS, not the technology – what are the problems that they want solved." (Kalawsky *et al.,* 2006)

According to the findings of this research, starting solely with scientists sets out a false pretence and imposes a limiting perspective on the complexity and scope of problems that exist in academic SSD projects. The implications of this research are that academic SSD projects must re-think the wider context of incorporating the UCD approach into academic SSD. In light of the statement made by Kalawsky *et al.,* (2006) and the work proposed in this research, I suggest instead that academic SSD must:

> "START EQUALLY WITH THE SCIENTISTS, THE SSD, AND THE SCIENTFIC SOFTWARE DEVELOPMENT PROJECT COMMUNITY GOALS – focusing on any single aspect will unbalance the direction of the SSD project."
> (Project Community Manifesto – Loynton, unpublished results)

In redefining this statement, the resulting proposal retains the focus of scientists but also recognises and aims to unite UCD with SSD and the community of the SSD project. The PCF then can be viewed as a medium for improving the uptake of UCD philosophies, methods and thinking in academic SSD.

9.3 Further testing

In addition to the limitations of the research work the Framework would require additional testing – this would be valuable and worthwhile for any future work because of the significance of the challenges that have been discussed in the research. The prominent challenge and first reason for the further testing is for the benefit of integration between UCD to SSD. Step three aims to integrate UCD into the SSD process and so step three would benefit from having a greater understanding on how UCD and SDD can be better united. Although, no specific UCD methods have been specified in step three. The major variable condition identified in step three is the range of UCD methods and how they are used throughout the project. The recommendation based on this for the future testing is that a wide spectrum of UCD methods are tested and used and documented in a variety of SSD

projects, so as to gain an insight into what types of UCD methods work best and under what type of conditions.

The second challenge and second reason for the further testing is for the purpose of examining the framework within the context of different SSD projects. The purpose of this role of testing is to aim to asses the framework in a wider range of SSD projects and the variety of constraints this brings with it. The necessary future testing would require the PCF to be tested in various specialised SSD projects as has been previously described by Killcoyne & Boyle (2009), Basili *et al.,* (2008), and Ackroyd *et al*., (2008) (See Section 2.7.1 pg 46). Killcoyne and Boyle (2009) have formed a specific research informatics team to meet the challenges of software development within the life sciences. The ten-man team is involved with many projects across the institute. This team have formed a process that supports good communication across the team, rapid development and delivery, and project management to coordinate development and manage dependencies. It is argued that this type of SSD project with already established practices may more readily support the adoption and transition of the PCF.

However, the testing would also need to be carried out with several new SSD projects where the context of the project is open to a wide set of variables. Table 9.1 has been constructed to provide some indication of the range of variables that could be investigated in further testing. Table 9.1 has been adapted from the work by Liu *et al.,* 2008. For the purpose of this work the additional variable factor of the Project Community Mediating Role(s) has been added. This variable consists of the Usability champion, Project Community Mediator(s) and so would require further testing based on availability of these roles in a SSD project. The goal of testing with new established SSD projects here would be to establish how the PCF could benefit a project when it has specific goals set from start and how this can effect the goals of the UCD process and the goals of integrating information across the project and for the evolution for the project.

Table 9.1: Scope of Possible Variables for Further Testing (modified Liu *et al.,* 2008)

Project	
Purpose	*Purpose*
	Scientific Contribution
Project Team	*Programmer Expertise*
	Team Management

	Programmer turn over
	Designer
	Project Community Mediating Role(s)
Process	*Development Process*
	Group Meeting
	Code Management
	Budget
	Testing
	User Centered Design process
	Schedule
Products	*Character of Software*
	Size and Complexity of Software
	Interface
	Documentation
	Open Source

9.4 Limitations of the research

- Choice of the research approach

The research approach used ethnography and secondary ethnographic data analysis. The limitations of the research approach therefore are linked to the known limitations of the ethnographic approach, specifically; the disruption caused by researcher's presence in the SSD team and poor generalisabilty of results. However, the context-rich data and focus on rich understandings made the ethnography approach suitable for exploring my research questions. The issue of the currency of the data set was identified as a limitation in Chapter 4 section 4.6.1. In any further work, a recommendation to overcome this would be to allow for any secondary analysis work to be carried out with the option to begin at the same time as the ethnographic work. The practical alternatives for a suitable research approach would have involved designing a more empirical approach by setting up multiple experiments with the groups involved. This technique would be required to present artificial scenarios and measurements to the groups. However, it is important to note that the set of suitable resources required for this approach was not easily obtainable. Therefore, it was not chosen, as the research was embedded within an existing collaboration with the Usable Image project and the OMERO project, so this had to be taken into account in the selection of the research method and so this was why the ethnographic method was chosen.

- PCF context and evolution

The two reviews of the PCF have allowed the research to iteratively develop the framework and identify the shortcomings of the work. The major limitation of the study surrounds the context in which the PCF could be evaluated, which in turn relates to the existing known limitations of empirical software development work with UCD, specifically as discussed in Seffah and Metzker (2008). This concerns the difficulty to control and replicate the benefits of UCD between separate software development teams, as multiple factors would need to be accounted for (e.g. skill, motivation, and the software engineering approach). This highlights the further question of how the PCF must continue to progress as the software evolves and how this may be evaluated. The critical issue here is dependencies on the people using the PCF and their ability to continue to learn and evolve alongside the software tools and techniques within the framework. It also brings additional limitations in the form of an increasing responsibility of running the Project Community and promoting its values. This dependency rests heavily on the Project Community Mediating role(s) within the Project Team to continue to promote the communication and reflection for the design and development work.

Nevertheless, the value of the evaluation approach used in this research is that it has allowed for the PCF to be understood in the context in which it was formed. The major limitation recognised for the approach was that it introduced a bias from the OMERO developer in choosing a second evaluation topic (the FLIM Review 2, proposed by Yvan). This was an advantage for my immediate evaluation as it gave me an existing area for the project. Thus, it did mean that my selection was externally influenced. Additionally, the evaluation review did explore existing data that I had been aware of from the analysis work carried out in Chapter 6 (Extract 14 presented in Chapter 8). This was for the purpose of the evaluation review, as it provided current and up to date information on the evolution of the feature. Because of the restrictions placed on the evaluation review, the application and evaluation of the PCF within multiple SSD projects would prove to be an appropriate area of future work, with the possibility that multiple SSD projects could be investigated with the necessary support from the funding body to setting up an appropriate Project Team.
Furthermore, a recognised limitation lies in how to effectively compare the use of the PCF between projects. This problem can be compared to the use of UCD organisational surveys (previously described in Chapter 2 section 2.7). Those surveys were difficult to conduct

because it would require comparing the use of the UCD against a different project that is not using the UCD. A similar situation could be applicable to a SSD project using the PCF to one that is not using the PCF.

9.5 Future research work

- Funding of scientific software projects

The RCUK report of e-Science (EPRSC 2010) suggests that usability is an overlooked issue for SSD. This highlights that despite the heavy investment in, and expectations of, academic SSD, UCD is still not critically considered. One problem identified in the research by Segal (2004) is that scientific software is developed for a single scientific question so has a limited shelf life. This SSD practice can mean usability is frequently overlooked. However OMERO represents a growing breed of academic SSD project aimed at providing more widely applicable toolsets rather than single question focussed tools, and in this respect the team and the setup of the OMERO project resembled more then SSD model of a specialised software development team (See Chapter 2 section 2.7.1)

The work conducted by OMII-UK (Open Middleware Infrastructure Institute) and funded by EPSRC demonstrates the positive role of funding bodies and their recognition for developing e-science projects at the levels of software, support, and sustainability. In the recent funding of the Software Sustainability Institute, however, this group of funding bodies and institutions have was created to ensure the sustainability of scientific software and not explicitly UCD. Because of this, a future question posed by this research asks if the funding of academic SSD projects can be re-formed on the basis of all three areas of SSD, UCD, and the community of the software. This area of further work would help to open up further opportunities to examine the requirements of the PCF from the perspective of funding bodies. The next steps in this work concerns two areas. The first area of work identified is in the integration and tracking of the use of the PCF in a wider range of contexts. This would include implementing the PCF from the very start of an SSD project, but the framework would also benefit from further work within a wider variety of SSD projects. This thesis has shown the nature of differences between SSD projects, with the type and scale of the software under development and the scale and size of the Project

Team. The other scenarios that are frequent in scientific software are in larger-scale collaborative SSD projects. The wider opportunities this opens up for the PCF concern how it may be applied at the local development level and then in the co-ordination of multiple development teams.

- Developing the PCF

The second step accounts for the social development of the PCF. This considers the Project Team and its experience as key, especially in terms of forming a jelled team where all those involved are moving towards the same goals. The effect of having multiple distributed parts of the Project Team, along with further variations in a Project Team's experience and expertise, would all play a part in how the Project Team functions. Nevertheless, many of these factors may move beyond the control of the PCF.

The communication of the Project Team could be further addressed through investigating software development tools that may be used in the scientific software development process. The difficulty in this is in getting various software development tools that complement each other and provide a suitable collection to meet requirements. This further work may draw on the field of CSCW, and would be complemented by an additional range of observational work on how tools and techniques of software development may evolve and be refined for the context of PCF driven SSD.

