2009

TOWARDS A REFLECTIVE-AGILE LEARNING MODEL AND METHOD IN THE CASE OF SMALL-SHOP SOFTWARE DEVELOPMENT: EVIDENCE FROM AN ACTION RESEARCH STUDY

Jeffry Babb
Virginia Commonwealth University

School of Business
Virginia Commonwealth University

This is to certify that the dissertation prepared by Jeffry S. Babb Jr. entitled TOWARDS A REFLECTIVE-AGILE LEARNING MODEL AND METHOD IN THE CASE OF SMALL-SHOP SOFTWARE DEVELOPMENT: EVIDENCE FROM AN ACTION RESEARCH STUDY has been approved by his or her committee as satisfactory completion of the dissertation requirement for the degree of Doctor of Philosophy

Dr. Allen S. Lee, Chair, Virginia Commonwealth University

Dr. Richard R. Redmond, Committee Member, Virginia Commonwealth University

Dr. Gurpreet Dhillon, Committee Member, Virginia Commonwealth University

Dr. H. Roland Weistroffer, Committee Member, Virginia Commonwealth University

Dr. Anson Seers, Committee Member, Virginia Commonwealth University

Dr. Richard R. Redmond, Chair, Department of Information Systems, Virginia Commonwealth University

Dr. Michael Sesnowitz, Dean, School of Business, Virginia Commonwealth University

Dr. F. Douglas Boudinot, Dean of the Graduate School

April 13th, 2009

TOWARDS A REFLECTIVE-AGILE LEARNING MODEL AND METHOD IN THE CASE OF SMALL-SHOP SOFTWARE DEVELOPMENT: EVIDENCE FROM AN ACTION RESEARCH STUDY

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Business at Virginia Commonwealth University.

By

JEFFRY STEPHEN BABB, JR.
Master of Science in Information Systems, Virginia Commonwealth University, 2005
Master of Urban and Regional Planning, Virginia Commonwealth University, 2000
Bachelor of Arts in Geography, University of Hawaii at Manoa, 1991

Director: Dr. Allen S. Lee

Professor of Information Systems

Virginia Commonwealth University
Richmond, Virginia
May 2009

Acknowledgement

When I began my Ph.D. journey, a peer further along in her own journey once told me: “this will fundamentally change your life; you won’t be the same person again.” No truer words have been spoken. However, while this dissertation represents the culmination of one phase of my journey as a scholar, it is also an open door to another phase. It would have been impossible to cross this threshold without the guidance, wisdom and patience of many people. I will humbly attempt to express my gratitude to mentors, family, friends and peers.

Without Dr. Richard Redmond’s guidance, encouragement and trust, I would not be writing this at all. When I needed a voice of reason and a “reality check,” I could count on Dr. Redmond to provide balance and perspective. Dr. Redmond’s faith in my abilities consistently allowed me to renew my own confidence and move forward in those times where the light seemed dim. The time that I have spent in Dr. Redmond’s care will remain cherished and treasured moments for the rest of my days. My debt to Dr. Redmond is a debt I can only hope to pay forward to my future students.

Dr. Allen Lee has been a patient teacher and has opened my mind to scientific inquiry and the philosophy of science. He has been a voice of calm reason no matter how turbulent my own waters were. If it were not for Dr. Lee’s steadfast patience and firm guidance, I would not have been capable of producing this work. Even as a nascent neophyte, Dr. Lee always saw through me, to the road ahead of me, and nudged me in the right direction. Dr. Lee was always there for me and always patient, even during those times where I was not deserving of this patience and was otherwise completely lost.

The mentorship I have received from both Dr. Lee and Dr. Redmond has been critical to my development and I look forward to our continued association as I progress as a scholar, teacher, and as a person. Dr. Lee and Dr. Redmond have also taught me, by their example, what it means to be an outstanding educator. I would find myself searching my own soul and direction regularly as a result of our conversations and I will dearly miss the ability to wander into their offices for enlightenment, encouragement, and friendly discussions.

Dr. Gurpreet Dhillon has been steadfast in his guidance and support throughout my time as a Ph.D. student. I owe a great deal of my awareness of the "mechanics" and craft of scholarship to Dr. Dhillon. Always a supportive and friendly listener, Dr. Dhillon, by example, has instilled in me the importance of having a "can do" and positive attitude in scholarly endeavors.

There are other faculty mentors that have been indispensable to my journey. Dr. Jim Wynne has taught me about the balance between teaching, research and service; Dr. Roland Weistroffer has patiently and persistently encouraged me to engage in scholarship; Dr. Peter Aiken lent me his ear when I needed it; Dr. Margaret "Peg" Williams and Dr. Anson Seers have both been instrumental in developing my research perspective on organizations and learning.

There are also peers and friends who accompany you on a journey such as this. Both Kofi Andoh-Baidoo and Manoj Thomas have provided me with encouragement and have been every bit a role model for me as my faculty mentors have been. Mitch, Angela, Lew, Niki and Santa have all been advocates and friends who were willing to listen when I needed to borrow an ear. I would also thank all of my students for the role they have played in my development; I am ever grateful for the good fortune I've been afforded to be an educator and a lifelong learner.

The last but greatest thank you I owe is to my family. There is no person more deserving of my gratitude than my loving wife, Tina. At times, Tina was my anchor, and yet she was also my sail and the force filling my sails at every step along the way. It is a pitiful understatement to say that this was not possible without her. Therefore, this accomplishment is just as much hers as it is mine. I also have my entire family to thank, especially my mother and my father, as they have given me their life-long support and unwavering faith. I thank God as well; I thank God that this part is OVER!

Table of Contents

Page

ACKNOWLEDGEMENT II

TABLE OF CONTENTS V

LIST OF TABLES XV

LIST OF FIGURES XVIII

ABSTRACT XXI

CHAPTER 1 INTRODUCTION 1

1.1 Research Objectives 3

1.2 Research Questions 6

1.3 Research Motivations: Transitions in the Professional Practice of Software Development 8

1.4 Research Motivations: Small-Team Software Development 11

1.5 Research Motivations: Reflective and Agile Practice 13

1.6 Research Motivations: Information Systems and Software Development 17

1.7 Significance of the Research Topic 19

1.8 Sections of the Dissertation 21

CHAPTER 2 LITERATURE REVIEW 23

2.1 On Method and Methodology 24

2.2 The Nature of Software Development Methodologies 28

2.3 The Professional Practice of Software Development 32

2.3.1 Professionals, Professionalism and Professionalization 34

2.3.2 The Emergence of Software Engineering 36

2.3.3 The Social and Historical Context of Software Engineering as a Profession 40

2.3.3.1 Historical Trends in the Software Engineering Paradigm 40

2.3.3.2 Professional Crisis: Defining Software Engineering 42

2.4 Small-Team Software Development 50

2.4.1 Appropriate Software Development Methods for Small Teams and Small Shops 57

2.4.2 Human Factors in Small-Team and Small-Shop Software Development 61

2.4.3 Process Diversity in Small-Team and Small-Shop Software Development 64

2.4.4 Critical Factors and “Home Grounds” for Small-Team Software Development 65

2.5 Comprehending Agile Software Development Processes 68

2.5.1 Background on Agile Methods 68

2.5.2 Selecting an Agile Method for Small-Team Software Development 71

2.5.3 Positioning Agile Methods 73

2.5.4 Cracks and Fissures in the Old Paradigm 76

2.5.5 The Emergence of Adaptive and Iterative Methods 79

2.5.7 Strengths and Limitations of Agile Processes 84

2.5.8 Philosophical Reflections on Agile Methods 89

2.6 Reflection, Reflective Practice and the Reflective Practitioner 91
2.6.1 Antecedent Work and Thinking 92
2.6.2 Reflecting on the Reflective Practitioner..... 95
2.6.2.1 Technical Rationality: The Positivist Epistemology of Professional Practice 96
2.6.2.2 Elements of Reflective Practice..... 99
2.6.3 Acceptance and Critique of Reflective-Practice..... 101
2.6.4 Situating the Utility of Reflective Action..... 102
2.7 Synthesizing the Literature: Emerging Themes 103
2.7.1 Understanding Agility through Reflective Action..... 104
2.7.3 Considering the Artisan Frame..... 105
2.7.4 Using Reflective-Agile as a Generative Metaphor for Small Teams 107
CHAPTER 3 DESCRIPTION OF THE BASELINE REFLECTIVE-AGILE METHOD..... 110
3.1 Criteria for the Selection of Extreme Programming 111
3.2 Examining the Baseline Reflective-Agile Method..... 112
3.2.1 The Agile Element: Extreme Programming Examined..... 113
3.2.1.1 Specifying the System 115
3.2.1.2 Planning the Releases 116
3.2.1.3 Iterations 116
3.2.1.4 Development..... 117
3.2.1.5 Acceptance and Small Release Cycles 119

3.2.1.6 Reflecting on Extreme Programming 121
3.3 Introducing Elements of Reflective Practice 122
3.3.1 The Ladder of Reflection 123
3.3.2 The Learning Organization 125
3.3.3 Models and Theories for Organizational Learning and Agility 126
3.4 Summarizing the Elements of the Reflective-Agile Methodology 128
CHAPTER 4 RESEARCH METHODOLOGY 130
4.1 Investigating IS Phenomenon with Dialogical Action Research 131
4.2 The Dialogical Action Research Cycle 132
4.3 The Rigor and Relevance of Dialogical Action Research 135
4.3.1 Why Action Research is an Appropriate Research Method 136
4.3.2 The Consequences of Experience 138
4.3.3 Truth in Practical Outcomes 138
4.3.4 The Logic of Controlled Inquiry 139
4.3.5 Social Context of Action 140
4.4 Dialogical AR is both Scientific and Rigorous 141
4.5 Validating the Designed Artifact using Design Science and Action Research 144
4.5.1 Appropriating Lee's Design Science and Action Research Framework 148
4.5.2 The Benefits of using an Action Research Partnership in Design Science 150
4.6 Using the Design Science and Action Research Framework 152

4.6.1 Fitting the DSAR Framework to the Research Approach 154
4.6.2 Taking the Interpretivist Mode of Inquiry for Evaluation 157
4.6.3 The Generalizability of the Research Outputs 158
CHAPTER 5 EVIDENCE FROM THE ACTION RESEARCH PARTNERSHIP 161
5.1 Description of the Dialogical Action Research Setting 161
5.1.1 The Practitioners and the Practitioner Setting 162
5.1.2 Description of the Dialogical Action Research Team 164
5.2 Iterations in the Dialogical Action Research Process 165
5.2.1 Timeline of the Dialogical Action Research Iterations 165
5.2.1.1 Discovery and Diagnosis in the Early Period 167
5.2.1.2 Adopting Extreme Programming 168
5.2.1.3 Adapting Extreme Programming 169
5.2.1.4 Achieving Reflective Practice 170
5.2.2 Reducing the Data: Synthesizing the Dialogical Action Research Iterations 171
5.3 Diagnosis 172
5.3.1 Directions and Rigor from Coding Dialogical and Observational Data 173
5.3.1.1 Grounded Theory as a Mode of Analysis 174
5.3.1.2 Computer-Assisted Qualitative Data Analysis 177
5.3.2 Grounding the Practitioners' Historical and Social Context 184
5.3.3 A List of Concerns from the Initial Diagnosis 190

5.4 Documenting the Practitioners' Extant Methods 194

5.4.1 Characterizing the Practitioners' Desire for a Method (and Methodology) 195

5.4.2 The Practitioners' Extant Software Development Process 201

5.5 Addressing the Initial Diagnoses 209

5.5.1 Issues Related to Quality 210

5.5.1.1 Ensuring Consistent Quality 214

5.5.1.2 Justifying Quality 215

5.5.2 Issues Related to Learning 218

5.5.3 Issues Related to Productivity and Process Optimization 223

5.5.4 Issues Related to Power, Risk, Skill and Leadership 227

5.5.4.1 Power, Leadership and Running a Business 228

5.5.4.2 Skills, Work Ethic and the Individual 232

5.5.4.3 Transferring Skills and Skills Cross-Training 233

5.5.4.4 Replication: Refactoring Team Habits to Align with Leader Habits 234

5.5.5 Client Confusion: Constituencies, Stakeholders and Team Membership 242

5.6 Action Planning 248

5.7 Action Taking 254

5.7.1 Adopting Agile Practices and Extreme Programming 254

5.7.2 Adapting Agile Practices and Extreme Programming 259

5.7.5 Adopting and Adapting Reflective Practice 261

CHAPTER 6 EVALUATING AND INTERPRETING THE EVIDENCE 266

6.1 Evaluating the Outcomes and Consequences of Action..... 266

6.1.1 Evaluating Productivity in Early and Sustained Successes 267

6.1.2 Evaluating Issues Related to Quality..... 274

6.1.3 Evaluating Issues Related to Client Relationships 281

6.1.4 Evaluating Issues Related to Skills Development 285

6.1.5 Evaluating Issues Related to Learning 298

6.1.6 Signs of Success 304

6.2 Focus on Learning and Communication 306

6.3 Philosophical Grounding for Interpretation 307

6.3.1 A Phenomenological Approach to Interpretation..... 308

6.3.2 Constructing a Subjective Understanding of Meaning..... 310

6.4 The Evidence and Theoretical Reflection on Agile Methods 311

6.4.1 Learning and Communication: Support for Reflective Practice 319

6.4.2 Learning and Communication: Support for the Learning Organization..... 324

6.5 Towards a Reflective-Agile Epistemology and Paradigm for Learning..... 328

6.6 Validating the Interpretive Mode of Inquiry in Dialogical Action Research..... 333

6.6.1 Framing the Dialogical Action Research Approach..... 334

6.6.2 Considering the Potential Pitfalls of Dialogical Action Research..... 334

6.6.3 A Principled Approach to Validating Interpretive Research Outcomes..... 339

6.6.4 A Principled Approach to Validating Dialogical Action Research Outcomes 345
6.7 Reflections on the Evidence 352
CHAPTER 7 ANALYSIS OF THE DESIGNED ARTIFACT 356
7.1 A Reflective-Agile Learning Model and Method Methodology 356
7.2 Elements of the Artifact 359
7.2.1 Explicating the Model 359
7.2.2 Explicating the Method 361
7.3 Iteratively Designing the Artifact 364
7.3.1 DSAR Iteration One: Early Discovery and Diagnosis 366
7.3.2 DSAR Iteration Two: Adopting XP 367
7.3.4 DSAR Iteration Three: Adapting XP 368
7.3.5 DSAR Iteration Four: Towards Reflective Practice 370
7.4 Improvements over Time 371
7.4.1 How the Designed Artifact Remedies the Real World Problem 373
7.4.2 How the Designed Artifact Improves the Practitioners' Expertise 375
7.4.3 How the Designed Artifact Improves the Researcher's Expertise 377
7.5 How the Designed Artifact Addresses the Research Questions 389
7.5.1 How the Designed Artifact Addresses Research Question One 389
7.5.2 How the Designed Artifact Addresses Research Question Two 391
7.5.3 How the Designed Artifact Addresses Research Question Three 392