- Education

Another area of further work identified is centred on the educational aspects for future life scientists, bio-informaticians, software engineers, and UCD professionals. A key reason for the project's success is the expertise of the people involved. My own research would also benefit from a wider "cross-pollination" of education between the disciplines in which this research is situated. This was illustrated in the transition between the scientific observations and the requirements of the ethnographic fieldwork to inform the software development process. In this transition, I observed that working between the two projects led me to realise that cross-pollination between UCD and SSD would have helped me integrate the process of UCD for the benefit of the Project Team.

Existing education work that implemented such a strategy has been carried out at the Bioinformatics Education Conference (http://casb.ucsd.edu/bioed/) and has examined ways to teach bioinformatics to undergraduate biology students. Pevzner and Shamir (2009) discuss that this awareness to teach bioinformatics is not new in a National Research Council report (called "Bio2010", National Research Council 2003). They recommended substantial changes in the mathematics curricula for biology undergraduates. Similar education changes would be made by this research in how they could be taught aspects of software engineering and UCD. The work by Bialek and Botstein (2004) and Pevzner (2004) also recognise the problem of educating biology students and have outlined some approaches to its solution. The solution offered by Pevzner (2004) is to address the shortcomings of many biology departments. Pevzner (2004) call for the introduction of additional required courses of 'Algorithms and Statistics' in the undergraduate molecular biology curriculum. The vision is of a problem-driven course with all examples and problems being biology-motivated. The introduction of the course is to modernise the biology curriculum and to give students computational ideas in molecular biology. Even so, the question of how best to deliver computational ideas to biologists remains. One of the critical problems highlighted by Pevzner and Shamir (2009) is how bioinformatics is taught as a science that explains the computational ideas; but it should rather be taught as a collection of a 'cookbook-style' recipe that relate to the biological problems the scientists encounter. There are also questions on how to educate software developers about how they work within the scientific software environment. Because of the SSD environment it underlines how a software developer has an increasing opportunity to specialise in SSD. The work by Baber (1982 cited in Downey, 2005) and Parnas (1999 cited in Downey, 2005) reviews how they express that software engineering education is dictated by the requirement to create products that are fit to use. Jackson (1998) argues that software engineers must become specialists in a way similar to physical engineers.

Certain institutions are currently putting in to place such teaching initiatives. For example such an initiative, in July 2010 by the University of Houston in the USA received a $2.4 million grant to fund a new multidisciplinary approach for fighting cancer. Its research combines cancer biology with computational disciplines like computer science, theoretical physics, or chemistry. There is a real realisation that "*all the problems of cancer won't be solved by biology*", said Merkl (2010). It is because of this growing recognition that science

must bring in a wider range of expertise fields if the promise of science is to be realised. In relation to my own research, I echo this requirement and advocate it for that wider expertise of fields in science to include trained specialist scientific software developers and UCD experts.

9.6 Closing words

To conclude, this research has explored in depth a serious and recognised problem – the gap between academic SSD and UCD – and presented an analysis from a rich extended field study of that gap in a real-world SSD setting. Academic SSD in this work has been observed from the dual perspectives of the scientific and SSD domains. The subsequent findings have demonstrated that academic SSD projects must focus not only on SSD and UCD but also on the community of the SSD project. Furthermore, it has presented a clear manifesto for the redesign of SSD to deal with the integration of UCD into the SSD process along with the consideration of this wider SSD project community. The research has proposed a framework (the Project Community Framework) for enacting this manifesto. Society increasingly demands science to provide actionable research findings that address the core problems of our current circumstances, from climate change to disease. Scientific software is a key tool for this endeavour, and improving its usability and sustainability in an era of increasing challenges to the funding available for science is a topic of interest to all – developers, scientists, funders, and society in general. It is hoped the work presented here will provide some assistance in meeting such challenges, both by providing a richer understanding of the roots and nature of the gap, and by developing the beginnings of a philosophical and practical response to it.

References

Abraham, V. C., Taylor, D. L. and Haskins, J. R. (2004). *High content screening applied to large-scale cell biology*. Trends in biotechnology, 22 (1), pp.15-22.

Ackroyd, K. S., Kinder, S. H., Mant, G. R., Miller, M. C., Ramsdale C. A. and Stephenson, P. C. (2008). *Scientific Software Development at a Research Facility*, IEEE Software, 25 (4) (Jul. 2008), pp. 44-51.

ACM. (2009). *Human-Computer Interaction: Definition of HCI.* [Online] Available at: http://old.sigchi.org/cdg/cdg2.html#2_1 [Accessed 21st September, 2012]

Aikio, K-P. (2006). Integrating Usability Engineering with Software Engineering: a preliminary view on aspects surrounding the topic of usability integration.

Aikio, K-P. (2007). Exporting usability knowledge into a small-sized software development organization: a pattern approach. In *Proceedings of the 2007 conference on Human interface: Part I*, Michael J. Smith and Gavriel Salvendy (Eds.). Springer-Verlag, Berlin, Heidelberg, pp. 3-11.

Akrich, M. (1995). User representations: Practices, methods and sociology. In Rip, A., Misa, T. J. and Schot, J. (eds.) *Managing technology in society*. London: Pinter Publishers, pp. 167-184.

Ambati, V. and Kishore, S. P. (2004*). How can academic software research and open source software development help each other?*, *Collaboration, Conflict and Control: The 4th Workshop on Open Source Software Engineering*, W8S Workshop - 26th International Conference on Software Engineering, pp. 5-8.

Anderson, R. J. (1994). *Representations and Requirements: The Value of Ethnography in System Design*, Human-Computer Interaction, 9 (2), pp. 151-182.

Anderson, J., Fleek, F., Garrity, K., and Drake, F. (2001). *Integrating usability techniques into software development*. IEEE Software, 18 (1), pp.46-53.

Angrosino, M. (2007). *Doing ethnographic and observational research*. London: Sage Publications.

Antikainen, M., Aaltonen, T. and Väisänen, J. (2007). *The role of trust in OSS communities - Case Linux Kernel community*, in IFIP International Federation for Information Processing Open Source Development, Adoption and Innovation, 234, pp. 223-228.

Bannon, L. J. (1991). *From human factors to human actors: the role of psychology and human-computer interaction studies in system design*. In J. Greenbaum, & M. Kyng (Eds.), *Design at Work: Cooperative Design of Computer Systems* (pp. 25-44). Hillsdale, NJ: Lawrence Erlbaum Associates

Basili, V., Daskalantonakis, M. and Yacobellis, R. (1994). *Technology Transfer at Motorola*, IEEE Software, 11(2), pp.70-76, March 1994.

Basili, V., Shull, F., Lanubile, F., (1999). *Building Knowledge through Families of Experiments*, IEEE Transactions on Software Engineering, 25(4), pp. 456-473.

Basili, V., Donzelli, P., Asgari, S., (2004). *A Unified Model of Dependability: Capturing Dependability in Context*, IEEE Software, 21(6) pp. 19-25.

Basili, V. R., Cruzes, D., Carver, J. C., Hochstein, L. M., Hollingsworth, J. K., Zelkowitz, M. V. and Shull, F. (2008). *Understanding the High-Performance-Computing Community: A Software Engineer's Perspective*, IEEE Software, 25 (4), pp. 29-36.

Bass, L., John, B. E. and Kates, J. (2001). *Achieving usability through software architecture*. Technical Report CMU/SEI-2001-TR-005, Software Eng. Inst., Carnegie Mellon Univ.

Battle, L. (2005). "Patterns of integration: Bringing user centered design into the software development lifecycle", in A. Seffah (eds.) *Human-Centred Software Engineering – Integrating Usability in the Development Process*, pp. 287-308, 2005 Springer, Netherlands.

Baxter, S. M., Day, S. W., Fetrow, J. S. and Reisinger, S. J. (2006). *Scientific Software Development Is Not an Oxymoron*. J. McEntyre, ed. PLoS Computational Biology, 2(9), pp. 4.

Beck, K. Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J. and Thomas, D. (2001). *Manifesto for Agile Software Development.* [Online] Available at: http://www.agilemanifesto.org/ [Accessed 13th October, 2010].

Beck, K. (2003). *Test-Driven Development by Example*, Wokingham: Addison-Wesley.

Beck, K. (2004). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Bell, G., Hey, T. and Szalay, A. (2009). *Beyond the Data Deluge, Science*, 323(5919), pp. 1297–1298.

Bellotti, V., Ducheneaut, N., Howard, M., and Smith, I., (2003). *Taking email to task: the design and evaluation of a task management centered email tool*. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 345-352.

Benner, K. M., Feather, M. S., Johnson, W. L., and Zorman, L. A., (1993). *Utilizing scenarios in the software development process*. Information system development process, *30*, pp.117-134.

Bentley, R., Hughes, J. A., Randall, D., Rodden, T., Sawyer, P., Shapiro, D., and Sommerville, I. (1992). *Ethnographically-informed systems design for air traffic control*, Proceedings of the 1992 ACM Conference on Computer-Supported Cooperative Work, CSCW '92. ACM, New York, NY, pp. 123-129.

Bergstrom, S. and Raberg, L. (2004). *Adopting the Rational Unified Process: Success*

with the RUP, Wokingham: Addison-Wesley.

Beyer, H. and Holtzblatt, K. (1998). *Contextual Design: Defining Customer-Centered Systems*, London: Academic Press.

Bialek, W. and Botstein, D. (2004). *Introductory science and mathematics education for 21st-Century biologists. Science*, 303(5659), pp.788-790.

Billingsley, P. (1995). *Starting From Scratch: Building a Usability Program at Union Pacific Railroad*. Interactions October 1995. ACM Press.