7.6 Validating the DSAR Framework as Interpretative Research 393
7.6.1 Guideline: Design as an Artifact 395
7.6.2 Guideline: Problem Relevance 397
7.6.3 Guideline: Design Evaluation 397
7.6.4 Guideline: Research Contributions 398
7.6.5 Guideline: Research Rigor 398
7.6.6 Guideline: Design as a Search Process 399
7.6.7 Guideline: Communication of Research 401
CHAPTER 8 CONCLUSION 402
8.1 Specification of Learning 404
8.2 Lessons from the Problem Solution 405
8.3 Improvements to the Practitioners' Expertise 407
8.4 Improvements to the Researcher's Expertise 409
8.4.1 Implications for the Body of Knowledge on Agile Methods 410
8.4.2 Implications for the Body of Knowledge on Action Research 411
8.4.3 Implications for the Body of Knowledge on Reflective Practice 413
8.4.4 Implications for the Body of Knowledge on Small Teams 415
8.4.5 Implications for the Body of Knowledge on Organizational and Team Learning 416
8.5 Limitations and Weaknesses 420
8.6 Directions for Future Research 424

8.7 Final Reflections 427

LITERATURE CITED 429

List of Tables

Table 1 Research Questions 8

Table 2 General Phases of the SDLC 30

Table 3 Synopsis of Selected Literature on Software Development Methodologies 32

Table 4 Synopsis of Selected Literature on the Professional Practice of Software Development 33

Table 5 Elements of a Mature Profession (Ford and Gibbs 1996) 43

Table 6 Assessing the Maturity of the Software Engineering Discipline (Pour et al. 2000) 45

Table 7 Synopsis of Selected Literature on Small-Team Software Development 51

Table 8 SD Companies by Number of Employees (Source: U.S. Census Bureau – 2005 County Business Patterns) 54

Table 9 Factors Relating Team Size and Software Development Method Use 55

Table 10 Desirable Properties of a SD Methodology for Small-Teams and Small-Shops (Fayad et al. 2000) 56

Table 11 Levels of Software Method Understanding and Use (Boehm and Turner 2004) 63

Table 12 Small Team and Large Team Methodology "Home Grounds" (Boehm and Turner 2004) 65

Table 13 Manifesto for Agile Software Development (Fowler et al. 2001) 69

Table 14 Principles behind the Agile Manifesto (Fowler et al. 2001) 70

Table 15 Comparison of Agile and Iterative Methods 71

Table 16 Results of Abrahamsson et al.'s (2003) Comparative Analysis 84

Table 17 Limitations of Agile Methods (Turk et al. 2002) 86

Table 18 Risk-based Analysis of Methodologies (Boehm and Turner 2004) 87

Table 19 Prescriptions for Method Use (Boehm and Turner 2004) 88

Table 20 Diffusion of Innovation in Traditional and Learning Systems (Schön 1973) 93

Table 21 Extreme Programming Practices (Beck 1999; Wake 2002) 113

Table 22 Rungs on the *Ladder of Reflection* 123

Table 23 A Ladder of Reflection: The Case of Pair Programming (Tomayko et al. 2004)..... 124

Table 24 The Baseline Reflective-Agile Software Development Method 129

Table 25 Supporting the Scientific Rigor of Dialogical AR (Mårtensson and Lee 2004)..... 143

Table 26 Guidelines for Design Science Research (Hevner et al. 2004)..... 145

Table 27 Lee and Baskerville's (2003) Generalizability Framework 160

Table 28 Code, Category and Occurrences from the Dialogical Evidence by Category..... 181

Table 29 Code, Category and Occurrences from the Dialogical Evidence by Occurrences 182

Table 30 List of Concerns from the Initial Diagnosis..... 192

Table 31 Addressing Issues from Diagnosis with XP 249

Table 32 Major Interventions of the Study..... 250

Table 33 Comparing Canonical AR and Reflection-in-action (Mårtensson et al. 2004; Schön 1987)..... 251

Table 34 Timeline of Action-Taking..... 254

Table 35 The Ladder of Reflection as presented to SSC..... 262

Table 36 SSC Revenue and Expenses for 2008..... 270

Table 37 Bodies of Theory for Learning and Communication..... 306

Table 38 Traditional and Emerging Perspectives of Design (Nerur et al. 2007)..... 314

Table 39 Key lessons in Reflective Systems Development (Mathiassen 1998)..... 318

Table 40 Considering Reflective Practice in Dialogical AR at SSC: Quality 321
Table 41 Considering Reflective Practice in Dialogical AR at SSC: Productivity 322
Table 42 Considering Reflective Practice in Dialogical AR at SSC: Skills Development and Transfer..... 323
Table 43 Considering Organizational Learning in Dialogical AR at SSC: Individual Learning 325
Table 44 Considering Organizational Learning in Dialogical AR at SSC: Team Learning..... 326
Table 45 Considering Organizational Learning in Dialogical AR at SSC: Strategic Partnerships 327
Table 46 Summary of Principles of Interpretive Field Research..... 340
Table 47 Principles and Criteria for Canonical AR (Davison et al. 2004) 346
Table 48 Distinguishing Features of Dialogical Action Research (Mårtensson and Lee 2004). 352
Table 49 Steps in the Reflective-Agile Learning Method 363
Table 50 Key to DSAR Activities in this Research..... 365
Table 51 DSAR Activities in Iteration One..... 366
Table 52 DSAR Activities in Iteration Two 367
Table 53 Evaluating the Outcomes of Dialogical AR 372
Table 54 Improvements to the Real World Problem 374
Table 55 How the Designed Artifact Improves the Practitioners' Expertise..... 376
Table 56 How the Designed Artifact Improves the Researcher's Expertise 384
Table 57 Action Strategies Observed in July 2008..... 386
Table 58 Action Strategies Observed in February 2009..... 387

List of Figures

Figure 1 Paradigmatic Transition from Engineering to Artisanship 16

Figure 2 A Systems View of an Information System (Lee 2008) 18

Figure 3 Elements of a Philosophy of Science - Effects on Methodology (Argyris et al. 1978; Kaboub 2001) 26

Figure 4 Evolution of a Discipline from Craft to Profession (Shaw 1990) (McConnell 2004b).. 36

Figure 5 Categories and Depth of SWEBOK Knowledge (Bourque et al. 1999)..... 47

Figure 6 Elements of a Software Development Methodology (Cockburn 2000) 58

Figure 7 Problem Size and Increases in People Costs (Cockburn 2000)..... 59

Figure 8 Communication Richness and Methodology Effectiveness (Cockburn 2000)..... 60

Figure 9 Cockburn's Methodology Selection Framework (2000) 60

Figure 10 Personality Type and Small Team Performance (Gorla and Lam 2004) 62

Figure 11 Dimensions Affecting Method Selection (Boehm and Turner 2004) 66

Figure 12 Classifying Software Development Methodologies from Predictive to Adaptive (Abrahamsson et al. 2003; McConnell 2004b)..... 73

Figure 13 The Waterfall Model of the SDLC..... 75

Figure 14 Evolutionary Map of Agile Methods (Abrahamsson et al. 2003) 81

Figure 15 Progression Towards a Craftsmanship Model of IT Practice..... 105

Figure 16 Elements of Extreme Programming (Wells 2000) 114

Figure 17 Iteration Phase of XP (Wells 2000)..... 117

Figure 18 The Development Phase of XP (Wells 2000)..... 118

Figure 19 Programming Practices within XP (Wells 2000) 119
Figure 20 The Iterative Nature of XP (Wells 2000) 121
Figure 21 Single-Loop and Double-Loop Learning (Argyris et al. 1974)..... 127
Figure 22 The Ladder of Reflection (Schön 1987)..... 128
Figure 23 Steps in the Dialogical AR Process 135
Figure 24 Action Research Cycle (Baskerville 1999) 140
Figure 25 Research Framework for Design Science (March and Smith 1995) 146
Figure 26 Design Science Research Evaluation Methods (Hevner et al. 2004) 147
Figure 27 Lee's Design Science and Action Research Framework (2007)..... 149
Figure 28 Adjusted DSAR Framework (Lee 2007)..... 151
Figure 29 Initial Model of Constructs for the Reflective-Agile Method and Methodology..... 155
Figure 30 Filling in the DSAR Matrix..... 156
Figure 31 Iterative Improvements in Expertise (Mårtensson and Lee 2004) 166
Figure 32 Timeline of the Dialogical AR Iterations 167
Figure 33 HyperRESEARCH - The Study Window..... 179
Figure 34 HyperRESEARCH - Code List Editor 180
Figure 35 HyperRESEARCH - Source Window 181
Figure 36 SSC's Extant Methodology..... 207
Figure 37 SSC's Extant Methodology and the Waterfall Model 208
Figure 38 Cycles of Dialogical Action Research (Mårtensson and Lee 2004)..... 253
Figure 39 SSC's Adoption of XP Processes..... 256
Figure 40 SSC's Adoption of the Iteration Activities of the XP Method 257
Figure 41 SSC's Adoption of the Collective Code Ownership Activities of the XP Method..... 258

Figure 42 XP Processes as Adapted by SSC 260
Figure 43 Introducing Theories of Reflection and Learning into SSC's Daily Agile Processes 264
Figure 44 Relating Change, Cost and Time in the SDLC..... 283
Figure 45 Espoused Skills Disparity at SSC..... 288
Figure 46 Estimated Future Skills Disparity at SSC..... 288
Figure 47 Ideal Skills Distribution for SSC..... 289
Figure 48 Simplifying Issues and Actions to those of Learning and Communication 307
Figure 49 Evolutionary Shifts in Design Thinking (Nerur et al. 2007)..... 316
Figure 50 Abduction: the Logical Fallacy of Affirming the Consequent (Lee and Hubona 2008) 338
Figure 51 Hevner et al.'s Information Systems Research Framework (2004) 358
Figure 52 Elements of the Reflective-Agile Learning Model 360
Figure 53 The Reflective-Agile Learning Model and Method evidenced in SSC's Adaptation of XP 362
Figure 54 DSAR Activities in Iteration Three..... 368
Figure 55 DSAR Activities in Iteration Four..... 370
Figure 56 Emergence in the Design/Test Cycle (Hevner et al. 2004) 400
Figure 57 Model for the application of Deutero-learning in RALMM 419
Figure 58 Lee's Integrated Interpretive and Positivist Framework (1991) 426

Abstract

TOWARDS A REFLECTIVE-AGILE LEARNING MODEL AND METHOD IN THE CASE OF SMALL-SHOP SOFTWARE DEVELOPMENT: EVIDENCE FROM AN ACTION RESEARCH STUDY

By Jeffry S. Babb, Jr., Ph.D.
A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Business at Virginia Commonwealth University.

Virginia Commonwealth University, 2009

Major Director: Dr. Allen S. Lee
Professor, Information Systems

The ascension and use of agile and lightweight software development methods have challenged extant software design and development paradigms; this is especially notable in the case of small-team and small-shop software development. In this dissertation, a Reflective-Agile Learning Method and Methodology (RALMM) for small-shop software development, is proposed to enhance communication and learning in the use of agile methods.

The purpose of the inquiry in this dissertation pertains to: the nature of the professional practice of small team software development; the implications of the epistemology of *Reflective Practice* has for the professional practice of small-team software development; and whether the introduction of *Reflective Practice* to an extant agile methodology improves process, productivity and professional confidence for a small development team.

This dissertation uses Dialogical Action Research (Mårtensson and Lee 2004), or Dialogical AR, a qualitative and interpretive research approach, to iteratively develop and refine the Reflective-Agile Learning Model and Method (RALMM). The proposed model and method also considers Hazzan and Tomayko's (2002, 2004, and 2005) synthesis of Schön's (1983, 1987) *Reflective Practice* and Extreme Programming (XP). RALMM is shaped by Argyris and Schön's theories of practice (1974) and *Organizational Learning* (1978, 1996) and Schön's ancillary work on generative metaphor (1979) and frames (Schön et al. 1994). The RALMM artifact was developed in a Dialogical AR Partnership using Lee's (2007) framework for synthesizing design science and action research. The development and use of RALMM facilitated theorizing on the role of *Reflective Practice* in the successful use of agile methods. To assist in interpretation and analysis, the data collected during Dialogical AR cycles are analyzed using Strauss and Corbin's (1998) Grounded Theory as a mode of analysis to guide in the coding and analysis of qualitative evidence from the research.

As a result of this research, RALMM improved the practitioners' processes and productivity. Furthermore, RALMM helped to establish, formalize and reinforce a team learning system for the continued development of the practitioners' professional repertoire. Additionally, the iterative development of RALMM provides a basis for theorizing on *Reflective Practice* as an epistemology, paradigm, metaphor and frame of reference for the professional practice of small-shop software development.

CHAPTER 1 Introduction

This dissertation examines the emergence of small team and small-shop software development in order to iteratively develop a Reflective-Agile Learning Model and Method (RALMM) as an outcome of design science research as explained by Hevner et al. (2004) and March and Smith (1995). In recent years, advances in connectivity and tools and techniques related to software development have influenced and changed the nature of software development. Today, software professionals enjoy abundant opportunities to work independently, entrepreneurially and within smaller teams; often-times, these professionals may be working alone[1]. As a result of these developments, some developers have recently questioned the prevailing and accepted paradigm for the professional practice of software development: software engineering. This research presents a Reflective-Agile Learning Model and Method (RALMM) for small-shop development in response to these structural changes in the professional environment of software development. More importantly, this dissertation asks: "what is the nature of the professional practice of small-shop development?" Furthermore, "how has the engineering paradigm, frame and metaphor for the professional practice of small team software development in the small-shop environment been challenged by the advent of agile methods?" In exploring this question, Donald Schön's (1983, 1987) epistemology of *Reflective Practice* and Argyris and Schön's theories of action for *Organizational Learning* are considered as the basis of an alternative paradigm for the professional practice of software development in

[1] These "lone ranger" programmers are often said to be engaging in "Cowboy Coding."

the use of agile software development methods. This question is explored in the iterative development of an IT artifact of design science developed in a Dialogical AR Partnership in the adoption of Extreme Programming (XP), which is an agile software development method.

The advent and popularization of the Internet has unquestionably changed the way in which software is developed. The Internet makes ideas and information readily available utilizing application-layer technologies which are ever-changing and proliferate as quickly as the Internet expands and evolves. Various Internet-oriented technologies drive idea and information sharing which expands the realm of possibilities for the professional software developer. Similarly, the tools available for efficient and rapid software development have also matured and proliferated, bringing new opportunities for productivity and creativity in small team and small-shop software development. As the Internet continually decreases information dissemination and innovation diffusion costs, small teams in the small-shop setting can leverage the availability of new tools and information to increase productivity and improve their development processes.

Thus, an imperative to improve scholarly understanding of the professional practice of small team software development in the small-shop environment is apparent. Whereas the professional practice of software development has been cast as engineering since the late 1960s, the advent of agile software development methodologies has fostered debate concerning the fundamental and epistemological nature of software development and the engineering paradigm.

This chapter will proceed as follows. First, the objectives of the research and research questions are discussed. Next, a justification for the significance of the research is presented. This is followed by a discussion of motivations for this research pertaining to the professional practice of small-team software development in the small-shop environment, agile software

development methods, *Reflective Practice*, and individual and team learning. This chapter concludes with an outline of the remaining chapters of the dissertation.

1.1 Research Objectives

This research is motivated by the ways in which the advent and use of agile methods in the small-team and small-shop setting have raised questions about the professional practice of software development. Furthermore, this researcher examines the implications that the introduction of *Reflective Practice* (Schön 1983, 1987) has for team and *Organizational Learning* in the use of agile software development methods. This inquiry raises philosophic, ontological, epistemological and paradigmatic issues related to the nature of small-team and small-shop professional practice and frames certain questions:

- What is unique about small-team software development in the small-shop environment?
- What are the qualities and attributes of agile methods which are suited to small-team software development in the small-shop environment?
- Assuming agile methods constitute a paradigmatic break from traditional software methods, what are the ontological, epistemological and methodological underpinnings of the agile paradigm?
- If the epistemology of *Positivism* (what Schön calls *Technical Rationality*) is intrinsic to software engineering, then what epistemology of practice is suited to agile methods?
- Can aspects of the epistemology of *Reflective Practice* be applied to an agile methodology in order to improve process, productivity and learning for small teams using an agile method in the small-shop environment?
- What would the application of *Reflective Practice* change about agile methods?

- What would the application of *Reflective Practice* change about prevailing concepts concerning the professional practice of small-team software development in the small-shop environment?

These open questions shape and motivate this research and provide a basis for the objectives and research questions in this dissertation. These questions will be addressed and answered in the formal research questions subsequently stated in this chapter.

This research described in this dissertation has iteratively developed a model and method for small-team software development in the small-shop environment utilizing the steps of Extreme Programming (XP) and Schön's (1987) *Ladder of Reflection* – a technique which facilitates *Reflection-in-action* and *Reflection-on-action*. Following Lee's (2007) Design Science and Action Research (DSAR) framework as a model, Dialogical Action Research (Dialogical AR) is used to iteratively develop the Reflective-Agile Learning Model and Method to provide a deeper understanding of:

- **Reflective Practice**: What are the impacts of introducing *Reflective Practice* into a small-team and small-shop software development setting change?
- **The Professional Practice of Small-Team Software Development**: What is the nature of process and productivity in small-team software development in the small-shop setting? What is the source of professional validation and guidance for small-team software development?
- **Agile Software Development Methods**: As agile methods have been demonstrated as a good fit for most small-team development projects, what improvements can be made to agile software methods that would benefit small teams?

The design science artifact, RALMM, addresses these issues through iterative design and reflection within a Dialogical AR Partnership. As agile methods most benefit small teams in the areas of change and risk management, design simplicity, retrospectives (reflection) and tacit

knowledge management (reflection), Schön's epistemology of *Reflective Practice* is well-suited to further study in these areas.

The development of the RALMM follows guidelines for design science research offered by Hevner et al. (2004). This guidance describes "…the boundaries of design science within the IS discipline via a conceptual framework for understanding information systems research and by developing a set of guidelines for conducting and evaluating good design-science research" (Hevner et al. 2004: 77). Hevner et al. also recognize, as does Lee (2007), that a design artifact may be an IT instantiation, organizational design, policies or practices. As a design for a learning system, RALMM addresses the methodological implications of the use of one agile method in a small software development shop. In this sense, the word *methodology* describes more than a collection of methods; a methodology also reflects philosophical, epistemology and social norms to which a small team, explicitly or implicitly, subscribes. Thus, the objectives for this research are:

- **The Impact of Agility** – To introduce an agile method, Extreme Programming, into a small-team and small-shop software development project where the team has not previously used XP or any other agile method. XP should provide improvements to the team's process and productivity. Thus, XP is used as a necessary baseline in order to determine what further improvements, if any, arise from the introduction of *Reflective Practice*.

- **The Impact of Reflective Practice** – To apply methods and techniques from Schön's epistemology of *Reflective Practice* to XP in order to assess improvements in the following areas:
    - Team effectiveness: improvements in process and productivity
    - Agile effectiveness: improvements in adaptation to change, simplicity of design, use of retrospective, application and utilization of tacit knowledge
    - Team Learning: determine the effects reflective practice has on team learning

- Professional Confidence: *Reflective Practice* should provide a framework for professional reinforcement and willingness to adapt and evolve methods.

- **Specify Learning** – To use the Dialogical AR and the DSAR framework to improve learning for both theory and practice in the following ways:

  - Suggest improvements for the professional practice of small-team development in the small-shop setting

  - Theorize on the success or failure of the introducing the epistemology of *Reflective Practice* to XP.

  - Theorize on the success or failure of the research framework (both Dialogical AR and the DSAR framework).

These objectives provide a basis and background for the research questions in this dissertation. Research questions provide bounds and scope for a research effort in addition to providing a basis to determine whether the dissertation has accomplished its aims.

1.2 Research Questions

The research questions should facilitate the objectives of the research and bound and shape the research effort. Specifically to the approach taken in this dissertation, *Reflective Practice* should improve our understanding of agile practices for small-team software development in the small-shop environment in the following ways:

- **Small team using no method**: The practitioners in the Dialogical AR Partnership work in a small software development shop which does not use any formal or specified method. Thus, agile methods are a suitable choice as agile methods have been proven effective for small-team and small-shop software development. The principle advantages of introducing XP would be in areas of the software team learning, development process and team productivity.

- **Small team already using an agile method**: In the case where a small software development team already uses an agile method, *Reflective Practice* may allow a small-team software development team more levity, clarity, adaptability and agility in

Taking a Reflective Practitioner approach to the professional practice of small-shop software development and their use of agile software development methodologies should also improve existing theories regarding the professional practice of small-shop software development. Lastly, the DSAR research approach promises to advance theory and practice in the area of small-shop software development. Therefore, the primary motivations driving this research are related to gaining a deeper understanding of the professional practice of small-shop software development from the Reflective Practitioner perspective and developing a learning system for the use of agile methods. With these motivations in mind, this dissertation uses empirical evidence to provide answers to the following research questions:

Table 1 Research Questions

Question	Description
Research Question One	"What is an explanation for how the professional practice of small-team software development in the small-shop environment benefits or does not benefit from the introduction of *Reflective Practice*, and what contributions can this explanation make to the body of theory on the professional practice of small-team software development?"
Research Question Two	"What is an explanation for how *Reflective Practice* improves or otherwise changes the use of agile software development methods in a small-team and small-shop software development environment, and what contributions can this explanation make to the body of theory on agile software development methods for small-team and small-shop software development?"
Research Question Three	"What is a design science artifact which provides the benefits of agile software development practices and *Reflective Practice* while satisfying accepted guidance for the development, implementation and evaluation of the design science artifact?"

Answers to these research questions provide a basis for determining the degree to which *Reflective Practice* can be infused into an extant agile method in order to both improve the method and create a clearer understanding of learning in the professional practice of small-team software development in a small-shop environment.

1.3 Research Motivations: Transitions in the Professional Practice of Software Development

Due to the "software crisis" of the 1950s and 1960s (Pour et al. 2000), software development has cast as an engineering activity where engineering serves as a frame of reference and generative metaphor (Schön, 1978) which presents a way of seeing (and not seeing) the practice of software development; as a frame of reference and metaphor, engineering shapes

perspectives within communities of practice and communities of research. Thus, to assume that software development is engineering is to assume a positivist epistemology or what Schön (1983) has termed *Technical Rationality*. Software development has not always been considered an engineering practice and the recent advent and use of agile development methodologies has again brought about epistemological debate concerning the nature of software development (Abrahamsson et al. 2003; Shaw 1990).

In essence, *Technical Rationality* represents a positivist epistemology for software development where the influence of the individual professional software developer is muted by generalized, theory-driven, and normative activities to which the practitioner must adhere. Thus, according to Schön (1983), an individual practitioner serves a purpose beyond that of a compliant automaton who applies theoretical treatments to well-understood problem sets. Instead, the professional practitioner utilizes her professional repertoire in daily practice, and significantly relies on her ability to reflect in the act of practice and, post-facto, on the act of practice in cases where novelty confounds her repertoire. *Reflection-in-action* represents a key distinction between the *Technical Rationality*, a positivist epistemology of practice, and *Reflective Practice*, which is a reflective epistemology of practice.

It would be easy to dismiss this questioning of engineering as a quibble over semantics, however, terms, definitions, frames of reference and generative metaphors all exert profound influence on the actions taken in software development. As software development strives for professional recognition, engineering, the dominant espoused reference discipline for professional software development, has tended to influence the philosophy of that profession. Thus, to adopt a reference discipline is to assume the paradigm of that discipline.

If engineering is "…creating cost-effective solutions to practical problems by applying scientific knowledge to building things in the service of mankind" (Shaw 1990), so too is software engineering. Thus the emergent profession of software development has adopted the governing epistemology of engineering and all of the implications and ramifications therein. The effect of frames and metaphors can also be understood in language and human communications. Languages also provide a way of seeing the world and not seeing the world; some languages naturally express a concept or phenomenon whereas other languages completely lack an expression for the same phenomenon. In other cases, a phenomenon has a word in one language whereas that word is missing in another. In this sense, it is possible that an engineering frame for small-team software development in the small-shop environment may not be fully informative and explanatory vis-à-vis the actual daily experience of the small-team developer. Furthermore, these ways of seeing and not seeing have implications for how a small-shop practitioner learns in his or her use of a software development methodology and whether his or her individual learning contributes to a team learning system. This is a fundamental and recurring theme in this research.

Whereas Schön (1983) suggests *Reflective Practice* as a means of explaining professional activity and learning in a manner that *Technical Rationality* cannot, using *Reflective Practice* to specify a learning system for XP enables an approach to learning which goes beyond what is possible through *Technical Rationality* alone. In this case, the selection of XP is appropriate as agile methods generally reject traditional (and positivist) software engineering methodologies. Furthermore, a challenge to engineering as the dominant metaphor provides opportunity to explore new frames for small-team software development. By their nature, agile software development methods already provide an initial direction for a new frame and metaphor for

small-team software development; *Reflective Practice* should move our understanding of this new frame and metaphor forward.

1.4 Research Motivations: Small-Team Software Development

This research focuses on small-team software development in the small-shop environment for a variety of reasons. As a direct and indirect result of the increasing dependence on and use of the Internet and the World Wide Web, many of the most significant software and technology innovations have come from individuals and small teams. Here are just a few examples:

- **Linus Torvalds (Linux)** – Wrote an operating system, Linux, based on Unix and Minix whilst a computer science student at the University of Helsinki. Linux is the cornerstone of the Free and Open Source Software movement and powers a significant portion of applications on the Web and the Internet.

- **Bill Gates and Paul Allen (Microsoft)** – Childhood prodigies who worked to create Altair BASIC and perfect MS-DOS. Their company, forged out of their early software development work, has gone on to dominate the software market for personal computing. When Bill Gates stated in 1985 that "…our goal is to get a workstation running our software onto every desk and eventually in every home," he obviously meant it and the track record is clear: Microsoft Windows has, at times, dominated over 90% of the market share for PC operating systems (Jackson 1999).

- **Larry Page and Sergey Brin (Google)** – Larry Page and Sergey Brin, under direction of Terry Winograd, developed a precursor to Google, called "BackRub," whilst doing research as Ph.D. candidates at Stanford University (Battelle 2005). Google, once a dissertation research project, has gone on to significantly signify information research and retrieval on the World Wide Web. Google commands nearly half of the search-engine market (Sullivan 2006).

- **Mark Zuckerberg (Facebook)** – Mark Zuckerberg wrote the original code for Facebook in his dorm room at Harvard in 2004[2]. Facebook has gone on to exemplify

[2] There is some controversy surrounding true authorship of Facebook – Aaron J. Greenspan and, separately, Divya Nerendra, Cameron Winklevoss and Tyle Winklevoss all have made legal claims on the Facebook idea. US District Court in Boston has continually ruled in Zuckerberg's favor.

social networking websites. (McGirt 2007)

- **Brendan Eich (Javascript)** – Brendan Eich created Javascript (now, ECMAScript) while working at Netscape Communications Corporation in 1995. This dynamic, weakly-typed and prototype-based language (with first-class functions a la Scheme and LISP) is the de facto client-side scripting and state language for Internet browsers. The connectivity and application revolution on the World Wide Web would not be what it is without this language. The entirety of Web 2.0 innovations rests on the client-side capabilities of this language (Lohr 1996).

- **Shawn Fanning (Napster)** – Shawn Fanning became synonymous with (illegal) online peer-to-peer file sharing with Napster, which he created in 1998. The online file sharing phenomenon, now with technologies like BitTorrent, has fundamentally changed the music, film and software industries and has brought software and Intellectual Property theft issues to the forefront of societies' discussion on the ethical use of the World Wide Web (Ante 2000).

This list of entrepreneurial developers illustrates the degree to which society-changing software innovations often come from small teams, small shops and individuals.

Debate regarding the importance of team size has existed as long as the professional practice of software development has existed. In his seminal tome, *The Mythical Man Month*, Frederick Brooks (1975) focuses on the travails of managing large teams and large projects, but acknowledges that a "small, sharp team of first-class people" is preferable to "hundreds of programmers" (Brooks 1995: 30). While large-scale industrial/military/financial systems[3] do require engineering approaches, the eCommerce-oriented and everything-is-connected, always-online, and web-orientation of many contemporary applications often does not; many software projects for the World Wide Web is developed by small teams in a small-shop environment (Ginige et al. 2001). Even within an extremely successful web-oriented company like Google, many projects are handled by smaller teams (Google 2008); this has also been the case at

[3] The "software crisis" of the 1950s-1970s arose as the results of failures and cost-overruns in the design, development and implementation of large-scale, industrial/military/financial systems.

Microsoft (Cusumano et al. 1997). Furthermore, as a large portion of software development in the United States occurs within the context of a small team, small shop or small company (U.S. Census Bureau – 2005 County Business Patterns), then small-team software development warrants further and continued study.

1.5 Research Motivations: Reflective and Agile Practice

In the mid-to-late 1990s, several notable and experienced software developers were using and honing new methodologies which focused on small teams and adaptive responses to risk. These lightweight methodologies coalesced and formalized as agile methodologies in early 2001 (Fowler et al. 2001). The synergy, synchronicity and serendipity involved in the emergence of agile methods and the increased prevalence of small-team software development is hardly accidental. Thus, there is an imperative that scholars of software and information systems development investigate the confluence of agile methods, small-team and small-shop development and new and emerging metaphors for professionalism in software development. Phenomena of most interest to scholars might be: the distinguishing features of small-shop software development; methods most suited to small-shop development; an epistemology of practice which best explains the new and emerging frames and metaphors for the professional practice of small-team development in the small-shop environment.

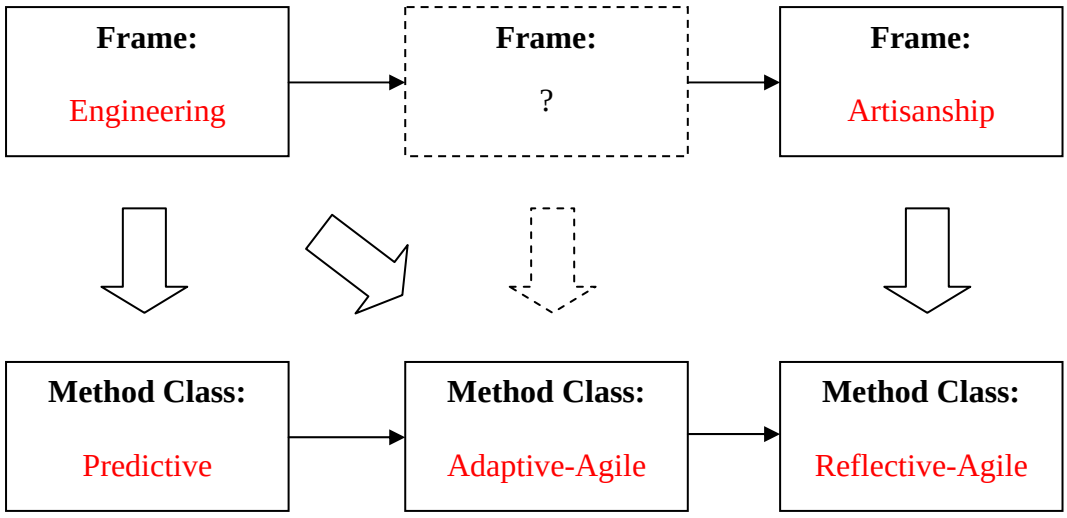
The growing agile software development community frequently speak of disconnect between their experiences in contemporary software development and the guidance of traditional software engineering. Thus, their collective response to the perceived inadequacies of traditional software engineering methods has motivated the recent diffusion of various lightweight and agile

methods (DeBaar 2007). One interpretation of the emergence of agile methods is that they represent a need for new frames and metaphors for the professional practice of software development which fit the new realities of web-oriented, small-team software development (DeBaar 2007; DeMarco et al. 2002).