Bird, C., Murphy, B., Nagappan, N., and Zimmermann, T., (2011). *Empirical software engineering at Microsoft Research*. In Proceedings of the ACM 2011 conference on Computer supported cooperative work. ACM, pp.143-150.

Blom, M. (2010). *Is scrum and XP suitable for CSE development?* Procedia Computer Science, 1, 1, pp. 1511-1517.

Blomberg, J. (2007). *Some Perspective*. Fieldwork for Design, 3, Springer London, pp. 59-88.

Bloomer, S., Croft, R. and Kieboom, H. (1997). *Strategic Usability: Introducing Usability into Organizations*. CHI.

Blythe, M. A., Overbeeke, K., Monk, A. F. and Wright, P. C. (2005). *Funology: From Usability to Enjoyment*. Kluwer Academic Publishers, Norwell, MA, USA.

Bødker, S., (2006). *When second wave HCI meets third wave challenges*. In Proceedings of the 4th Nordic conference on Human-computer interaction: changing roles, pp. 1-8.

Boehm, B. (1986). *A Spiral Model of Software Development and Enhancement*, ACM SIGSOFT Software Engineering Notes, 11 (4), pp.14-24.

Boehm, B. (1988). *A spiral model of software development and enhancement*, IEEE Computer, 21 (5), pp. 61-72.

Boivie, I., Gulliksen, J. and Goransson, B. (2006). *The lonesome cowboy: A study of the usability designer role in systems development*. Interacting with Computers, 18(4), 601-634.

Bolchini, D., Finkelstein, A., Perrone, V. and Nagl, S. (2009). *Better bioinformatics through usability analysis*, Bioinformatics, 25, pp. 406–412.

Brooks, F. P. (1995). *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition*, 2nd edition, Addison-Wesley Professional.

Burke, J., and Kirk, A., (2001). *Ethnographic methods. Choosing Human- Computer Interaction (HCI) Appropriate research methods* (CHARM). [Online] [Accessed 17th November, 2012]. Available from: http://otal.umd.edu/hci-rm/ethno.html

Bygstad, B., Ghinea, G., and Brevik, E. (2008). *Software development methods and*

usability: Perspectives from a survey in the software industry in Norway. Interacting with Computers, 20(3), pp.375–385.

Card, S. K., Moran, T. P. and Newell, A. (1983). *The Psychology of Human-Computer Interaction*, Hillsdale: Erlbaum Associates.

Carpenter, A. E., Kamentsky, L. and Eliceiri, K. W. (2012). *A call for bioimaging software usability*. Nat Methods, 9(7), pp. 666-670.

Carroll, J. M. (2000). *Making Use: Scenario-Based Design of Human-Computer Interactions*, Cambridge: MIT Press.

Carroll, J. (2002). *Human-computer interaction in the new millennium*, New York: Boston.

Carroll, J. M. (ed.) (2003). *HCI Models, Theories, and Frameworks: Toward a Multidisciplinary Science,* Interactive Technologies, Morgan Kaufmann.

Carroll, J. M. (2009). 'Human Computer Interaction (HCI)', in Soegaard, Mads and Dam, Rikke Friis (eds.). *Encyclopedia of Human-Computer Interaction*. Aarhus, Denmark: The Interaction-Design.org Foundation. [Online] Available at: http://www.interaction-design.org/encyclopedia/human_computer_interaction_hci.html.

Carver, J. C., Kendall, R. P., Squires, S. E. and Post, D. E. (2007). *Software Development Environments for Scientific and Engineering Software: A Series of Case Studies*, 29th International Conference on Software Engineering (ICSE'07), pp. 550-559.

Chan, S. S., Wolfe, R. J. and Fang, X. (2003). *Issues and strategies for integrating HCI in masters level MIS and e-commerce programs*. International Journal of Human-Computer Studies.

Charmaz, K. (2006) *Constructing grounded theory*. London: Sage Publications.

Chilana, P. K., Wobbrock, J. O. and Ko, A. J. (2010). *Understanding usability practices in complex domains*, Proceedings of the 28th international Conference on Human Factors in Computing Systems (Atlanta, Georgia, USA, April 10 - 15, 2010). CHI '10. ACM, New York, pp. NY, 2337-2346.

ClearSy (2011). *B Method Web Site.* [Online] Available at: http://www.bmethod.com/ [Accessed 15th April, 2011]

Cohn, M. (2005). *Agile Estimating and Planning*, Prentice Hall PTR.

Collins, T. J. (2007). *ImageJ for microscopy*. Biotechniques, 43(1), pp. 25-30.

Cooper, A. (1999). *The inmates are running the asylum: Why high tech products drive us crazy and how to restore the sanity*, Indiana:Sams.

Constantine, L. L. and Lockwood, L. A. D. (1999). *Software for use: a practical guide to the essential models and methods of usage-centered design*. Reading, MA: Addison-Wesley.

Constantine, L. L. (2002). *Process agility and software usability: Toward lightweight usage-centered design*. Information Age, 8(8), pp.1-10.

Constantine, L. L. and Lockwood, L. A. D. (2002). User-centered engineering for web applications. *IEEE Software*, 19(2), pp. 42–50.

Constantine, L. L. and Windl, H. (2003). *Usage-centered design: scalability and integration with software engineering*. In C. Stephanidis and J. Jacko (Eds.) Human-Computer Interaction: Theory and Practice. Proceedings of the 10th International Conference on Human-Computer Interaction, New Jersey: Lawrence Erlbaum Associates, 2003.

Constantine, L. L., Biddle, R., and Noble, J. (2003, May). Usage-centered design and software engineering: models for integration. In IFIP Working Group (Vol. 2, No. 13.4, pp. 3-10).

Crabtree, A. (2003). *Designing collaborative systems: A practical guide to ethnography*. London: Springer.

Crabtree, A. and Rodden, T. (2004). *Domestic Routines and Design for the Home*. Computer Supported Cooperative Work, v.13, n.2, pp.191-220.

Crabtree, A., Rodden, T., and Mariani, J., (2004). *Collaborating around collections: informing the continued development of photoware*. In Proceedings of the ACM conference on Computer supported cooperative work, ACM, pp. 396-405.

Crabtree, C. A., Koru, A. G., Seaman, C. and Erdogmus, H. (2009). *An empirical characterization of scientific software development projects according to the Boehm and Turner model: A progress report*, Proceedings of the 2009 ICSE Workshop on Software Engineering for Computational Science and Engineering, SECSE. IEEE Computer Society, Washington, DC, pp. 22-27.

DeMarco, T. and Lister, T. (1999). *Peopleware: Productive Projects and Teams*, 2nd edition, Dorset House Publishing Company, Incorporated.

Denaro, G., and Pezze, M., (2002). *An Empirical Evaluation of Fault-Proneness Models*. In Proceedings of the 24th International Conference on Software Engineering, ACM, pp. 241-251.

De Roure D. and Goble, C. (2009). *Software Design for Empowering Scientists,* IEEE Software 26 (1), pp. 88-95.

De Roure, D., Goble, C., Aleksejevs, S., Bechhofer, S., Bhagat, J., Cruickshank, D., Fisher, P., Kollara, N., Michaelides, D., Missier, P., Newman, D., Ramsden, M., Roos, M., Wolstencroft, K., Zaluska, E. and Zhao, J. (2010). *The Evolution of myExperiment*, Sixth IEEE e–Science conference (e-Science 2010), December 2010, Brisbane, Australia.

Desel, J. and Juhás, G. (2001). *What Is a Petri Net*? *Informal Answers for the Informed Reader. In* Unifying Petri Nets, 2128, pp.1-25.

Dey, I. (2003) *Qualitative data analysis a user-friendly guide for social scientists*, London: Routledge.

Diaper, D. (1989) (ed.) *Task Analysis for Human-Computer Interaction*. Ellis-Horwood

Diaper, D. and Sanger, C. (2006). *Tasks For and Task In Human- Computer* Interaction. Interacting with Computers, 18(1), pp.117- 138.

Dix, A., Finlay, J., Abowd, G. and Beale, R. (1993). *Human-Computer Interaction*, Prentice Hall.

Dourish, P. (2006). 'Implications for design', in Grinter, R., Rodden, T., Aoki, P., Cutrell, E., Jeffries, R. and Olson, G. (ed.) *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Montréal, Québec, Canada, April 22 - 27, 2006, CHI '06. ACM, New York, NY, 541-550.

Dourish, P. (2007). *Responsibilities and implications: further thoughts on ethnography and design*. In Proceedings of the 2007 conference on Designing for User eXperiences. ACM, pp. 2-16.

Dourish, P. (2012). *Reading and Interpreting Ethnography*, [Online] Available at: http://www.gillianhayes.com/Inf231F12/wp-content/uploads/2012/10/ethnography-ways-submit.pdf [Accessed 14th October, 2012].

Downey, J. (2005). *A framework to elicit the skills needed for software development,* Proceedings of the 2005 ACM SIGMIS CPR Conference on Computer Personnel Research, Atlanta, Georgia, USA, April 14 - 16, 2005, SIGMIS CPR '05. ACM, New York, NY, 122-127.

Dubberly, H., Pangaro, P. and Haque, U. (2009). *On Modelling What is interaction? Are there different types*? Interactions, 16(1), pp.69–75.