Noteworthy in the nascent history of agile software development is the persistent presence of the engineering metaphor in the literature; agile methods are still cast as software engineering methods despite leanings to the contrary inherent in agile methods. There are a few "Agilists" who understand that agile methods represent paradigmatic shift away from the engineering frame of reference (Rajlich 2006), yet software professionals have merely incorporated the tenets of Agility into an existing and dominant framework of software engineering. According to Argyris and Schön (1974, 1978), this is a failure to modify fundamental governing variables and strategies for action planning in response to the consequences of action. Moreover, this constitutes a possible obstruction to individual and team learning in the use of agile methods. Thus, this research also examines the validity of the engineering metaphor in the use of agile methods in small-team software development in the small-shop setting. Given that the typical "sweet spot" for agile methods is within small-team software development projects (Boehm et al. 2004), this gives occasion to question the epistemological nature of agile methods in general. Are small-team and small-shop software developers using agile methods engineering? Are they designing? Developing? Crafting? Furthermore, as engineering is aligned with the epistemology of *Technical Rationality*, how does the use of an agile method in a small-shop imply an alternative epistemology of professional practice?

This research proposes that the success of agile software development methods in small teams and small shops provides an opportunity to examine and question the engineering metaphor and positivist epistemology as it relates to small-team software development. Thus, this dissertation also undertakes an empirical investigation to determine the suitability of *Reflective Practice* as a frame of reference and epistemology for agile method use in small software development shops. In taking the Reflective Practitioner approach, one possible frame of reference for the agile software developer in the small-shop environment would be that of artisan or craftsman. In this sense, artisans are craftspeople whose handiwork resembles pre-industrial professionalism. This is in contrast to the subsequent influence of the Industrial Revolution, the Scientific Revolution and the Age of Enlightenment which popularized scientific approaches to practice (i.e. Taylorism, etc.) and relegate an artisan as engaged in mere avocation. However, artisanship and craftsmanship may be more appropriate frames of reference for agile methods if the epistemology of reflective practice is used as a frame of reference for agile small-shop software development. This is an epistemological dilemma that Schön calls "frame conflict" (1979, 1983, and 1994). Figure 1 illustrates this frame conflict and suggests the need to theorize on a new frame for agile methods. This frame conflict has implications for the use of agile methods in small teams and small shops and also influences models of learning in the use of software development methods. Thus, Figure 1 suggests that adaptive and lightweight processes have not been sufficiently theorized such that a clear frame of reference and metaphor has emerged.

Figure 1 Paradigmatic Transition from Engineering to Artisanship



In order to understand the frame conflict inherent in agile methods, this research references a full range of Donald Schön's theoretically rich perspectives on the shaping of professional activity in his work on *Reflective Practice*, generative metaphor and framing (Schön 1979; Schön 1983; Schön 1987; Schön et al. 1996; Schön et al. 1994). For the purposes of this dissertation, a "frame" refers to a "frame of reference" which is a means by which new phenomena are understood and incorporated into an extant framework of understanding. In his work on generative metaphor, Schön (1979, 1983) describes how the metaphors which a community of practice uses to describe new and novel situations guide understanding and actions within that community. Put more simply, the language and thinking used to set a problem may determine the actions taken to solve the problem. In this sense, the metaphor of engineering has

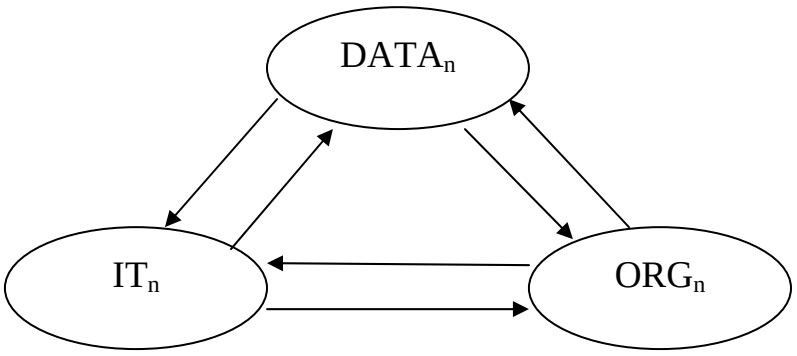
become a way of seeing, and not seeing, the professional practice of software development in general, but also in the case of agile software development in the small-shop setting specifically.

If agile methods are meant to improve process and productivity within a small software development shop, then how much more effective would the use of agile methods be if the epistemological nature of agile methods is more clearly understood? Can the epistemology of *Reflective Practice* be infused into an agile method in order to both improve the method and improve our theoretical understanding of agile methods as used in small-shop software development? Exploring and answering these questions is also central to this dissertation.

1.6 Research Motivations: Information Systems and Software Development

A review of the literatures on Information Systems Development (ISD) and Software Engineering (SE) reveals a considerable degree of overlap and potential for confusion. In order to distinguish between these two disciplines, which, in their respective literatures, seem to be speak about the same phenomenon, explicating a model of what an information system is may help. For the purposes of this research, an information system is defined as a socio-technical system which is "… the result of an information technology enabling an organization, as much as an information system is the result of an organization enabling an information technology" (Lee 2004: 13). We can expand this definition with the addition of a separate data system (Lee 2008) shown in Figure 2.

Figure 2 A Systems View of an Information System (Lee 2008)



In Figure 2, each subsystem of the information system, IT, Data and the Organization, iteratively exacts requirements on other subsystems; thus, each subsystem is structured by its interactions with other subsystems. This structuring concept of systems is well established in the literature (Barley 1986; Giddens 1984; Orlikowski et al. 1991; Schön 1967). A considerable number of articles in the seminal literature on Information Systems Development (ISD) (Hirschheim et al. 1989; Hirschheim et al. 1996; Iivari et al. 1998; Orlikowski et al. 1991; Truex et al. 2000) afford considerable focus on the organizational and social aspects of information systems development. Thus, this research characterizes Information Systems Development, and the literature thereon, as being organization and systems-centric. While there are increasing calls in the IS literature for stronger consideration of the design aspects of ISD (Agarwal et al. 2005; Hevner et al. 2004; Lee 2007; Lee 2008; Orlikowski et al. 2001), the majority of ISD literature leans towards an organizational focus. This research considers both organizational and design aspects of ISD.

The literature on software engineering shares some of the same concerns as the literature on ISD yet the software engineering literature considers the organizational system insofar as this system provides design and acceptance requirements. Hence, organizational concerns are secondary in the large majority of the Software Engineering. This is also changing; some researchers and practitioners in Software Engineering have called for more focus on values and human issues (Boehm 2003; Boehm 2006; Tomayko et al. 2004). In summary, this research assumes that software engineering traditionally takes an artifact-centric view of systems development, where the technological and data concerns of the software component of a system are of primary importance. ISD usually takes an organization-centric or systems-centric view of systems development.

For the purposes of this study, Software Engineering – the dominant paradigm for the professional practice of software development – is seen as a subset of the overall activities of Information Systems Development. Consequently, this research does not equate Software Engineering with ISD as there are important paradigmatic differences between them. Moreover, as this research focuses on small-shop software development, any resulting theoretical outcomes should be accepted within the wider body of scholarly knowledge concerning Information Systems Development and Software Engineering.

1.7 Significance of the Research Topic

A theory is also a way of seeing and not seeing; that which is not seen would not be explicit within the philosophy, ontology, epistemology and methodologies which inform that theory. While both traditional software engineering and agile software development methods

likely share a philosophical ontology, traditional software engineering methods and agile methods may not share the same epistemology with respect to professional practice. The literature on agile methods suggests that, in some cases, the prevailing paradigm insufficiently explains emerging and new realities in practice. This process is discussed at length by Kuhn (1996) as a symptomatic part of normal science. This paradigmatic uncertainty (Rajlich 2006) suggests that theorizing on an appropriate paradigmatic conception of agile methods is also a research-worthy topic.

In some respects, agile software methodologies have largely abjured the overt influence of software engineering in favor of "seeing" software development from a different perspective – a perspective which does not place scientific rigor as paramount. In fact, the Agile Manifesto (Fowler et al. 2001) does not mention science at all – it is not "seen." Moreover, the Agile Manifesto concludes with a very important caveat (Highsmith et al. 2002): "that is, there is value in items on the right, we value items on the left more" (p.121). In this case, items on the right are the tenets of software engineering and the items on the left are those of agility, adaptation and artisanship. This sentiment clearly indicates that agile methods do not entirely eschew the antecedent knowledge and wisdom of software engineering (Highsmith et al. 2002), but that there are new values which have proven more useful in daily practice. Schön (1983, 1987) addresses a similar disconnect between theory and practice in contrasting *Technical Rationality* and *Reflective Practice*.

The frame conflict between software engineering and agile methods can also be framed as a Hegelian dialectic where software engineering is the *thesis*, agile software development methods are the *antithesis* and a synthesis has not yet been reached. One possible synthesis would be to inform *Technical Rationality* from the results of on-the-spot frame experimentation

as practitioners use, refine and augment repertoire through *Reflection-in-action* and *Reflection-on-action*. Thus, a generalized body of knowledge developed though a positivist tradition can also serve as an important source and basis for a practitioner's repertoire. Just as agile methods cannot replace traditional heavyweight methods in all cases, *Reflective Practice* is an augmentation of *Technical Rationality* in the same sense that *Double-Loop Learning* is an Augmentation of *Single-Loop Learning* (Argyris et al. 1978; Argyris et al. 1996; Argyris et al. 1974).

An empirical investigation into the efficacy of *Reflective Practice* in the use of agile methods has already been suggested in the literature (Hazzan et al. 2004a; Tomayko et al. 2004). Thus, a goal for iteratively developing the designed artifact is to test the efficacy of *Reflective Practice* and suggest how *Reflective Practice* can augment the positivist epistemology of professional practice in order to provide balance. This goal is consistent with calls in the literature to balance and augment *Positivism* with other epistemologies, such as *Interpretivism* (Lee 1991; Lee 1999; Mingers 2001) or design science (March et al. 1995).

Lastly, any successful application of Schön's epistemology of practice should elevate the role of art and craft in the use of agile software development methods. As Schön (1983: 18) asserts, the art and craft of professional practice not only lies within reflection, but art and craft are indispensable aspects of professional practice often relegated by the epistemology of *Positivism*.

1.8 Sections of the Dissertation

The remaining sections of this dissertation are now outlined. Chapter Two presents a literature review which discusses and outlines the literatures which scope and frame the theoretical and practical bases for the dissertation. Chapter Three outlines and illustrates Extreme Programming as a baseline which is extended by designed artifact. Chapter Four outlines the research methods used: Dialogical AR and Lee's (2007) DSAR Framework. Chapter Four also positions this research as a mixed-methods approach which combines design research conducted in a Qualitative and Interpretive mode of inquiry. Chapter Five presents the particulars of the Dialogical AR Partnership with respect to the Diagnosis, Action Planning and Action Taking phases of Canonical action research (Canonical AR) as realized within the Dialogical AR method. Chapter Six presents an evaluation and analysis of the consequences of actions and interventions taken in the Dialogical AR Partnership and uses accepted evaluation criteria for Interpretive fields studies (Klein et al. 1999) and Action research (Davison et al. 2004). Chapter Seven evaluates and theorizes on the designed artifact utilizing accepted design science evaluation criteria (Hevner et al. 2004). Chapter Seven also uses the designed artifact to address the research questions. Chapter Eight provides concluding remarks, limitations of the study and directions for future work.

CHAPTER 2 Literature Review

This chapter provides a review of the literature to support the legitimacy of the research questions and objectives and provide background concerning the nature of these questions and objectives. With this stated, the parts of the literatures explored are:

- **Method and methodology**: A review which dissects and differentiates these two commonly-confused concepts and a definition for each of these concepts for the purposes of this dissertation.

- **Software development methodologies**: This section explores the following questions: What is the history behind software development methodologies and what are the philosophical, ontological and epistemological perspectives governing these methodologies? What is the history of software development that has driven these methodologies? Why was there a need for methodologies in the first place?

- **The professional practice of software development**: Explores the nature of profession and professionalism in the practice of software development. This includes an examination of the professions in general and of the prevalent ontological, epistemological and paradigmatic views on the professional practice of software development.

- **Small-team/Small-shop software development**: Distinguishes small-team development from other kinds of development. Explores the nature of this emergent segment of the professional practice of software development and the nature of software development methods which are most suited to small-team software development.

- **Agile software development methodologies**: This section of the literature review ties together previous sections on small-team software development and software development methods in order to present the history and case of agile software development methods.

- **Reflective practice and learning**: This section explores the nature of and reasoning behind Schön's (1983, 1987) epistemology of *Reflective Practice*, including application areas where *Reflective Practice* has been well-accepted. Additionally, *Reflective Practice* is linked to a wider program of research and theory shared by Argyris and Schön (1974, 1978, and 1996) concerning learning, reflection and change.

The remaining sections of this chapter provide background and support for the research questions in this dissertation. This chapter will proceed as follows; first, a discussion on the nature of method and methodology from a research and practical perspective; next, the nature of software development methodologies is examined; this is followed by a discussion on the professional practice of software development; next is a focus on the nature of small-team and small-shop software development; next is a description of agile methods and their suitability for small-team software development in the small-shop setting; the chapter then concludes with discussion on *Reflective Practice* and Argyris and Schön's larger program of research on learning, reflection and change.

2.1 On Method and Methodology

It is important to distinguish method from methodology as numerous references are made to both concepts throughout this dissertation. A method is subject to many meanings and interpretations; generally, most formal definitions agree that a method is a procedure or a set of steps. The Oxford English Dictionary (2001) states that a method is *"... a way of doing anything, especially according to a defined and regular plan; a mode of procedure in any activity..."* This is in keeping with the generally accepted notion of a scientific method in which a series of repeatable steps are undertaken to acquire knowledge, develop theory and/or test theory.

When considering the word *methodology*, it is important to consider this word's components. The suffix *-ology* is defined by the Oxford English Dictionary (2004) as "...*an academic discipline or field of knowledge."* Further to this is the Oxford English Dictionary

(2004) definition for a methodology: *"The branch of knowledge that deals with method generally or with the methods of a particular discipline or field of study."* Or, further: *"A method or body of methods used in a particular field of study or activity."* Therefore we can derive that a methodology is concerned with a collection of individual methods and also concerned with the generalized knowledge and philosophy surrounding these methods. Others propose that a methodology is the theory or principles guiding the application of logical rules and syllogism as relates to a field of study (Gower 1996). The Merriam-Webster Dictionary is generally in agreement with the Oxford English Dictionary in defining a methodology as "*a body of methods, rules and postulates employed by a discipline"* and *"the analysis of the principles or procedures of inquiry in a particular field."* Therefore, we can see that methodologies encapsulate methods.