Easterbrook, S. M. and Johns, T. C. (2009). *Engineering the Software for Understanding Climate Change*. Computing in Science and Engineering, 11 (6), pp. 64-74.

Eisenstein, M. (2006). *Tower of Babel – OME file format*. Nature, 443 (7114), pp. 1021.

Emmott S. et al. (2006). Towards 2020 Science, Technical report, Microsoft, [Online] Available: http://research.microsoft.com/en-us/um/cambridge/projects/towards2020 science/downloads/T2020S_ReportA4.pdf [Accessed 27th October, 2010].

EPSRC. (2010). *RCUK review of e-Science 2009 Town Meeting Transcript*, [Online] Available at: http://www.epsrc.ac.uk/SiteCollectionDocuments/transcripts/transcript-review_of_e-science_town_meeting.pdf [Accessed 3rd November, 2010].

EPSRC. (2011). *Introduction to e-science programme.* [Online] Available at: http://www.epsrc.ac.uk/about/progs/rii/escience/Pages/intro.aspx [Accessed 10th March, 2011]

e-Science Usability Task Force. (2011). *Usability Research Challenges in e-Science*. [Online] Available at: http://www.cs.nott.ac.uk/~tar/UTF.pdf [Accessed 12th April,

2011].

Faulk, S., Loh, E., Van De Vanter, M. L., Squires, S. and Votta, L. G. (2009). *Scientific Computing's Productivity Gridlock - How Software Engineering Can Help*, Computing in Science and Engineering, 11 (6), pp. 30-39.

Faulkner, X. (2000). *Usability Engineering*. Palgrave Publishers Ltd. New York.

Faulkner, X. and Culwin, F. (2000). *Enter the Usability Engineer: Integrating HCI and Software Engineering*. ITicSE 2000 7/00 Helsinki, Finland. ACM Press.

Fellenz, C. (1997). *Introducing Usability into Smaller Organizations*. ACM 4, pp. 29–33.

Fitzgerald, B., (1998). *An empirically-grounded framework for the information systems development process*, In Proceedings of the international conference on Information systems, pp.103-114.

de la Flor, G., Jirotka, M., Lloyd, S. and Warr, A. (2010). *Embedding e−Research Applications: Designing for Usability*, In Dutton, W. and Jeffreys, P. (eds). World Wide Research: Reshaping the Sciences and Humanities. Cambridge, MA: MIT Press.

Fong, A., Valerdi, R. and Srinivasan, J. (2007). *Boundary Objects as a Framework to Understand the Role of Systems Integrators*. Systems Research Forum, 2(01), pp.11.

Fry, J. and M. Thelwall (2006). *Using Domain Analysis and Organisational Theory to Understand E- Science Sustainability*. NCeSS 2nd International Conference on e-Social Science.

Gabriel R. P. and Goldman, R. (2002). Open Source: Beyond the Fairytales [Online] Available at: http://opensource.mit.edu/papers/gabrielgoldman.pdf. [Accessed 15th November, 2010]

Glaser, B. (1963). *The use of secondary analysis by the independent researcher*. The American Behavioural Scientist, 6, pp. 11–14.

Glaser, B. (1998). *Doing grounded theory*. Mill Valley, CA: Sociology Press.

Glass, R., (1995). *A structure-based critique of contemporary computing research*, Journal of Systems and Software, 28(1), pp. 3-7.

Goffman, E. (1974). *Frame Analysis: An Essay on the Organization of Experience*, New York: Harper and Row.

Göransson, B. Gulliksen, J. and Boivie, I. (2003). *The Usability Design Process – Integrating User-centered Systems Design in the Software Development Process*. Software Process Improvements and Practice. Vol.8, no. 2, pp. 111-131.

Gould, J. D. and Lewis, C. (1985). *Designing for usability: Key principles and what designers think*, Communications of the ACM, 28, 3, pp. 300–311.

Gould, J. D., Boies, S. J. and Ukelson, J. (1997). How to design usable systems. In

Helander, M. G., Landauer, T. K., and Prabhu, P. V. (Eds.), Handbook of human-computer interaction, 2nd ed., 231-254. Amsterdam, The Netherlands: North-Holland.

Grudin, J. (1991). *Systematic sources of suboptimal interface design in large product development organizations*. Hum.-Comput. Interact, 6 (2), pp. 147-196.

Grudin, J. (2006). *Is HCI homeless?: in search of inter-disciplinary status*. Interactions, 13, 1 (January 2006), pp. 54-59.

Guion, L. A., Diehl, D. C., McDonald, D. (2011). Triagulation: establishing the validity of qualitative studies. [online]. [Accessed 8 November 2012]. Available from: http://edis.ifas. ufl.edu/fy394

Guowu, X., Chen, J. and Neamtiu, I. (2009). *Towards a better understanding of software evolution: An empirical study on open source software*, Software Maintenance, IEEE International Conference on. IEEE.

Gulliksen, J., Göransson, B., Boivie, I., Blomkvist, S., Persson, J. and Cajander, Å. (2003). *Key principles for user-centred system design*, Behaviour & Information Technology, 22(6), pp. 397-409.

Gulliksen, J, Boivie, I. Persson, J., Hektor, A. and Herulf, L. (2004). *Making A Difference – A Survey of the Usability Profession in Sweden*. In Hyrskykari A. (ed.), Proceedings of the 3rd Nordic Conference on Human Computer Interaction, NordiCHI, ACM Press, pp. 207-215.

Gunther, R., Janis, J. and Butler, S. (2001). *The UCD Decision Matrix: How, When, and Where to Sell User-Centered Design into the Development Cycle*. [Online] Available at http://www.ovostudios.com/upa2001/ [Accessed 29th June, 2011].

Hall, J.K. (2004). *Language learning as an interactional achievement*, The Modern Language Journal, *88*, pp. 606-612.

Hannay, J. E., MacLeod, C., Singer, J., Langtangen, H. P., Pfahl, D., and Wilson, G. (2009). *How do scientists develop and use scientific software*?, in Proceedings of the 2009 ICSE Workshop on Software Engineering For Computational Science and Engineering (May 23 - 23, 2009), SECSE. IEEE Computer Society, Washington, DC, pp. 1-8.

Harning, B., and Vanderdonckt, J. (2003). Closing the Gaps: Software Engineering and Human-Computer Interaction, Proceedings of IFIP INTERACT03: Human-Computer Interaction

Hartswood, M., Procter, R., Rouncefield, M., & Slack, R. (2003). *Making a Case in Medical Work: Implications for the Electronic Medical Record*. Computer Supported Cooperative Work (CSCW), *12*(3), pp. 241-266 Kluwer Academic.

Harvey, L, J., and Myers, M., (2002). *Scholarship and Practice: The Contribution of Ethnographic Research Methods to Bridging the Gap*, In Qualitative Research in Information Systems: A Reader', M.D. Myers and D.E. Avison (eds.), Sage Publications, London, pp. 169-180.

Hatton, L. (1997). *The T Experiments: Errors in Scientific Software*, IEEE Computational Science and Engineering, 4 (2), pp. 27-38.

Heaton, J. (1998). *Secondary analysis of qualitative data*. Social Research Update (22), Department of Sociology, University of Surrey, [Online] Available at: http://sru.soc.surrey.ac.uk/SRU22.html [accessed 13 September 2012]

Hinds, P. S., Chaves, D. E., and Cypess, S. M. (1992). *Context as a source of meaning and understanding*. Qualitative health research, *2*(1), pp. 61-74.

Hinds P. S., Vogel R, J., and Clarke-Steffen, L., (1997). *The possibilities and pitfalls of doing a secondary analysis of a qualitative dataset*. Qualitative Health Research 7(3), pp. 408–424.

Hine, C. (2006). *Databases as Scientific Instruments and Their Role in the Ordering of Scientific Work*, Social Studies of Science, 36 (2), pp. 269-298.

Hix, D. and Hartson, H. R. (1993). *Developing User Interfaces: Ensuring Usability Through Product and Process*. John Wiley & Sons, New York.

Holtzblatt, K., Wendell, J. B., and Wood, S. (2005). *Rapid contextual design: A how-to guide to key techniques for user-centered design*. San Francisco, CA: Morgan Kaufmann.

Holtzblatt, K. and Beyer, H. R. (2013). *Contextual Design*. In: Soegaard, Mads and Dam, Rikke Friis (eds.). "The Encyclopedia of Human-Computer Interaction, 2nd Ed.". Aarhus, Denmark: The Interaction Design Foundation. [Online] Available at http://www.interaction-design.org/encyclopedia/contextual_design.html [Accessed 20th October, 2012].

Howison, J. and Herbsleb, J. D. (2011). *Scientific Software Production: Incentives and Collaboration*. Physics, pp. 513-522.

Hughes, J. A., Randall, D. and Shapiro, D. (1993). *From Ethnographic Record to System Design: Some Experiences From the Field,* Computer Supported Cooperative Work (CSCW): An International Journal, 1 (3), pp. 123-141.

Hughes, J., King, V., Rodden, T. and Andersen, H. (1994). *Moving out from the control room: ethnography in system design*, Proceedings of the 1994 ACM conference on Computer supported cooperative work (CSCW '94). ACM, New York, NY, USA, 429-439.

Hughes, J., King, V., Rodden, T. and Anderson, H. (1995). *The role of ethnography in interactive systems design*, Interactions, 2 (2), pp. 56-65.

Hutchins, E. (1995). *Cognition in the wild,* Cambridge: MIT Press.