This dissertation refers to methodology in two senses: a research methodology, which is guided by a philosophy of science; and a design methodology, which is usually just a collection of related methods, but is actually quite similar to a research methodology in that a design methodology is also guided by a philosophy of science and practice. In either case, relating methods and methodologies to each other develops a picture whereby a method is considered a specific procedure, while methodology refers to methods at a higher level of analysis. Generally, a methodology is informed by theories, concepts and ideas which then influence the use methods. A particular field of study has underlying rationales, frames, metaphors and philosophical assumptions all of which combine to reveal an ontological and epistemological view of phenomena and the systematic study thereon. Therefore, a given method, as a process, is only part of a methodology.

Lee (2004) suggests that a methodology lies in a continuum of subject matter pertinent to a philosophy of science:

Philosophy of science = [ontology, epistemology, methodology and method]

Another means of considering these relationships between the elements of a philosophy of science is to think of them as being interdependent. Accordingly, a method presupposes a methodology; a methodology presupposes epistemology, an epistemology presupposes ontology, etc. This implied sequence may not be as simple as previously stated; there is likely interplay between these philosophic elements which has an impact on methodology and method. Figure 3 demonstrates one possible set of relationships:

Figure 3 Elements of a Philosophy of Science - Effects on Methodology (Argyris et al. 1978; Kaboub 2001)

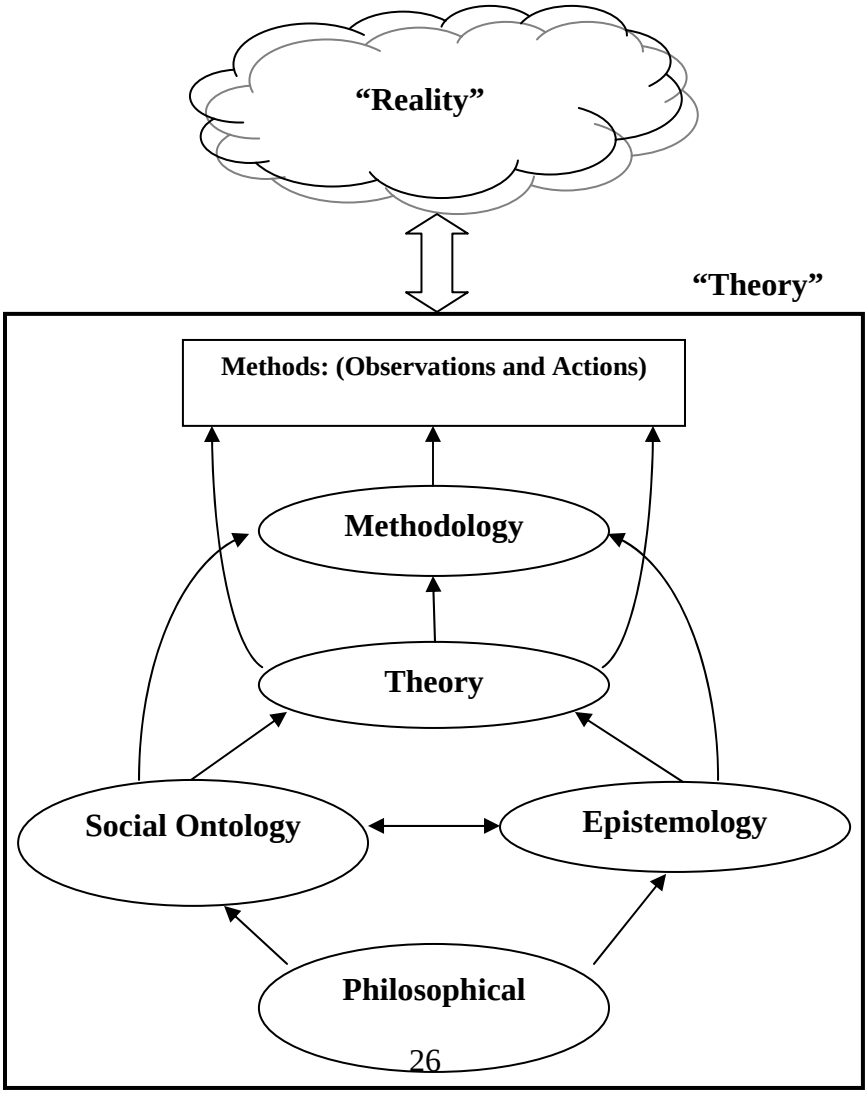


Figure 3 suggests that theory is shaped by a number of subjects and concepts. First, there is a philosophical ontology, which is comprised of a philosophical and systematic description of reality and asks the question: "what exists?" The philosophical ontology influences an epistemology, which is a theoretical description and set of beliefs on knowledge and truth. The philosophical ontology also influences a social ontology, which is a theoretical description of the nature and constitution of social reality[4]. Taken together, the philosophical ontology, the social ontology and the epistemology provide a means of comprehending the objects of study within a discipline and any theorizing on these objects. Therefore, theory encourages the systematic use of methods for empirical testing and/or action-taking (Kaboub 2001; Searle 2006).

A methodology can also be understood as a conceptual framework for understanding methods and method use. In this sense, software development methods are considered to belong to some larger conceptual framework (Cockburn 2000). If methodology and conceptual frameworks linked, then it is important to establish that a framework or methodology is an intermediary for theories which shape inquiry and action. In this sense, a conceptual framework such as "agility" in agile software development maps the territory of professional software development practice and gives coherence to agile methods.

For the purposes of this dissertation, method is defined as "*a means, algorithm or procedure for directing purposeful action*" and methodology is defined as "*a body of methods, rules and postulates employed by a discipline in accordance with a paradigm, epistemology and*

[4] The social ontology can be likened what Kuhn calls a paradigm: A concern with ontology and epistemology which is socially shaped. This is a kind of observer subjectivity to ontological and epistemic objectivity - Searle, J.R. "Social ontology: Some basic principles," *Anthropological Theory* (6:1) 2006, pp 12-29..

ontology guiding these methods." In this sense, a methodology is an operationalization of theory, epistemology and ontology and where action planning and strategizing takes place. A method is the action-taking component of action-planning guided by methodology. Therefore, a methodology shows "how things should be, or ought to be, done" and a method takes the steps to achieve the desired end-state.

It would be easy to dismiss the importance of the term *methodology* as it implies all that philosophically informs a method. We could just separate method and philosophy and avoid the apparent redundancy of the word *methodology*. However, *methodology* is a useful concept as it is paradigm-centric, whereas a method's concept is method-centric. Thus, a methodology represents a buffer between theoria and praxis where the concerns of both meet and ideas and actions exchange. For example, *Positivism* serves as a methodology when conducting scientifically-controlled laboratory experiments and *Interpretivism* is a methodology when conducting an ethnographic case study.

2.2 The Nature of Software Development Methodologies

This section discusses software development methodologies as guides and models for the professional practice of software development. In order to understand the role of *Reflective Practice* in small-team and small-shop software development, a general consideration of software development methodologies is discussed. A software development methodology arises from two factors: a software development model (such as waterfall, spiral, RAD, Agile, etc.) and software development techniques or methods (Cleanroom, PSP, Extreme Programming, Prototyping, etc.). We can consider that a "mix-and-match" approach to software development

methodology is possible such that one or more models and one or more techniques may be employed in any given project. Furthermore, it is not essential to utilize all steps within a method (Sorensen 1995). Hence, there is a degree of flexibility possible when using software development methodologies.

A software development methodology should facilitate most of the steps in the Software Development Life Cycle (SDLC) where that methodology embodies the constructs, models and methods for a software development process (Boehm 1996; March et al. 1995). Therefore, we can define a software development methodology via a simple formula (Sorensen 1995):

METHODOLOGY = MODEL + TECHNIQUES

Generally, the methodological activities undertaken by software professionals correspond to one or more stages in the Software Development Life Cycle (SDLC) (Bentley 1990; Brugha 2001). The SDLC describes the tasks and activities associated with designing, developing, delivering and maintaining software. Table 2 below describes the principle steps evident within most SDLC models.

Table 2 General Phases of the SDLC

SDLC Phase	Phase Description
Domain Analysis	The domain pertains to the area of endeavor in which the client organization and individuals are situated (Hjørland et al. 1995). In this phase, general domain models and architectures are established (Frakes et al. 1995).
Requirements, Specification and Scope Analysis	Identify the need for software product features, attributes, capabilities, characteristics or qualities. This phase demonstrates the business value for various features and ensures stakeholder vesting (Boehm et al. 1988).
Architecture and Design	Description and specification of the structures, components and interfaces to the software and/or system. These are high-level decisions made at an early stage which also consider stakeholder involvement. Strategies for reuse, modularity, component design and patterns are established (Bass et al. 2003).
Coding	Computer programming code is written, debugged and tested for logical consistency with design instructions. Application domain, algorithms and logic are manifest within the code pursuant to the specifications of the system/software architecture.
Testing	Software components, modules and systems are empirically tested for quality assurance against the original architectural specifications (Gelperin et al. 1988).
Documentation	Requirements and design documentation is finalized. The intent of this phase is to collate disparate architectural and technical documentation to prepare for implementation and acceptance.
Implementation and Acceptance	The client adopts and integrates the new software/system. User acceptance predicates on training and prevailing views on how software should work. Discrepancies between design steps and acceptance are not uncommon but should be minimized.
Maintenance	Ongoing activities to ensure software compatibility and quality after delivery, implementation and acceptance.

Some software development methods approach the SDLC as a prescriptive set of linear and sequential processes which lead to successful software; this is more evident with *heavyweight* and process-oriented engineering models of the SDLC, such as the waterfall model. The process-heavy waterfall model is often necessary for high-risk and/or large-scale development where the discipline and rigor of engineering are required.

We can also add the various aspects of theory – paradigm, ontology and epistemology – to this formulaic definition of a software development methodology in order to develop a fuller understanding of a given software development process. Therefore, we can now derive a new formula:

SOFTWARE DEVELOPMENT PROCESS = METHODOLOGY +

THOERTICAL_ASSUMPTIONS (Ontology, Epistemology and Paradigm)

Rather than exhaustively discuss every software development methodology historically used in the professional practice of software development, this section has considered how these methodologies are processes. This focus on process is congruent to the goal of this research to determine what effects, if any, the introduction of *Reflective Practice* has on the software development process in small software development shop. The literature on software development methodologies is summarized in Table 3; this table shows how these sources contribute to the objectives of this research and shows, by omission or incompleteness, opportunities for further contribution.

Table 3 Synopsis of Selected Literature on Software Development Methodologies

Source	Contributions	Omissions and/or Opportunities
(Sorensen 1995)	Describes predictive and adaptive methodologies for methodology selection. Provides a model for software methodologies.	Does not recognize the paradigmatic aspects of methodology selection.
(Hirschheim et al. 1989)	Provides a paradigmatic framework for methodology classification using the Burrell and Morgan (1979) framework.	Does not distinguish ISD and SE, does not address the professional implications of method selection.
(Iivari et al. 1998)	Provides a deeper analysis which contrasts the Hirschheim and Klein analysis. Introduces specific examples of methodologies from each of the Burrell and Morgan traditions	The mentioned Professional Work Practice approach to ISD is similar to Schön's reflective practice but does not PWP directly to Schön's epistemology. Schön's epistemology is widely accepted in many professions.
(Blum 1994)	Provides taxonomy of software development methodologies according to the formality/conceptuality of the method and the problem or product orientation of the method.	Does not provide a paradigmatic discussion on what informs a conceptual and problem-oriented methodology.

2.3 The Professional Practice of Software Development

In order to examine the professional practice of software development following terms and concepts require further definition: professional, professionalism and professionalization. The remainder of this section sets out to define these terms. The reviewed literature on professionals, professionalism and professionalization is summarized in Table 4; this table shows how these sources contribute to the objectives of this research and shows, by omission or incompleteness, opportunities for further contribution. The remainder of this section reviews selected literature on the professions, professionalism and professionalization.

Table 4 Synopsis of Selected Literature on the Professional Practice of Software Development

Source	Contributions	Omissions and/or Opportunities
(Angus 2001; Baugh et al. 1994; Boehm 2002b; Brien 1998; Cheetham et al. 1996; Dingwall 2004; Gotterbarn et al. 1999; McCalla 2002; Parker 1968; Raymond et al. 1990)	Underscores the importance of ethics and trust in professional practice. Emphasizes societal vulnerability and the need for professional competence.	An opportunity to emphasize the degree to which reflection facilitates ethical and trustworthy behavior and professional confidence.
(Larson 1979; Pour et al. 2000; Ritzer 1975; Ritzer et al. 1988; Wasserman 1996)	Characterizes the process of professionalization and the potential pitfalls thereon.	The professional practice of software development is still emerging and paradigmatic shifts in methodologies constitute processes of professionalization.
(Bagert 1999; Ensmenger 2001; McConnell 2004b; Shaw 1990; Speed 1999)	Discussion on the professionalization of software development: the controversies, requirements and parameters, and the status of achieving a profession.	The professionalization of software development has been extensively cast as engineering; agile methods have challenged this characterization.
(Shaw 1990)	Software development as a profession of design – justification for the engineering frame of reference.	Does not address the role of art in the designing process.
(Bourque et al. 1999; Ford 1996; Holmes 2000; Orden 1967; Parnas 2002)	Establishes the positivist world-view of software engineering and a body of knowledge based on this world-view.	Alternative world-views, such as reflective practice, provide a more complete understanding of agile methods.
(Davison 2000; Orlikowski et al. 1988; Oz 1992)	Information Systems Development perspectives on professionalization.	Software Engineering and Information Systems Development are not sufficiently distinguished.
(Boehm 2002b; Denning 2001a; Denning et al. 2001; Elliot et al. 2002; Purgathofer 2006; Trauth 1982)	Professional Identity	This needs to be clarified for small-team development; the means by which professional confidence can be attained and sustained in a small-team environment is required.

2.3.1 Professionals, Professionalism and Professionalization

Our modern world has been extensively forged by the professions and professional activity. The professions and professional activity shape most of the principal business of societies: defense, education, medicine, law, management, design and social work are just a few examples (Schön 1983). The primacy of the professions lies within the large body of expert knowledge held within professional practitioners; this knowledge is honed through years of intensive study and is founded in theory and best practices. Furthermore, professions are self-regulating and ensure that ethical bounds are placed upon practicing professionals (Brien 1998). Professionals serve as the "tools" of technological and social progress such that society at-large enjoys the benefit of the complex and important knowledge of the professions. The power wielded by professionals also creates vulnerability within the rest of society as it is subject to the failures and successes of professional practice (Brien 1998; Cheetham et al. 1996; Dingwall 2004; Ritzer 1975). The power embedded within the professions presents ethical issues and brings these issues to the forefront of self-regulating activities within the professions (Brien 1998).

Within professional ethics and self-regulation we can find the impetus for *professionalism*: an understanding of the rights and obligations which a profession bestows upon a sanctioned practitioner as he or she exercises the profession. This sanction is important as the concept of professions is often loosely employed. Thus, in the strict and classical sense, professionals have a fiduciary duty to ensure that their decisions and actions in practice serve the welfare of their clients (Jonsen et al. 1998). Therefore, *professionalism* has a great deal to do with obligations, responsibilities and values. A profession must have infrastructure and

apparatus with which these values and mores are institutionalized; many professions have some governing body which functions at an international, national or local level for these purposes. These professional bodies serve not only to protect the interests of the public and the profession, they often serve as legal guides where expertise in professional subject matter is critical. Additionally, many professions license practitioners in their field to ensure quality and ethical behavior. Fields of medicine, law, education, accounting, engineering, and architecture and, recently, project management, are all regulated through licensing (Ritzer et al. 1988).

It is generally accepted that a profession emerges from the *professionalization* of a trade or occupation. The body of knowledge, professional qualifications and formalization of ethics promoted by professional associations are the hallmarks of the professions; ascension to these endeavors is part and parcel of the processes of *professionalization*. The professionalization process can be easily understood in the example of engineering. As an engineering discipline matures, it traverses several stages from ad hoc activity to professional activity. Generally, the introduction of production efficiencies through management and technology, followed by the development of a supporting science, transforms craftsmanship to professional activity (Shaw 1990).

Without a unifying set of professional practices, the craftsman's effective activity is ad hoc and does not necessarily contribute to a wider body of knowledge for effective practice. When it becomes necessary that the ad hoc activities of the craftsman should support wide-scale production, craft is commercialized as management and production techniques encourage greater and more efficient output. A profession develops when a systematic science emerges which studies and formalizes management, technology, economies and efficiencies within the craft (McConnell 2004b; Shaw 1990). This formalization transforms craft to professional practice and

formalizes the know-how of craftsmanship into structures which supersede the efforts of the individual.

Figure 4 Evolution of a Discipline from Craft to Profession (Shaw 1990) (McConnell 2004b)

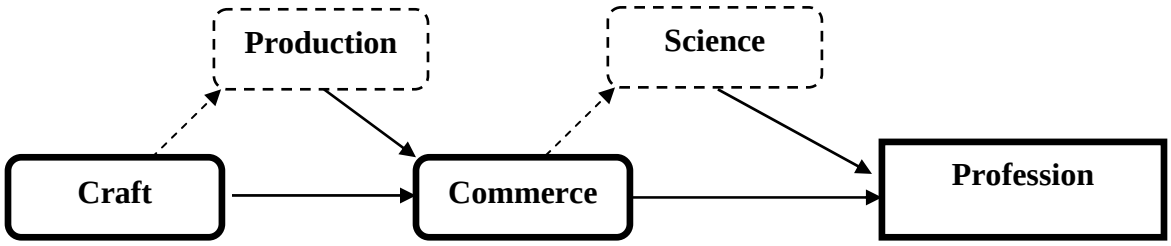


Figure 4 describes the general progression from craft to profession where the outcomes of craft are improved by production techniques to realize commerce. With commerce in place, a formal science concerned with the problems related to craft and production within commercial endeavor can develop. A discipline emerges when science and practice create a body of knowledge which formalizes the required knowledge and skills for professional practice within a field; thus, professionalism within this discipline is known and explicated.

2.3.2 The Emergence of Software Engineering

In the professionalization of software development, the sciences which arose to support it are widely accepted as the science of design and computer science. Simon (1996) indicates that sciences of design are sciences of the artificial: man-made artifacts not otherwise found in the natural world. Among the principle areas of knowledge and professional activity governing design is engineering. Engineering is concerned with designing artifacts having specific properties and the science and knowledge of processes required to design these artifacts

(McConnell 2004b; Simon 1996). Design is not necessarily the sole purview of engineering; design is a measured effort to determine and implement courses of action which change existing situations into preferred situations (Simon 1996: 111). Furthermore, it can be argued that design and designing are central to most professional endeavors; design is concerned with what ought to happen rather than what is happening. Designs are concerned with desired end states regardless of the found state. Many traditional, long-standing and well-known professions are concerned with design: education, engineering, architecture, law and medicine. Therefore, the design discipline which has most closely been associated with software development is engineering; hence, the profession is largely recognized and accepted as software engineering.

Software engineering arose in response to a "software crisis" characterized by a general dissatisfaction with the results of large software projects in the 1950s, 1960s and early 1970s. Early software applications were cobbled together through creative bouts of trail-and-error as the software development was still very new and novel. The tools, concepts and technologies that contemporary software developers enjoy today did not exist for early software developers beyond theory predicting their possibility. In response to the software crisis, the rigor and discipline of science and engineering was called upon to create an ordered approach to software development which would optimize output and productivity and minimize waste.

It is generally accepted that software engineering emerged in 1968 and 1969 at the North Atlantic Treaty Organization (NATO) workshop on the state of software development (Shaw 1990). The environment at the time was receptive to the professionalization of software development as the first few decades of computer programming were fraught with failures and cost overruns (Boehm et al. 1988; Brooks 1995; Orden 1967). The acceptance of software

development as a discipline of design and engineering brought an opportunity to formalize software tools, technologies, management, processes and design activities (Shaw 1990).

As a discipline of design, software engineering draws from a body of knowledge and patterns for designing which encourage design reuse to solve familiar problems. These problems become familiar as their solutions are added to body of professional knowledge within the discipline; which is a benefit of a stable science. The degree to which problems and solutions can be classified and routinized signals the maturity of a profession (McConnell 2004b; Raymond et al. 1990). Thus, the *engineering* in software engineering would suggest that the professional practice of software development entails "…creating-cost effective solutions to practical problems by applying scientific knowledge to build things in the service of mankind" (Shaw 1990). These cost-effective solutions represent a set of solved problems; thus, the scientific paradigm of software engineering should provide a set of reusable artifacts which can be applied to well-known classes of problems (Kuhn 1996). Among the reusable artifacts of software engineering are (McConnell 2004b):

- **Software design architectures** - Client-Server, Structured and Procedural, Modular and Object-Oriented, Three-tier, Service oriented, etc.
- **Software Design Methodologies –** Agile, XP, Scrum, RAD, RUP, Spiral, Waterfall
- **Design Patterns -** a reusable solution to common problems in software design – Factory method, Singleton, Bridge, Adapter, Observer, etc.
- **Requirements Specification –** Use cases, IEEE 830, etc.
- **User interfaces and Human Computer Interaction –** GUIs, Web design, etc.
- **Estimation processes –** Parametric Estimating, COCOMO, Function Point Analysis, Proxy-based estimating, the planning game, etc.

- **Testing** – Unit testing, Integration testing, System testing, Acceptance testing, Beta testing, etc.

- **Revision control and software configuration management –** BitKeeper, Bugzilla, Perforce, CVS, VSS, etc.

- **Project and process management and quality control** – ISO 9001, Total Cost Management, Total Quality Management, Six Sigma, Capability Maturity Model

Subsequent to the adoption of engineering as a frame of reference and metaphor for the professional practice of software development, software development professionals have struggled to define what software engineering means. The engineering frame brings philosophical, ontological and epistemological assumptions with it; thus it presumes a paradigm or world-view. The world-view of software engineering suggests that a systematic, disciplined and quantifiable method for software construction is required to create effective and reliable software (Bourque et al. 1999; Holmes 2000; Orden 1967; Parnas 2001; Shaw 1990). According to the software engineering world-view, the knowledge concerning the tools, people and technologies required to build software are manifested within the formal methods and mathematical rigor of engineering. In the software engineering world-view, the human element in the problem space represents an irrational source of error which must be controlled. Therefore, software engineering favors predictive planning and action sequencing where the interaction between people and processes is closely monitored and managed. Software engineering, as a discipline, can be considered as an example of *Technical Rationality* (Schön 1983, 1987). The rational attitude inherent within software engineering underscores the primacy placed on generalized theoretical knowledge and can thus be described as paradigmatic in nature (Boehm 1976; Boehm 1979).

2.3.3 The Social and Historical Context of Software Engineering as a Profession

The progression towards professionalization in software has been fraught controversy and disagreement on the tenets of the discipline and the meaning of professionalism. Boehm (2006) has characterized the evolution of software engineering as a Hegelian dialectic pursuing Kuhn's (1996) "normal science": the business of defining and redefining the ontological, epistemological and paradigmatic space within which the discipline lies. This is apropos as normal science "… possesses a built-in mechanism that ensures the relaxation…" of questions surrounding the appropriateness of the paradigm (Kuhn 1996: 24). While the controversy continues, the profession is still cast in the engineering paradigm.

2.3.3.1 Historical Trends in the Software Engineering Paradigm

The Hegelian dialectic is a learning process whereby a *thesis* (an intellectual proposition) is met with a negation of that thesis, *antithesis*; this conflict is resolved by reconciling their common truths, *synthesis*, and forming a new proposition. By using a Hegelian dialectical approach, Boehm (2006) offers a framework for understanding the history of the professional practice of software development and provides an understanding of the controversies surrounding the adoption of engineering as a frame and metaphor for the professional practice of software development.

The 21st Century has brought antitheses to software engineering. Global connectivity has resulted in global thinking and global economies placing new and greater demands on software and has provided an occasion for redefining software engineering. Traditional heavyweight

processes and plan-based approaches, which emphasize extensive and contractual documentation and maturity models, are said to confound the ability for software engineering to provide relevant and timely software (Boehm et al. 2004). This dilemma has parallels in the rigor and relevance debate found in the information systems literature (Baskerville et al. 2004; Benbasat et al. 1999; Benbasat et al. 2003; Orlikowski et al. 2002; Orlikowski et al. 2001). Thus, as software engineering matures and undergoes a plethora of Hegelian dialectic episodes, balancing agility with discipline becomes a fundamental concern (Boehm et al. 2004).

Among the most compelling antithetical developments in software engineering has been the arrival of agile software development methods. Agile methods favor individuals, interactions, working software, customer collaboration and response to change especially in areas where teams are small, risk is low, personnel are highly-capable, requirements are in flux and in an organization/team which thrives on chaos vs. order (Boehm 2006; Boehm et al. 2004; Cockburn 2002; McBreen 2002a). Recently, agile software development methods have given rise to "post-agile" methods such as "value-based" software engineering, whereupon Boehm (2006) calls for a focus on adapting technology to people rather than vice versa. This concept resonates with the raison d'être for a large portion of Information Systems research and development (Benbasat et al. 2003; Hirschheim et al. 1989; Orlikowski et al. 2001; Whinston et al. 2004). However, as Boehm is a died-in-the-wool software engineer, his call for an increased focus on human issues is somewhat contrary to the Positivist paradigm of software engineering.

2.3.3.2 Professional Crisis: Defining Software Engineering

The history of software engineering illustrates an ever-present push towards the professionalization of software development. However, debate regarding the nature of the profession goes back to the earliest days of software development practice (Ensmenger 2001). As Ensmenger (2001) points out, "one of the most intriguing and influential developments in the history of software has been the widespread adoption of the rhetoric and ideology of software engineering" (p. 70). This engineering frame of reference and metaphor has pervasively directed an agenda influencing most technological, managerial, and professional developments in the field for the past 30 plus years. Whereas in the 1950s and 1960s competing ideas of professional software developers as "certified public programmers" (after accounting) or as computer scientists were considered: the community chose engineering (Ensmenger 2001). In adopting engineering as the frame of reference and metaphor to guide the professional practice of software engineering, a set of ontological, epistemological and paradigmatic assumptions were also assumed – willfully or otherwise.

One aspect of the controversy surrounding software engineering concerns software engineering's body of knowledge. McConnell and Tripp (1999) claim that, despite the advances of the 1970s, 1980s and 1990s, most software development follows a "code and fix" approach: hacking away at the problem in an unstructured manner – à la "cowboy coding." (Highsmith et al. 2001) The elements of a mature profession, however, are far from ad-hoc in nature (Ford 1996; McConnell et al. 1999; Pour et al. 2000). In a comprehensive study covering the aspects and attributes of mature professions such as medicine, law, engineering and accounting, Ford and Gibbs (1996) have described the elements of a mature profession (Table 5).

Table 5 Elements of a Mature Profession (Ford and Gibbs 1996)

Element	Implementation in Software Engineering
Initial Professional Education	Education is usually obtained in a computer science program, yet it has been argued that computer science is not software engineering (Parnas 1999). Roughly 40% of professional software engineers are computer science trained (Blevis et al. 2006; Boehm 2006).
Accreditation	The Federation of accrediting bodies in areas of engineering education has greatly progressed with organizations like ABET, Inc. (née Accreditation Board for Engineering and Technology).
Skills development	Initial education is a first step whereas initial experiential training is more valuable in developing a competent professional. IEEE and ACM Software Engineering Body of Knowledge has provided a basis for explicating these skills (Bourque et al. 1999).
Certification	Vendor certifications are very popular but have a questionable shelf-life. While doctors and lawyers take board exams and accountants pass CPA exams, software engineers have few options which are widely seen as legitimate (McConnell et al. 1999).
Licensing	In other professions, licensing is similar to certification with the exception that licensing is mandatory and administered by government authority. Licensing has had ups and downs in software engineering, with most efforts having fizzled out (Bagert 1999; El-Kadi 1999; Knight et al. 2002; Parnas 2002).
Professional Development	This is a shared activity between the various stakeholders within the profession. The practitioner has a personal obligation to herself and the profession to engage in ongoing professional development. Furthermore, organizations and employers need to accommodate this professional development need (Bourque et al. 2002).
Professional Societies	There are special interest teams within the IEEE and ACM for software engineering.
Code of ethics	The IEEE and ACM have made significant progress in this area and have provided the profession with a code of ethics for software engineering (Gotterbarn et al. 1999). Similar calls have been made for a code of ethics for Information Systems and Information Technology professions (Davison 2000; Orlikowski et al. 1988; Parker 1968), however, these professions are less matured than software engineering and in a less developed state with respect to form and function (Hirschheim et al. 2003).

Creating legitimacy in the eyes of academia and society is primary among the motivations for establishing a professional tradition for software development (Ensmenger 2001). With respect to legitimacy in society, the elements of a mature profession serve as a barometer for progress in gaining mindshare and legitimacy. In a study conducted by the ACM and IEEE Computer Science Software Engineering Coordinating Committee (SECC), software engineering's maturity in areas of an initial professional education, code of ethics, accreditation, skills, development, professional development, certification and licensing was found to be either non-existent of ad hoc (Pour et al. 2000: 36). While some of these areas have likely since improved, this may suggest that the profession of software engineering has yet to establish a unique identity. The roots of this identity lie in the academic delivery of the software engineering body of knowledge in programs dedicated to software engineering. Table 6 shows the progress of the professionalization of software development.

Table 6 Assessing the Maturity of the Software Engineering Discipline (Pour et al. 2000)

Infrastructure component of a mature profession	How software engineering measures up
Recognized body of knowledge	IEEE-CS/ACM taskforce has ratified the SWEBOK (level 2-3)
Professional societies	IEEE, ACM, SIGSOFT (level 2-3)
Code of ethics	Both IEEE and ACM have codes of ethics specific to SE (level 2-3) (Gotterbarn et al. 1999)
Initial professional education system	Programs in SE are on the rise, yet not widespread (level 1-2)
Accreditation of professional education programs	ABET and CSAB (level 2-3)
Skills development mechanism for professionals entering the practice	Certificate programs, MSE degrees (level 1)
Professional development programs to maintain skills and knowledge currency	Ad hoc and not consistent across the profession (level 1)
Certification of professionals administered by the profession	Limited, inconsistent or vendor-based (i.e. Microsoft MSCE) (level 0-1)
Licensing of professionals administered by government authority	Recently in Texas, Canada and the UK

With respect to formalized training, presently, professional software developers are: self taught, trained in computer-science, trained in mathematics, trained in engineering, trained in information systems (and related fields), and/or trained in some other discipline orthogonal to the needs of software engineering (Bourque et al. 2002; Pour et al. 2000). Each of these disciplines has a world-view and history which may not be in strict harmony with those of software engineering. The Software Engineering Body of Knowledge (SWEBOK), in its present accepted and ratified form (IEEE, ACM 2004) was incrementally developed via three iterations: Straw Man (ISO/IEC 12207), Stone Man and Iron Man. A final version was accepted by the Industrial Advisory Board and the IEEE Computer Society Board of Governors in February,

2004 (Society 2004). The SWEBOK provides for the following knowledge areas relevant to software engineering:

- Requirements
- Design
- Constructing
- Testing
- Maintenance
- Configuration Management
- Engineering Management
- Engineering Process
- Engineering Tools and Methods
- Quality

It is quickly apparent that these knowledge areas roughly follow the pattern of the SDLC. In explicating these knowledge areas, and perhaps in recognition of the paucity of dedicated software engineering programs, the SWEBOK also specifies the following related disciplines:

- Computer engineering
- Computer science
- Management
- Mathematics
- Project management
- Quality management
- Software economics
- Systems engineering

It is worth noting that existing university programs can provide tutelage in these areas from well-established disciplines. For instance, the information systems, management or accounting program in a school of business could partner with a computer science and/or computer engineering program in a school of engineering to provide a foundation in the majority of these knowledge areas. Figure 5 depicts how depth of knowledge in each knowledge area is handled.