Hvannberg, E. T. (2009). 'Cause and Effect in User Interface Development Human-Centered Software Engineering', in Seffah, A., Vanderdonckt, J. and Desmarais, M. C. (eds.), Springer London, Volume II, pp. 201-222.

IBM. (2012). *User-Centered Design* [Online] Available at: http://www-

01.ibm.com/software/ucd/ucd.html [Accessed 23rd August, 2012].

ifip (2011) 5th Workshop on Software & Usability Engineering Cross-Pollination: Patterns, Usability and User Experience [Online] Available at: http://wwwswt.informatik.uni-rostock.de/PUX2011/ [Accessed 22th September, 2011]

Iivari, N. (2006). *Representing the User, Software Development - a cultural analysis of usability work in the product development context.* Interacting with Computers, 18 (4), pp. 635-664.

Iivari, J. and Iivari, N. (2006). *Varieties of User-Centeredness*. Proceedings of the 39th Annual Hawaii International Conference on System Sciences HICSS06, 00(C), pp.176a-176a.

Jacobson, I., Christerson, M., Jonsson, P. and Övergaard, G. (1992). *Object-oriented software engineering - a use case driven approach*. Addison-Wesley.

Javahery, H., Seffah, A., and Radhakrishnan, T. (2004). *Beyond power: making bioinformatics tools user-centered*. Commun. ACM 47, (11), pp 58-63.

Jerome, B. and Kazman, R. (2005). 'Surveying The Solitudes:An Investigation into the Relationships between Human Computer Interaction and Software Engineering in Practice', in Seffah, A., Gulliksen, J. and Desmarais M. (ed.) *Human-Centered Software Engineering - Integrating Usability in the Development Process*, Kluwer Academic Press.

John, B. E., Bass, L., Sanchez-Segura M. I. and Adams R. J. (2004). Bringing Usability Concerns to the Design of Software Architecture. In *Proceedings of EHCI-DSVIS'04: The 9th IFIP Working Conference on Engineering for Human-Computer Interaction and the 11th International Workshop on Design, Specification and Verification of Interactive Systems*. Hamburg, Germany, July 11-13.

Kalawsky, R. S., Holmes, I. R., O'Brien, J., Lewin, M., Armitage, S. and Goonetilleke, T. S. (2006). Human Factors Audit of Selected e-Science Projects : Joint Information Systems Committee, [Online] Available at: http://syseng.lboro.ac.uk/JISC%20HF%20Audit%20-%20Final%20Report%20Issue%204.pdf [Accessed 15th May, 2011].

Kane, D. (2003a). Finding a Place for Discount Usability Engineering in Agile Development: Throwing Down the Gauntlet. In *Proceedings of the Conference on Agile Development* (ADC '03). IEEE Computer Society, Washington, DC, USA, pp. 40.

Kane, D. (2003b). Introducing Agile Development into Bioinformatics: An Experience Report. In *Proceedings of the Agile Development Conference* (Salt Lake City, USA) ADC'03, IEEE Computer Society Washington, DC, USA, pp. 132-139.

Kane, D. W., Hohman, M. M., Cerami, E. G., McCormick, M. W., Kuhlmman, K. F. and Byrd, J. A. (2006). *Agile methods in biomedical software development: a multi-site experience report*, BMC Bioinformatics, 30 (7), pp. 273.

Kapor, M. (1996). *A software design manifesto*. In T. Winograd, ed. Bringing Design to Software. Addison-Wesley, pp. 1-6.

Karat, J., and Dayton, T., (1995). Practical education for improving software usability. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '95), Katz, I. R., Mack, R., Marks, L., Rosson, M. B. and Nielsen, J. (Eds.). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, pp. 162-169.

Karat, J. (1996). *User Centered Design: Quality or Quackery*?, interactions, 3 (4), pp.18-20.

Karat, J. (1997). *Evolving the scope of user-centered design*, Commun. ACM, 40 (7), pp. 33-38.

Katz-Haas, R. (1998) *User-centered design and web development.* [Online] Available at: http://www.stcsig.org/usability/topics/articles/ucd%20_web_devel.html [Accessed 24th November 2012]

Kazman, R. Gunaratne, J. and Jerome, B. (2003). *Why Can't Software Engineers and HCI Practitioners Work Together*?, Human-Computer Interaction Theory and Practice - Part 1, pp. 504-508.

Kazman, R. and Bass, L. (2003). *Special issue on bridging the process and practice gaps between software engineering and human–computer interaction*, Software Process: Improvement and Practice, 8 (2), pp. 63–65.

Keinonen, T. (2008). *User-centered design and fundamental need*, Proceedings of the 5th Nordic Conference on Human-Computer interaction: Building Bridges, Lund, Sweden, October 20-22, 2008), NordiCHI '08, vol. 358. ACM, New York, NY, pp. 211-219.

Kelly, D. F. (2007). *A Software Chasm: Software Engineering and Scientific Computing*, IEEE SOFTWARE, New York: IEEE Computer Society, 24 (6), pp 120-119.

Kelly, D. and Smith, S. (2009). 2nd CASCON Workshop on Software Engineering for Science. In Proceedings of the 2009 Conference of the Center for Advanced Studies on Collaborative Research, Martin, P. Kark, A.W. and Stewart, D. (Eds.). ACM, New York, NY, USA, pp. 345-347.

Kensing, F. and Blomberg, J. (1998). *Participatory Design: Issues and Concerns*, Comput. Supported Coop. Work 7, 3-4 (Jan. 1998), pp. 167-185.

Kerr, J. M. and Hunter, R. (1993). *Inside RAD: How to Build a Fully-Functional System in 90 Days or Less*, New York: McGraw-Hill.

Killcoyne, S. and Boyle J. (2009). *Managing Chaos: Lessons Learned Developing Software in the Life Sciences*, Computing in science & engineering, 11 (6), pp. 20-29.

Kirwan, B. and Ainsworth, L. K. (eds.), 1992. *A Guide to Task Analysis*. London: Taylor & Francis, Ltd.

Kitchenham, B., Pfleeger, S., Pickard, L., Jones, P., Hoaglin, D., Emam, K. and

Rosenberg, J. (2002). *Preliminary guidelines for empirical research in software engineering*, IEEE Trans. Softw. Eng, pp. 721-734.

Kling, R., and McKim, G. (2000). *Not just a matter of time: Field differences in the shaping of electronic media in supporting scientific communication*, Journal of the American Society for Info. Science, 51(14), pp. 1306-1320.

Kruchten, P. (1998). *The Rational Unified Process: An Introduction*, 3rd ed., Wokingham: Addison-Wesley Professional.

Kubicki, S. and Halin, G. (2010). *Usage-centered design of adaptable visualization services - Application to cooperation support services system in the AEC sector*, Exploring Services Science.

Kujala, S. (2003). *User involvement: a review of the benefits and challenges*, Behaviour & Information Technology, 22 (1), pp. 1-16.

Kuutti, K. (2001). *Hunting for the lost user: From sources of errors to active actors and beyond*. Cultural usability seminar, University of Art and Design Helsinki.

Kyng, M. (1995). *Making representations work*. Communications of the ACM, 38(9), pp.46-55.

Latour, B. and Woolgar, S. (1979). *Laboratory life: The social construction of scientific facts*, London: SAGE.

Latour, B. (1987). *Science in Action: How to Follow Scientists and Engineers Through Society*, Cambridge: Harvard University Press.

Latour, B. (1999). *Pandora's hope: essays on the reality of science studies*, Cambridge: Harvard University Press.

Larman, C. and Basili, V. R. (2003). *Iterative and Incremental Development: A Brief History*, IEEE Computer (IEEE Computer Society), 36 (6), pp 47–56. doi:10.1109/MC.2003.1204375. ISSN 0018-9162.

Lazar, J., Feng, J. and Hochheiser H. (2010). *Research methods in human-computer interaction*, Chichester, West Sussex, U.K.: Wiley.

LeCompte, M.D., and Goetz, J.P. (1982). *Problems of reliability and validity in educational research*, Review of Educational Research, 52(2), pp. 31-60.

Lesser, E.L. and Storck, J. (2001). *Communities of practice and organizational performance*. IBM Systems Journal, 40(4), pp.831-841.

Letondal C. and Mackay W. E. (2004). *Participatory Programming and the Scope of Mutual Responsibility: Balancing scientific, design and software commitment*, in Proceedings of PDC 2004 (Participatory Design Conference), July 27 -31, 2004 - Toronto, Canada.

Letondal, C. (2005). 'Participatory programming: developing programmable bioinformatic tools for end-users', in Lieberman, H., Paterno, F. and Wulf, V. (ed.) *End*

user development, Springer, pp. 207-242.

Letondal, C. (2006). *Participatory Programming: Developing Programmable Bioinformatics Tools for End-Users*, In End User Development, Eds Lieberman, H., Fabio Paternò, F. and Wulf, V. pp. 206-242.

Liu, D., Xu, S. and Brockmeyer, M. (2008). *Investigation on Academic Research Software Development*, 2008 International Conference on Computer Science and Software Engineering, 2, pp. 626-630.

Losada, B., Urretavizcaya, M., López-Gil, J. M. and Fernández-Castro, I. (2012). *Combining InterMod agile methodology with usability engineering in a mobile application development*, Proceedings of the 13th International Conference on Interacción Persona-Ordenador (INTERACCION '12), ACM, New York, NY, USA, Article 39.