Figure 5 Categories and Depth of SWEBOK Knowledge (Bourque et al. 1999)

Specialized – Practices used only for certain types of software	**Generally accepted** – Established traditional practices recommended by many organizations
	Advanced and Research – Innovative practices tested and used only by some organizations and concepts still being developed and tested in research organizations

With the existence of SWEBOK and with the beginnings of a mature profession in place or under way, continued debate concerning the professional practice of software development presents a curiosity. What are the major issues and sticking points? Among the issues are: the lack of a formalized program in higher education specifically for software engineering; debate on the need for certification and licensing; debate concerning the appropriateness of engineering to describe the professional activities of software developers; debate on the degree to which ethical standards for professional conduct are educated, upheld and enforced.

With the SWEBOK in place and general support from IEEE and ACM, the number of programs in Software Engineering will continue to increase. Whereas some feel that independence from software engineering's current *de facto* home in computer science is a necessary step (Denning 1998), while others hold that computer science continues to provide the necessary scientific and mathematic rigor that a mature discipline of software engineering requires (Denning 2005; El-Kadi 1999). It is clear from reading the SWEBOK that a great deal of the subject matter is carried by programs in computer science, engineering and business. However, there is a concern that software engineering needs its independence if maturity is to be

attained. A few principle arguments stand out: (1) software engineering demands an application area for the knowledge computer science teaches, whereas computer science does not; (2) There are epistemological issues relating to the Positivist influence on computer science and the design influence on software engineering; (3) There is ongoing debate as to the balance between art and science in the professional practice of software development – designing requires both (Pour et al. 2000).

The certification and licensure question is understood in terms of the level of formalization desired and extant in the discipline: how much is art and how much is science? Regarding the ongoing development and dissemination of SWEBOK, some favor the rigor of formal education, training, certification and testing (Bagert 1999; Jonsen et al. 1998; Parnas 2001; Pour et al. 2000; Ritzer 1975) in line with other professions, while others call for consideration of the human aspects of software engineering (Boehm 2002b; Denning et al. 2001; Tomayko et al. 2004) and still others doubt the maturity of the discipline has progressed enough to successfully conduct licensing and certification (Knight et al. 2002; Schaefer 2006). Further to this debate is the opinion that not all software development need be considered engineering (Denning et al. 2009); at least this is the position of the State of Texas (Speed 1999).

The debate on the nature of professional software development, and the degree to which professional practice will continue to mature as an engineering discipline, has not yet been reached. It is clear that disagreement exists as to whether the professional practice of software development is indeed engineering; or the degree to which software engineering has formalized and matured; or what balance is there between art and science in the professional practice of software engineering. Lastly, it is clear that other related professions in computer and

information technology also struggle with these issues (Carayannis et al. 2001; Klobas et al. 1995; Miser 1987; Moore 2000; Oz 1992; Purgathofer 2006).

A final issue related to the maturing of the professional practice of software development is that of ethics. While a code of ethics is the hallmark of a matured profession, there are some who feel that computing disciplines are difficult to define; this confounds attempts to formalize the mores of the profession. With this, we return to the use of professional expert knowledge and the public good. The non-professional laity is vulnerable to lapses in ethical behavior and yet the professional practice of software development, ever increasing in its importance to society at large, struggles with a basic concept of its ethical responsibilities. Examples where damages to society are the result of careless or unethical professional behavior are numerous and it may be that many advances in engineering come as responses to disasters. When a bridge collapses (Quebec City, 1907, Tacoma Narrows, 1940, Minneapolis, 2007), society looks to engineering to learn from the disaster and protect the public from future harm. However, failures in the engineering of software and systems have also had serious impact:

- The IRS cost taxpayers $50 million per year in lost revenues in the 1990s due to software design and implementation failures (McConnell 2004b).
- An FAA Advanced Automation System overran its planned budget by $3 billion (Britcher 1998).
- The 1998 Mars Climate Orbiter mission, a $193.1 million NASA project, failed when the landing vehicle crashed into the surface on mars due to a software error where imperial and metric units were confused (Euler et al. 2001).
- The Denver International Airport, espoused as a feat of technology, was delayed in opening for a year with software-induced faults in the baggage handling system which cost up to $1.1 million per day (Glass 1998).
- The first launch of the Ariane-5 rocket for the European Space Agency exploded due to a software error (Nueibeh 1997).

Each of these failures represents great cost to society, who entrust professional software developers to be responsible, accountable and ethical. While the IEEE and the ACM have a code of ethics for software engineers (Gotterbarn et al. 1999), consistent uptake and use of this code is not certain as very few professional software engineers are systematically held accountable to this code (Gotterbarn et al. 1999). This topic of ethics in computing is not new (Davison 2000; Martin et al. 1990; Raymond et al. 1990), but it represents another puzzle piece for attaining a professional tradition for software development.

As this dissertation progresses to account for the professional practice of small-team development, it is important to realize that the professional practice of software development has not stabilized to the extent that professions in medicine, law, architecture and accounting have; software development is still nascent and environmental trends provide ample opportunity for further study. With this we can frame the matter of ethics in the professional practice of software development as a matter of trust, responsibility and accountability. Moreover, the structures which reinforce professionalism in software development may be different in the small-shop setting as opposed to what exists in a larger organization. As *Reflective Practice* presents a methodological proposition for the use of agile methods in the small-shop environment, the role which professionalism plays is in the use of these methods is of acute importance.

2.4 Small-Team Software Development

This section discusses the nature of small-team and small-shop software development and the increasing incidence and importance of software development done "in the small." The reviewed literature on small-team software development is summarized in Table 7; this table

shows how these sources contribute to the objectives of this research and shows, by omission or incompleteness, opportunities for further contribution. The remainder of this section reviews the literature on small-team software development.

Table 7 Synopsis of Selected Literature on Small-Team Software Development

Source	Contributions	Omissions and/or Opportunities
(Cockburn 2000; Constantine 2002; Cragg et al. 1993; Dybå 2000; Faraj et al. 2000; Fayad et al. 2000)	Distinguishes the conditions and requirements of small scale software development	Very little address on the implications for professionalism and professional practice at this level. Stronger link can be made for the need for adaptability.
(Boehm et al. 2004; Cockburn 2000; Cragg et al. 1993; Cusumano 2007; Kostamovaara et al. 2007)	Distinguishing small-team from small-firm	More emphasis needed on the possibility of small teams existing within a large organization.
(Cockburn 2000; Fayad et al. 2000; Nunes et al. 2000)	Team size and method selection	More emphasis needed on methodologies. More emphasis needed on the paradigmatic implications of methodology selection.
(Gorla et al. 2004)	The need to emphasize people over process. Presents the use of the Meyers-Briggs Type Indicator as a basis for personality and role matching.	More emphasis needed on paradigmatic concerns and epistemological concerns – how are people emphasized? Personality types are concerned with *nature* whereas a concern with establishing a learning system may emphasize *nurture*.
(Boehm et al. 2004; Cockburn 2002)	Professional competency and repertoire	Reflective practice can assist in elevating professional competency
(Boehm 2002a; Boehm et al. 2004; Cockburn 2002)	Critical considerations for method selection and use depending on team size	Can be used to characterize the reflective-agile software development methodology

As the Internet, the World Wide Web, and other Information and Communication Technologies have improved human commerce and information-sharing through connectivity, it has become increasingly clear that small teams are "…developing significant products that need effective, tailored software engineering practices…" (Fayad et al. 2000: 115). In fact, the awareness of and distinction between large-team and small-team software development has existed for quite some time (DeRemer et al. 1975). Whereas large-scale industrial and military software projects gave rise to software engineering (ostensibly for large-team development), it is arguable that the rise in the importance in personal computer and microcomputer in the 1980s, and of Internet-related applications and technologies in the 1990s and 2000s, have increased interest in small-team development (Cockburn 2000; Constantine 2002; Cragg et al. 1993; Eppinger et al. 1994; Faraj et al. 2000; McDonald et al. 2001a; Rajlich 2000; Reddy et al. 1991). As the number of personal computers grew to an estimated 1 billion from 1980 to 2007, and is estimated to grow to 2 billion by 2015 (Chapman 2007), there is little wonder that the demand for more computer programs and a wider variety of computer programs has markedly increased; a significant portion of these programs are not large-scale and large-team products (Fayad et al. 2000).

Even in units sold, mass-market software (such as Common Off-the-shelf Software (COTS), customizations of COTS, and software resulting from web engineering) surpasses the number of government and large industrial applications (Fayad et al. 2000; Ginige et al. 2001; Gorla et al. 2004). As software growth has produced a larger number of small companies and small teams, it would naturally follow that research on software development and development methodologies should shift to accommodate this trend; however, research in this area is not abundant prior to the 2000s (Cockburn 2000; Cusumano et al. 1997; Fayad et al. 2000). With the

onset of agile software development methods, a considerable amount of research and field reports have concentrated on agility in software development with some reports concluding the utility of agile methods is greatest for small-team development (Abrahamsson et al. 2003; Cockburn et al. 2001; Cusumano et al. 1997). As such, there is room for continued scholarly investigation into phenomenon related to small-team and small-shop development as rapid change maintains turbidity within the domain of practice.

It is important to distinguish between small teams and company size. A small team can exist in a small company (hence the term "small shop") or a small team can exist within a larger company. In both cases, the small team can be considered as mostly autonomous where the small-shop team would be the most autonomous. Furthermore, when considering application sectors (such as finance, telecom, aerospace, etc), the number of software projects utilizing five developers or less accounts for as much as 66% of the overall number of software development projects (Boehm et al. 2004: 226). While evidence from the literature varies, there is ample support to define a small team as consisting of 10 members or less; in many cases five members or less is common (Boehm et al. 2004; Cockburn 2000; Cragg et al. 1993; Dybå 2000; Ginige et al. 2001; Reifer 2000). For the purposes of this dissertation, the terms "small team" and "small shop" should be read synonymously as they pertain to a small team located in a small company of 10 or fewer practitioners and, more often, 5 or fewer practitioners.

The U.S. Census Bureau records county business patterns by a NAICS[5] code which breaks down the distribution of companies by employee size and by industry. The most applicable NAICS codes for software development are 541511 (Custom Computer Programming

[5] North American Industry Classification System

Services) and 511210 (Software Publishers). County Business Pattern Data from 2005 indicate that companies in NAICS classification 511210 (Software Publishers) with nine or fewer employees comprised 60.01% of the total of 8793 companies nationally that year. The 2005 County Business Pattern data also indicate that companies in NAICS classification 541511 (Custom Computer Programming Services) with nine or fewer employees comprise 83.56% of the total of 47,673 companies nationally that year. Table 8 shows these figures graphically and underscores the magnitudinal importance of small-team development in small companies.

Table 8 SD Companies by Number of Employees (Source: U.S. Census Bureau – 2005 County Business Patterns)

Breakdown of software development companies by number of employees and company size						
	NAICS 511210 (Software Publishers)		NAICS 541511 (Custom Computer Programming Services)		Total (511210 and 541511)	
Number of Employees	*Number of Companies*	*Percentage of Whole*	*Number of Companies*	*Percentage of Whole*	*Number of Companies*	*Percentage of Whole*
1-4	3941	44.82 %	34698	72.78 %	38639	68.43 %
5-9	1336	15.19 %	5137	10.78 %	6473	11.46 %
10-19	1211	13.77 %	3449	7.23 %	4660	8.25 %
20-49	1187	13.50 %	2648	5.55 %	3835	6.79 %
50-99	527	5.99 %	1020	2.14 %	1547	2.74 %
100-249	371	4.22 %	539	1.13 %	910	1.61 %
250-499	140	1.59 %	126	0.26 %	266	0.47 %
500-999	46	0.52 %	38	0.08 %	84	0.15 %
>= 1,000	34	0.39 %	18	0.04 %	52	0.09 %
Total	8793	100.0 %	47673	100.0 %	56466	100.0 %

Even as larger companies increasingly adopt a service model for interactions between internal business units, insights into the mechanics of small teams may hold true regardless of whether small teams are located within large companies or small firms (Cusumano et al. 1997; Kostamovaara et al. 2007). Therefore, the study of small teams is valid because of and despite what is suggested by the data in Table 8.

Fayad et al. (2000) impacts of team size on software engineering method are shown in Table 9:

Table 9 Factors Relating Team Size and Software Development Method Use

Factor	Impact
Company Size	As was shown in Table 8, the number of companies requiring software development methodologies for small teams is in the majority.
Development Mode	While agile methods have challenged the efficacy of the contract model for software development (which a clear customer for whom the work is being done), those developing COTS software or those providing non-contract services within an organization do not fit the contract model. Methodologies for small teams will have to go beyond the contract model.
Development Speed	While innovations in software development methodologies since the 1990s have largely focused on agility and adaptation, there is a very real need for rapid application development: competition and connectivity. With the Internet, barriers to market entry with software are extremely low, the volatility of markets is high (software piracy) and customer expectation (competition) is fierce. Small companies need to get products to customers and receive billing monthly and/or quarterly, not multi-year.
Development Size	Modern code configuration and management tools, in addition to deep code libraries and intelligent editors, allow a small team to produce a larger number of well-tested and effective lines of code. Furthermore, frequent and iterative updates and version is often expected.

The literature on software engineering (and information systems development) commonly misjudges these aspects of small-team development. Especially in the case of small teams in the small-shop environment (where the team is the company) or within startup companies[6], the considerations for the selection of a software development methodology may not be in step with the literature on software engineering. Some desirable properties in a software development methodology for the small-team or small-shop are show in Table 10 (Fayad et al. 2000):

Table 10 Desirable Properties of a SD Methodology for Small-Teams and Small-Shops (Fayad et al. 2000)

Method Property	Desirability for the Small Team
Reuse	While reuse is a tenet of modern and modularized development, a small team may get more value by concentrating on iteratively releasing a working product. A small team may not be able to afford the luxury of the time required to perfectly engineer their code base when schedules are tight. Furthermore, especially in the case of web development, requisite technologies to fulfill customer demand may have shifted; rendering an old codebase obsolete[7].
Cost estimation	Extrapolating across prior projects may be difficult unless templates are used. If innovation and competition based on novelty is required, as is typically the case when developing web applications, then traditional cost estimating strategies are confounded.
Requirements Stability	Rapid competition and customer expectations for quick product delivery make advice on traditional requirements gathering somewhat passé for many small teams. For many agile methods, requirements are solicited from the customer incrementally and through frequent releases of working software.
Limited Resources	In a small team, especially those operating within small companies, the traditional divisions of labor favored by large-scale development methodologies are not possible. The developers in the small team must wear many hats and understand a multitude of roles.