Macaulay, C., Sloan, D., Jiang, X., Forbes, P., Loynton, S., Swedlow, J. R. and Gregor, P. (2009). *Usability and User-Centered Design in Scientific Software Development*, IEEE Software, 26 (1), pp. 96-102.

Macmillan Publishers Limited, (2013). |*Enterprise software,* [Online] Available: http://www.macmillandictionary.com/thesaurus/british/enterprise-software [Accessed 30th September 2012]

Maguire, M (2001). *Context of use within usability activities*, International Journal of Human-Computer Studies, 55 (4), pp. 453-483.

Mao, J. and Vredenburg, K. (2001). User centred design methods in practice: A survey of the state of the art, In 11th IBM centres for advanced studies conference.

Mao, J., and Vredenburg, K., Smith, P. W. and Carey, T. (2005). *The state of user-centered design practice*. Communications. ACM 48(3) pp. 105-109.

Mayhew, D. J. (1996). *Managing User Interface Design*, InContext Enterprises Incorporated.

Mayhew, D. J. (1998). *The usability engineering lifecycle*. In CHI 98 Conference Summary on Human Factors in Computing Systems. ACM, pp. 127-128.

Mayhew, D. J. (1999). *The Usability Engineering Lifecycle: A Practitioner's Handbook to User Interface Design*, San Francisco: Morgan Kaufmann Publishers Inc.

Mayhew, D. J. (2008) *User Experience Design: The Evolution of a Multi-Disciplinary Approach*, Journal of Usability Studies, 3(3), pp. 99-102.

McCann, T. and Clark, E. (2003). *Grounded theory in nursing research: Part 2 - Critique*, Nurse Researcher, 11 (2), pp. 19-28.

McCarthy, J. and Wright, P. (2004). *Technology as Experience*. The MIT Press

McInerney, P. and Maurer, F. (2005). *UCD in agile projects: dream team or odd couple*? Interactions, 12(6), p.19-23.

McGrath, O. G. (2006). *Balancing act: community and local requirements in an open source development process*. SIGUCCS Proceedings of the 34th annual ACM SIGUCCS, pp. 240-244.

McNally, J. G., Karpova, T., Cooper, J. and Conchello J. A. (1999). *Three-dimensional imaging by deconvolution microscopy*, Methods, 19(3), pp. 373–385.

Merkl, L. (2010). *Biology, Computer Science Combine Efforts to Fight Cancer*, [Online], Available: http://www.uh.edu/newsevents/stories/2010articles/July2010/07272010MontePetti CPRIT.php. [Accessed 11th September, 2010].

Metzker, E. and Offergeld, M. (2001). An Interdisciplinary Approach for Successfully Integrating Human-Centered Design Methods into Development Processes Practiced by Industrial Software Development Organizations. In *Proceedings of the 8th IFIP International Conference on Engineering for Human-Computer Interaction* (EHCI '01), Murray Reed Little and Laurence Nigay (Eds.). Springer-Verlag, London, UK, 19-34.

Mohammad, A. F. (2010). *A New Perspective in Scientific Software Development*. In Innovations and Advances in Computer Sciences and Engineering Ed Editor: Sobh, Tarek. Springer Netherlands. pp. 129-134.

Molich, R., and Nielsen, J. (1990). *Improving a human-computer dialogue*, Communications of the ACM, 33(3), pp. 338-34

Mugridge, R. (2003). *Test Driven Development and the Scientific Method*, in Proceedings of the Agile Development Conference (Salt Lake City, USA) ADC'03, IEEE Computer Society Washington, DC, USA, pp. 47-52.

Myers, G. (1990). *Writing Biology: Texts in the Social Construction of Scientific Knowledge*, Madison: University of Wisconsin Press.

Myers, B. A. (1998). *A Brief History of Human Computer Interaction Technology*, ACM interactions, 5(2), pp. 44-54.

National Research Council. (2003). *BIO2010 Transforming Undergraduate Education for Future Research Biologists*, Washington. The National Academies Press.

Nardi, B. A. (1996). *Context and Consciousness: Activity Theory and Human - Computer Interaction*, Cambridge: MIT Press.

Naur, P. and Randell, B. (ed.) (1969). *Software Engineering: Report of a conference sponsored by the NATO Science Committee*, Garmisch, Germany, 7-11 Oct. 1968, Brussels: Scientific Affairs Division, NATO 231.

Newman, M. W. and Landay, J. A. (2000). Sitemaps, storyboards, and specifications: a sketch of Web site design practice. In *Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques* (DIS '00), Boyarski, D. and Kellogg W. A. (Eds.). ACM, New York, NY, USA.

Nielsen, J. (1990). *Big paybacks from 'discount' usability engineering*. IEEE Software, 7, 3, pp. 107-108.

Nielsen, J. (1993). *Usability Engineering.* W. Weber, ed., Academic Press.

Nielsen, J. (1994a). Heuristic evaluation. In Nielsen, J., and Mack, R.L. (Eds.), *Usability Inspection Methods*, John Wiley & Sons, New York, NY.

Nielsen, J. (1994b) *Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier*. [Online] Available at http://www.useit.com/papers/guerrilla_hci.html [Accessed 1st September, 2012].

Nielsen, J. and Mack, R. L. (ed.) (1994). *Usability Inspection Methods*, New York: Wiley.

Norman, D. A. (1981). A psychologist views human processing: Human errors and other phenomena suggest processing mechanisms

Norman, D. A. (1986) User centered system design, Lawrence Erlbaum Associates, Proceedings of the 7th international joint conference on Artificial intelligence, (2) pp. 1097-1101.

Norman, D. A. and Draper, S. W. (1986). *User Centered System Design; New Perspectives on Human-Computer Interaction*. L. Erlbaum Assoc. Inc., Hillsdale, NJ, USA.

Norman, D. A. (2004). *Emotional Design: Why We Love (Or Hate) Everyday Things*, New York: Basic Books.

Norman, D. A. (2007). The Design of Future Things, The MIT Press.

Norman, D. A. (2010). *Living with Complexity*, The MIT Press.

Nuin, P. (2008). 'How to improve scientific software?' in *The Blind Scientist*, [Online], Available: http://blindscientist.genedrift.org/2008/04/23/how-to-improve-scientific-software/.

NSF. (2011). Cyberinfrastructure training, education, advancement, and mentoring for our 21st century workforce program [Online] Available at: http://www.nsf.gov/pubs/2011/nsf11515/nsf11515.htm [Accessed 17th April, 2011]

Oudshoorn, N. E. J. and Pinch, T. J. (2003). *How users matter. The co-construction of users and technology*. MIT Press.

Chilana, P, K., Wobbrock, J, O. and J. (2010). Understanding usability practices in complex domains. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* ACM,

Parnas, D. L (1999). *Software Engineering Programs Are Not Computer Science Programs*. IEEE Software. 16(6), pp. 19-30.

Patton, J. (2002). *Hitting the target: adding interaction design to agile software development*. Conference on Object Oriented Programming Systems, pp.1.

Patton, J. (2003). *Improving On Agility: Adding Usage-Centered Design to a Typical Agile Software Development* [Online] Available at: http://www.agileproductdesign.com/writing/index.html [Accessed 13th May 2011].

Paulson, J. W., Succi G. and Eberlein, A. (2004). An empirical study of open-source and closed-source software products, *Software Engineering, IEEE Transactions*, pp. 246-256.

Pavelin, K., Cham, J. A., de Matos, P., Brooksbank, C., Cameron, G., *et al.* (2012). *Bioinformatics Meets User-Centred Design: A Perspective*, PLoS Comput Biol, 8(7).

Perry, D., Porter, A. and Votta, L., (2000). *Empirical Studies of Software Engineering: A Roadmap*. Proceedings of the Conference on the Future of Software Engineering. (ed.) ACM, pp. 345-356.

Petersen, M. G., Iversen, O. S., Krogh, P. G., and Ludvigsen, M. (2004). *Aesthetic Interaction: a pragmatist's aesthetics of interactive systems*. In Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques, pp. 269-276.

Pevzner, P. A. (2004). *Educating biologists in the 21st century: bioinformatics scientists versus bioinformatics technicians*. Bioinformatics, pp. 2159.161

Pevzner, P. and Shamir, R. (2009). *Computing has changed biology - biology education must catch up*. Science, 325(5940), pp.541-542.

Phillips, C. H. E and Kemp, E. A. (1996). Human- computer interaction in software engineering courses: A discussion summary. *Software Engineering: Education and Practice (SE:E&P'96)*, Dunedin, 24-27 January 1996, pp. 520- 521.

Pitt-Francis, J. O Bernabeu, M. Cooper, J. Garny, A. Momtahan, L. Osborne, J. Pathmanathan, P. Rodriguez, B. Whiteley, J. P. and Gavaghan, D. J. (2008). *Chaste: using agile programming techniques to develop computational biology software.* Phil. Trans. R. Soc. 366(1878), pp. 3111-3136

Poppendieck, M. and Poppendieck, T. (2003). *Lean Software Development: An Agile Toolkit*, Wokingham: Addison-Wesley Professional.

Potts, C., (1995). *Using schematic scenarios to understand user needs*. In: ACM Symposium on Designing Interactive Systems, ACM Press, pp. 247-256.

Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. and Carey, T. (1994). *Human-Computer Interaction*. Addison-Wesley: Wokingham, UK.

Preece, J., Rogers, Y. and Sharp, H. (2002). *Interaction design*, Wiley & Sons.

Pressman, R. S. (2005). *Software engineering: a practitioner's approach*, 6th ed., New York: McGraw-Hill Higher Education

Procter, R., Borgman, C., Bowker, G., Jirotka, M., Olsen, G., Pancake, C., Rodden, T. and Schraefel, M. C. (2006). *Usability Research Challenges for Cyberinfrastructure and*

Tools, Proceedings of ACM CHI 2006 Conference on Human Factors in Computing Systems, Workshops, (2), pp. 1675−1678.

Prowell, S. J., Trammell C. J., Linger R. C. and Poore J. H. (1999). *Cleanroom Software Engineering: Technology and Process*, Wokingham: Addison-Wesley.

Pyla, P. S., Pérez-Quiñones, M. A., Arthur, J. D. and Hartson, H. R. (2004). What we should teach, but don't: Proposal for a cross pollinated HCI-SE curriculum. In *Proc. Frontiers in Education (FIE) Conference*, S1H17-22.

Pyla, P. S., Pérez-Quiñones, M. A., Arthur, J. D. and Hartson, H. R. (2005). *A. Seffah (eds.), Human-Centered Software Engineering – Integrating Usability in the Development Process*, pp. 245–265.

Radle, K. and Young, S. (2001). *Partnering Usability with Development: How Three Organizations Succeeded.* IEEE Software, 18 (1), pp. 38-45.

Randall, D., Rouncefield, M. and Hughes, J. A. (1995). Chalk and cheese: BPR and ethnomethodologically informed ethnography in CSCW. In *Proceedings of the fourth conference on European Conference on Computer-Supported Cooperative Work* (ECSCW'95), Hans Marmolin, Yngve Sundblad, and Kjeld Schmidt (Eds.). Kluwer Academic Publishers.

Rasmussen, J. (1986). *Information Processing and Human-Machine Interaction: An Approach to Cognitive Engineering*, Elsevier Science Inc.

Reed, R. (2005). *Software Engineering: The Past, the Future, and Your TCSE*. IEEE Software. 22(4), pp106-107.

Reiterer, H. (2000). *Tools for Working with Guidelines in Different User Interface Design Approaches*. Annual Workshop of the Special Interest Group on Tools for Working with Guidelines, Biarritz, France, 2000.

Rideout, T., Uyeda, K. and Williams, E. (1989). *Evolving the Software Usability Engineering Process at Hewlett-Packard*. 1989 IEEE.

Roberts, D. (2005). *Coping with Complexity*. In A. Seffah, J. Gulliksen, and M. C. Desmarais (eds.), Human-centered software engineering: Integrating usability in the development process. Springer, Dordrecht, Netherlands, 2005, pp. 201–217.

Robinson, M. and Bannon, L. (1991). *Questioning representations*, in Proceedings of the second conference on European Conference on Computer-Supported Cooperative Work, Kluwer Academic Publishers, pp. 233.

Rockwell, C. (1999). *Customer connection creates a winning product: building success with contextual techniques*. Interactions, 6, (1), pp. 50-57.

Rolland, C., Achour, C. B., Cauvet, C., Ralyté, J., Sutcliffe, A., Maiden, N., and Heymans, P. (1998). *A proposal for a scenario classification framework*. Requirements Engineering, 3(1), 23-47.

Rosenbaum S, Rohn J, Humburg J, Bloomer S, Dye K, Nielsen J, Rinehart D & Wixon

D (1999). What Makes Strategic Usability Fail? Lessons Learned from the Field. A panel. CHI 99 Extended Abstracts, Pittsburgh, USA, ACM, New York**:** 93-94.

Rosenbaum, S., Rohn, J. and Humburg, J. (2000). *A toolkit for strategic usability: Results from workshops, panels, and surveys*. Conference on Human Factors in Computer Systems.

Rosenbaum, S., Chauncey, E., Jokela, W., Rohn, T., Smith, A. and Vredenburg, K. (2002). *Usability in Practice: User Experience Lifecycle – Evolution not Revolution*, Conference on Computer Human Interaction,

Royce, W. W. (1970). *Managing the development of large software systems:Concepts and techniques*, in WESCON Technical Papers. Reprinted in Proceedings of the Ninth International Conference on Software Engineering, 1987, pp. 328–338.

Sanders, R. (2008). *The Development and Use of Scientific Software.* [Online] Available at:http://catspaw.its.queensu.ca/jspui/bitstream/1974/1188/1/Sanders_Rebecca_J_20080 4 _MSc.pdf [Accessed 12th September, 2010]

Sanders, R. and Kelly, D. (2008). *Dealing with risk in scientific software development*, IEEE Software, 25 (4), pp.21–28, July/August 2008.

Schaffer, E. 2004. *Institutionalization of usability: a step-by-step guide*. Addison-Wesley. Pearson Education, Inc. 2004. Boston. ISBN-0-321-17934-X.

Schmidt, K. (2000). *The Critical Role of Workplace Studies in CSCW*, in Luff, P., Hindmarsh, J. and Heath C. (Eds.), Workplace Studies – Recovering Work Practice and Informing System Design, Cambridge: Cambridge University Press.

Schön, D. A. (1973). *Beyond the Stable State*, New York: Norton.

Schön, D. A. (1983). *The Reflective Practitioner: How Professionals Think In Action*, Basic Books,

Schraefel, M. C., Hughes, G. V., Hugo R. M., Smith, G., Payne, T. R. and Frey, J. (2004). *Breaking the book: translating the chemistry lab book into a pervasive computing lab environment*. In Proceedings of the SIGCHI conference on Human factors in computing systems. ACM, New York, NY, USA, pp. 25-32.

Science. (2011). *Challenges and Opportunities*, Introduction to special issue, Science, 331 (6018), pp.692-693. [Online] Available at: https://www.sciencemag.org/content/ 331/6018/692.full.pdf [Accessed 23rd May, 2012].

Seaman, C. (1999) '*Qualitative Methods in Empirical Studies of Software Engineering'*, IEEE Transactions on Software Engineering, 25(4), July/August, pp. 557-572.

Seffah, A. and Metzker, E. (2004). 'The Obstacles and Myths of Usability and Software Engineering', in *Communications of the ACM*, New York: ACM, 47 (12), pp.71-76.

Seffah, A., Desmarais, M. C., and Metzker, E. (2005). 'HCI, usability and software engineering integration: present and future', in Seffah, A., Gulliksen, J., and Desmarais, M. C. (ed.) *Human-centered software engineering: Integrating usability in the*

development process, Dordrecht: Springer, pp. 35-57.

Seffah, A. and Metzker, E. (2008). *Adoption-Centric Usability Engineering: Systematic Deployment, Assessment and Improvement of Usability Methods in Software Engineering* (1 ed.). Springer Publishing Company.

Seffah, A., Vanderdonckt, J. and Desmarais, M. C. (2009). *Human-Centered Software Engineering Software Engineering Models, Patterns and Architectures for HCI*.

Seffah, A., Vanderdonckt, J. and Desmarais, M. C. (2009). 'Human-Centered Software Engineering: Software Engineering Architectures, Patterns, and models for Human Computer Interaction', in Seffah, A., Vanderdonckt, J. and Desmarais, M. C. (ed.).

Seffah, A. and Metzker, E. (2009). *Adoption-centric Usability Engineering*, Systematic Deployment, Assessment and Improvement of Usability Methods in Software Engineering, Springer 2009.

Segal, J. (2001). *Organisational learning and software process improvement: a case study*. In: Althoff, Klaus-Dieter; Feldmann, Raimund L. and Muller, Wolfgang eds. Advances in learning software organizations: Third international workshop, LSO 2001. Lecture notes in Computer Science, XI (2176). Springer, pp. 68–82.

Segal, J. (2004). *The Nature of evidence in empirical software engineering*. In Software Technology and Engineering Practice Eleventh Annual International Workshop on, pp.40-47.

Segal, J. (2005). *When software engineers met research scientists: a case study*, Empirical Software Engineering, 10 (4), pp. 517–536.

Segal, J. (2007). *Some problems of professional end user developers*. Visual Languages and HumanCentric Computing 2007 VLHCC 2007 IEEE Symposium on, pp.111-118.

Segal, J. (2008). *Models of Scientific Software Development*, in Proceedings of the 2008 Workshop on Software Engineering in Computational Science and Engineering, Leipzig: SECSE.

Segal, J. and Morris, C. (2008). *Developing scientific software*, IEEE Software, 25, (4), pp. 18- 20.

Segal, J. (2009). *Some challenges facing software engineers developing software for scientists*, in Proceedings of the 2009 ICSE Workshop on Software Engineering For Computational Science and Engineering (May 23 - 23, 2009), SECSE, Washington: IEEE Computer Society, pp. 9-14.

Segal, J. A. and Morris, C. (2011). *Developing software for a scientific community: some challenges and solutions*, in Leng, J. and Sharrock, W. (eds), Handbook of Research on Computational Science and Engineering: Theory and Practice, USA: IGI Global, pp. 177–196.

Seidel, J, V., (1998). Qualitative Data Analysis, www.qualisresearch.com (originally published as Qualitative Data Analysis, in The Ethnograph v5.0: A Users Guide, Appendix E, 1998, Colorado Springs, Colorado: Qualis Research

Severin, A. J. (2011). *Dealing with data: training new scientists*. Science, 331(6024), pp.1516.

Shankar, K. (2004). *Recordkeeping in the production of scientific knowledge: An ethnographic study*. Archival Science, 4, pp. 367–382.

Shankar, K. (2006). *Recordkeeping in the production of scientific knowledge: An ethnographic study*, Archival Science, 2006, 4, pp. 367-382.

Sharp, H., Woodman, M. and Hovenden, F. (2004). *Tensions in the adoption and evolution of software quality management systems: a discourse analytic approach*, International Journal of Human Computer Studies, 61(2), pp. 219-236.

Sharp, H. Rogers, Y. and Preece, J. (2007). *Interaction Design: Beyond Human-Computer Interaction*. Second Edition. John Wiley.

Shneiderman, B. (1987). *Designing the user interface: Strategies for effective human-computer interaction*. Reading, MA: Addison-Wesley.

Shneiderman, B. (2002). *HCI theory is like the public library*, Posting in CHIplace online discussion forum, Oct 15th 2002, [Online], Available: www.chiplace.org [Accessed 12th September, 2010].

Shneiderman, B. (2008). *Science 2.0*. Science, 319(5868), pp.1349-1350.

Siegel, D. and Dray, S. (2003). Living on the edges: user-centered design and the dynamics of specialization in organizations. *Interactions*, (5). ACM.

Siegel, D. and Dray, S. (2005). *Avoiding the next schism: ethnography and usability*, interactions, 12(2), pp. 58-61.

Sletholt, M. T., Hannay, J. E., Pfahl, D., Benestad, H. C. and Langtangen, H. P. (2011). *A literature review of agile practices and their effects in scientific software development*, in Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering (SECSE '11), ACM, New York, NY, USA, pp. 1-9.

Sletholt, M. T., Hannay, J. E., Pfahl, D., Langtangen, H. P. (2012). *What Do We Know about Scientific Software Development's Agile Practices?* Computing in Science & Engineering, 14(2), pp.24-37.

Sloan, D., Macaulay, C., Forbes, P. and Loynton, S. (2009). *User research in a scientific software development project*, in Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology (BCS-HCI '09), British Computer Society, Swinton, UK, pp. 423-429.

Smith, E. (2008). *Using Secondary Data in Educational and Social Research*, Open University Press.

Software Sustainability Institute. (2011). *The Software Sustainability Institute* [Online] Available at: http://www.software.ac.uk/ [Accessed 27th April, 2011]

Spencer, R. (2000). *The streamlined cognitive walkthrough method, working around social constraints encountered in a software development company*. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems. ACM, pp. 353-359.

Spinuzzi, C. (2003). *Tracing genres through organizations: A sociocultural approach to information design.* MIT Press.

Star, S. L. and Ruhleder, K. (1996). *Steps Toward an Ecology of Infrastructure: Design and Access for Large Information Spaces* J. Yates and J. Van Maanen, eds.Information Systems Research, 7(1), pp.111-134.

Stern, P. N. (2007). *On solid ground: essential properties for growing grounded theory*, In: Bryant A, Charmaz K (Eds) The Sage Handbook of Grounded Theory. Sage Publications, London, pp. 114-126.

Stiemerling, O. Kahler, H. and Wulf, V. (1997). *How to make software softer: Designing tailorable applications*. Designing Interactive Systems, pp. 365–376.

Suchman, L. A. (1987). *Plans and Situated Actions: the Problem of Human-Machine Communication*, Cambridge: Cambridge University Press.

Sutcliffe, A. G. (2011). Requirements Engineering: from an HCI Perspective. In: Soegaard, Mads and Dam, Rikke Friis (eds.). "Encyclopedia of Human-Computer Interaction". Aarhus, Denmark: The Interaction Design Foundation. [Online] Available at http://www.interaction-design.org/encyclopedia/requirements_engineering.html [Accessed 25th August, 2012]

Swedlow, J. R and Eliceiri, K. W. (2009). *Open source bioimage informatics for cell biology*, Trends Cell Biol., 19 (11), pp. 656-660.

Tarpin-Bernard, F., Samaan, K. and David, B. (2009). 'Achieving Usability of Adaptable Software: The Amf-Based Approach Human-Centered Software Engineering', in Seffah, A., Vanderdonckt, J. and Desmarais, M. C., (eds.), Springer London, Volume II, pp. 277-297.

Taylor, A. S. and Swan, L. (2005). *Artful systems in the home.* Conference on Human Factors and Computing systems, pp. 641-650.

Thew, S., Sutcliffe, A., Procter, R., De Bruijn, O., McNaught, J., Venters, C. C. and Buchan I. (2009). *Requirements engineering for e-science: experiences in epidemiology*, IEEE Software, 26 (1), pp. 80- 87.

Thimbley, H. (2000). *On discerning users*, In how to make user centred design usable, Gulliksen, J., Lantz, A. and Bovie, I. Eds., Stockholm: Royal Institute of Technology Stockholm – Centre for User Orientated Design pp. 63-85.

Treviño, A. J. (2003). *Goffman's Legacy*, Lanham, Md: Rowman & Littlefield Publishers.

UPA (2010). *What is UCD* [Online] Available at: http://www.upassoc.org/usability_resources/about_usability/ what_is_ucd.html.

[Accessed 16th October, 2010].

Venturi, G., Troost, J. and Jokela, T. (2006). *People, Organizations, and Processes: An Inquiry into the Adoption of User-Centered Design in Industry*. International Journal of Human- Computer Interaction 21, pp. 219–238.

Vessey, I. (1997). *Problems versus solutions: the role of the application domain in software*. In Papers presented at the seventh workshop on Empirical studies of programmers, S. Wiedenbeck and J. Scholtz (Eds.). ACM, New York, NY, USA, pp. 233-240.

Vredenburg, K., and Butler, M. B., (1996). *Current Practice and Future Directions in User-Centered Design*. Usability Professionals' Association Fifth Annual Conference, Copper Mountain.

Vredenburg, K., Mao, J. Y., Smith, P. W. and Carey, T. (2002). *A survey of user-centered design practice*, Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves (CHI '02), ACM, New York, NY, USA, pp. 471-478.

Wagner, F. R Schmuki, R. and Wolstenholme, P. (2006). *Modeling Software with Finite State Machines*, Auerbach Publications.

Warr, A., de la Flor, G., Jirotka, M. and Lloyd, S. (2007). *Usability in e−Science: The eDiaMoND Case Study*. CHI 2007, San Jose, California.

Weinberg, G. M. (1971). *The Psychology of Computer Programming*, Van Nostrand Reinhold Company.

Wellcome Trust. (2010). *Wellcome Trust Annual Report and Financial Statements,* [Online] Available at: http://www.wellcome.ac.uk/stellent/groups/corporatesite/@msh_publishing_group/documents/web_document/wtx063982.pdf [Accessed 25th September, 2010].

Wenger, E. (2007). *Communities of practice: A brief introduction*, Communities of practice, [Online], Available at: http://www.ewenger.com/theory/. [Accessed 13th September, 2010].

Wenger, E., White, N. and Smith, J. D. (2009). *Digital Habitats: stewarding technology for communities*, Portland: CPsquare.

Weyuker, E. J. (2011). *Empirical Software Engineering Research - The Good, The Bad, The Ugly*, Empirical Software Engineering and Measurement (ESEM), 2011 International Symposium, pp. 1-9.

Whiteside, J., Bennett, J., and Holtzblatt, K. (1988). *Usability engineering: Our experience and evolution*. In Helander, M. (Ed.), Handbook of Human-Computer Interaction, pp. 791-817.

Wilson, G. (2006). *Software Carpentry: Getting Scientists to Write Better Code by Making Them More Productive*, Computing in Science and Engineering, 8 (6), pp. 66-69.

Winograd, T. (1986). *Understanding computers and cognition: a new foundation for design*, Wokingham: Addison-Wesley.

Winograd, T. and Flores, F. (1986) *Understanding computers and cognition*. Norwood, N.J.: Ablex Pub. Corp.

Winograd, T. (1996). *Bringing Design to Software*, ACM Press.

Wixon, D. (2011). *The unfulfilled promise of usability engineering.* Journal of Usability Studies, *6*(4), pp. 198-203.

Wood, W. A. and Kleb, W. L. (2003). *Exploring XP for Scientific Research*, IEEE Software, 20 (3), pp. 30-36.

Wright, P. and McCarthy, J. (2008). *Empathy and experience in HCI*, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08), ACM, New York, NY, USA, pp. 637-646.

Wright, P. and McCarthy, J. (2010). *Experience-Centered Design: Designers, Users, and Communities in Dialogue*, Synthesis Lectures on Human Centered Informatics, 3(1), pp.1-123.

Wyeth, P. (2006). *Ethnography in the kindergarten: examining children's play experiences*. In Proceedings of the SIGCHI conference on Human Factors in computing systems (CHI '06), Grinter, R., Rodden, T. and Aoki, P. Ed Cutrell, Robin Jeffries, and Gary Olson (Eds.). ACM, New York, NY, USA, pp. 1225-1228.

Xie, T., Thummalapenta, S., Lo, D. and Liu, C. (2009). *Data Mining for Software Engineering. IEEE Computer*, 42(8), pp. 35-42.

Yourdon, E. (2007). *Celebrating Peopleware's 20th Anniversary*. IEEE Software, 24(5), pp.96-100.

Zimmerman, J., Forlizzi, J., and Evenson, S. (2007). *Research through design as a method for interaction design research in HCI*. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, pp. 493-502.