[6] In the late 1990s and throughout the 2000s, many of the most impactful and innovative software has arisen from small startup software development teams and companies.

[7] Imagine the Perl CGI programmer when she then had to transition to PHP or the PHP programmer who then had to transition to AJAX and Ruby-on-Rails. These codebases are not reusable across the technologies and could, at best, be reused as mashups.

Incremental and Frequent Releases	Small teams need to regularly release software or they are not paid. In this situation a full run of traditional analysis and design techniques is not possible. It is not uncommon for unfinished and/or beta-stage software to be released and then subsequently patched after release. A small team can't afford to wait until the software is perfect[8].

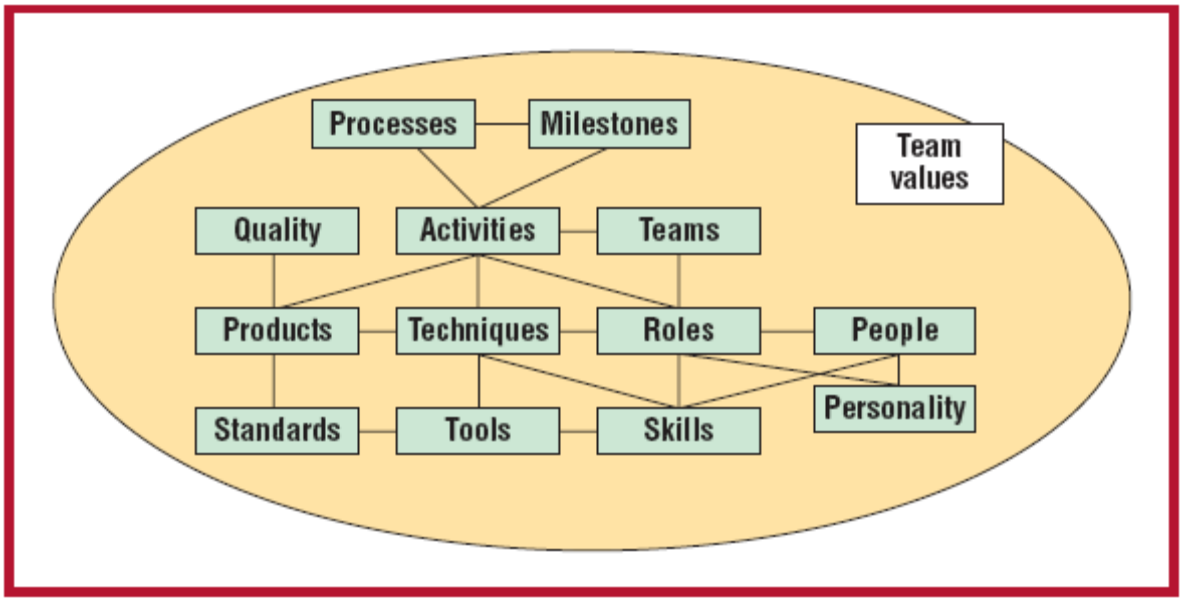
It is clear that Fayad et al. (2000) have well-anticipated the advent of agile software development methodologies by clearly spelling out the needs of small teams over methods which focus on larger-scale projects.

2.4.1 Appropriate Software Development Methods for Small Teams and Small Shops

If the majority of large-scale and large-team methodologies are inappropriate for small teams, then criteria for small-team development method selection is required. Furthermore, it would be important to distinguish, through classification, the nature and character of a software development methodology in regards to team size. Some have suggested that methodology selection can be linked to team size, project size and project criticality by classifying methods on a continuum from lightweight methodologies to heavyweight methodologies (Cockburn 2000). When all of the constructs affecting team size are considered, it becomes clear that a set of principles can provide a rubric by which a small team could find the best methodological fit for their specific situation. Cockburn (2000) depicts relationships between the elements of a software development methodology in Figure 6.

[8] This phenomenon is very apparent in the computer games industry

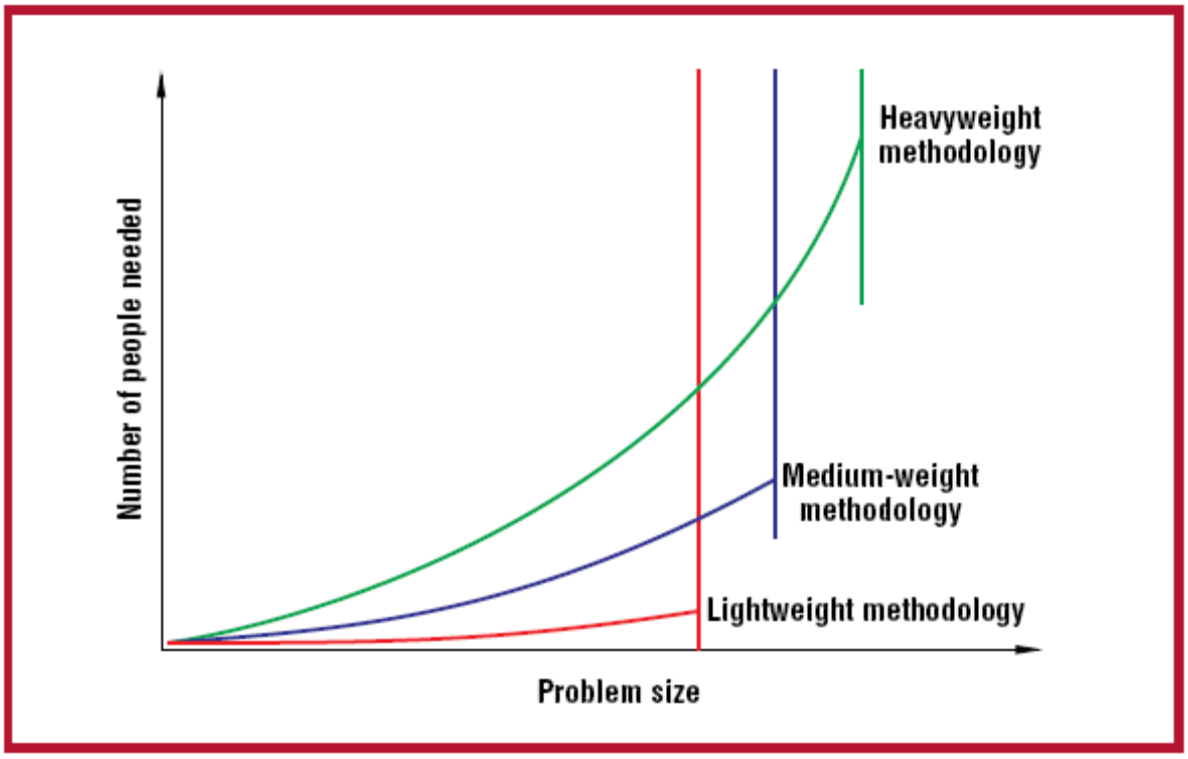
Figure 6 Elements of a Software Development Methodology (Cockburn 2000)



Cockburn (2000) suggests that adopting a set of principles strikes the right balance between the various methodological elements:

- **Size** – Methodology size depends on project size; heavyweight for larger projects and lightweight for smaller projects. This also means that more elements from the model in Figure 6 would be present in a larger methodology.
- **Criticality –** If a system failure will cause significant losses to comfort, property, money and/or life, a heavyweight method is required.
- **Cost –** Increases in methodology size add exponentially to cost (see Figure 7).
- **Communication Richness –** Methodologies which allow for interactive and face-to-face communication will be the most effective (see Figure 8).
- **Project Priorities –** The degree to which the customer wants the project completed in a timely manner, defect-free and transparent/visible will influence method selection.
- **Embedded Assumptions on Risk –** Methodologies are largely risk mitigations against known adversities. Project risk tolerance can be matched to the risk assumptions inherent within a methodology.

Figure 7 Problem Size and Increases in People Costs (Cockburn 2000)



Cockburn (2000) uses these principles to influence his framework for method selection for small teams. Cockburn's (2000) framework for small-team software development methodology selection allows for differentiation according to the principles discussed above and project elements related to size, criticality and priorities. As shown in Figure 9, each cell contains values for the methodology selection variables, where any given cell can contain multiple methodologies. The principles discussed above would guide method selection within the framework. If multiple methodologies occupy a cell, then team culture and project priority would likely be deciding factors.

Figure 8 Communication Richness and Methodology Effectiveness (Cockburn 2000)

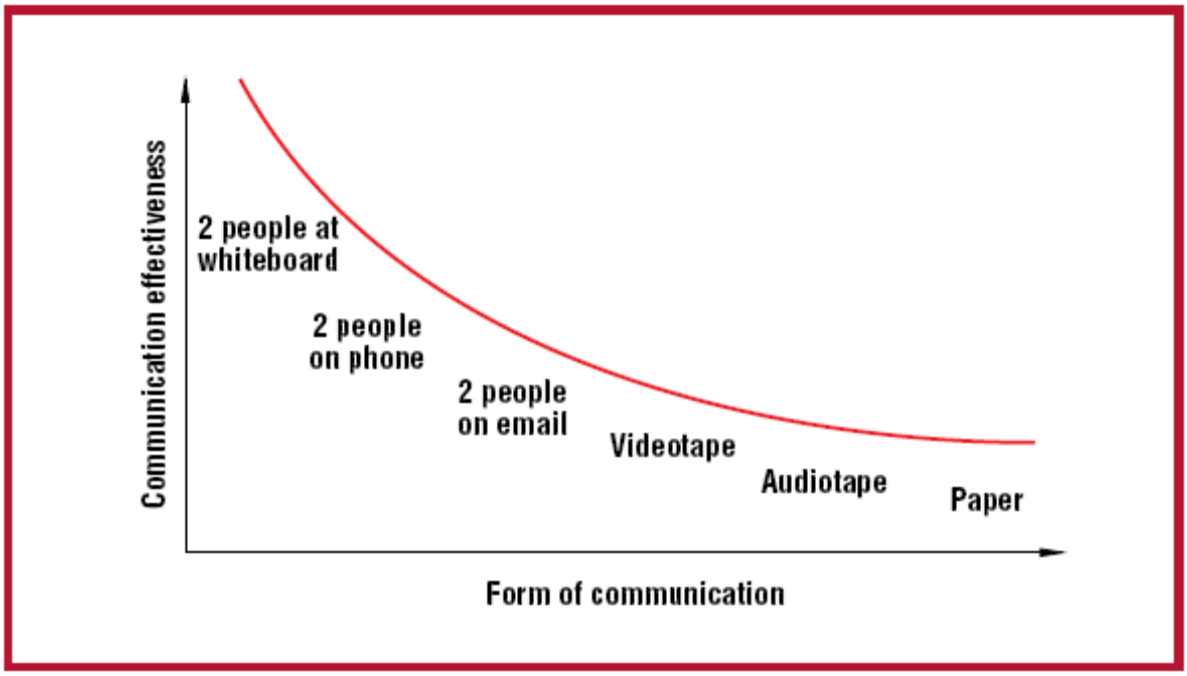
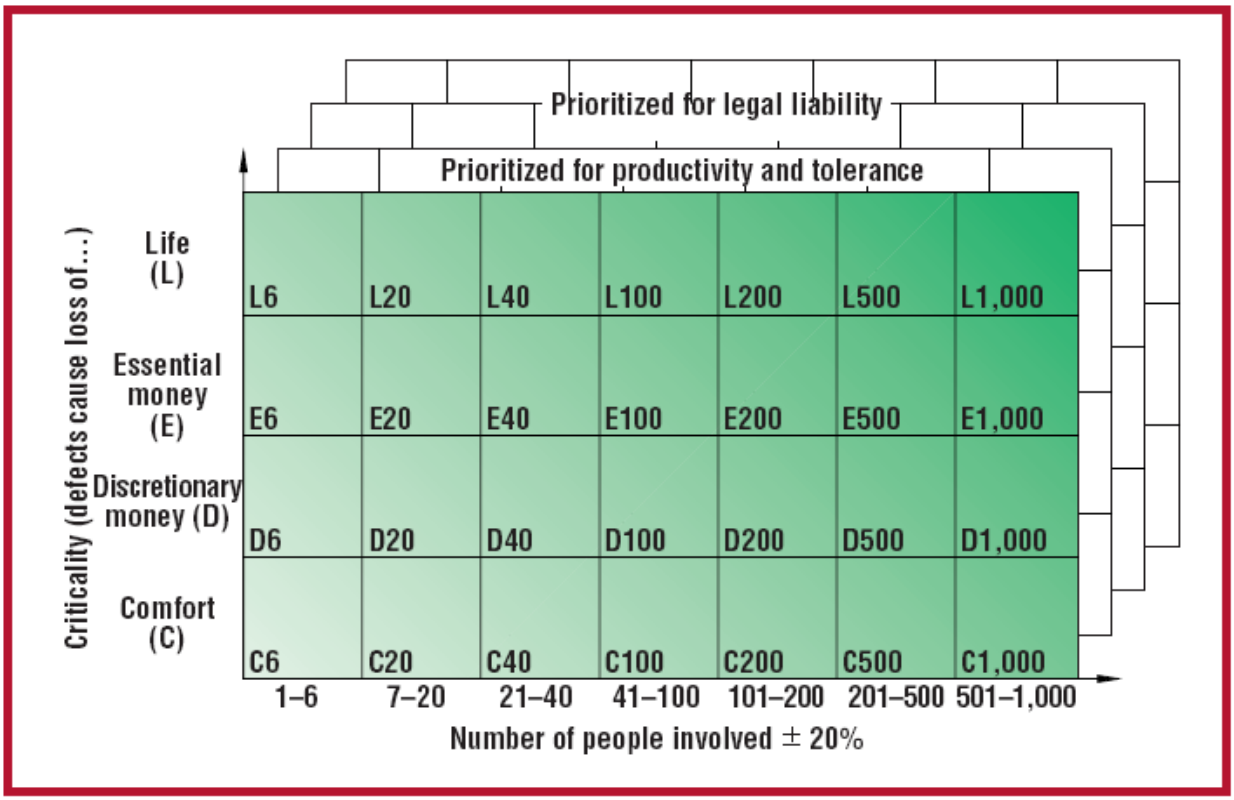


Figure 9 Cockburn's Methodology Selection Framework (2000)



While Cockburn's (2000) methodology selection framework is one among many possible approaches to method selection for small teams, Cockburn's framework is principle-driven and illustrates the considerations which a small team would face. Furthermore, Cockburn enjoys a modicum of prestige and is renown in the area of methodology development (Boehm et al. 2004; Cockburn 2002; Cockburn et al. 2001).

2.4.2 Human Factors in Small-Team and Small-Shop Software Development

Practitioners in a small shop will commonly share and assume a multitude of responsibilities and roles; the practitioners' individual strengths and capabilities become important (Blackburn et al. 1996; Cragg et al. 1993; Cusumano et al. 1997; Ginige et al. 2001; Gorla et al. 2004; Kraut et al. 1995; Lehman 1998; Rettig et al. 1993). Gorla and Lam (2004) recommend personality assessment for small teams in order to maximize effectiveness: "personality type analysis can help take the guesswork out of putting together a high-performance software project team" (p. 79). In several studies, it has been demonstrated that human considerations outweigh technical considerations as factors for project success (Gorla et al. 2004). Gorla and Lam (2004) conduct survey research to determine the effects of personality type in small software development teams using the Myers-Briggs Type Indicator (MBTI).

The utility of Gorla and Lam (2004) study is the degree to which developer personality and team culture influence team effectiveness. The results of the study reveal the influence personality type in Figure 10.

Figure 10 Personality Type and Small Team Performance (Gorla and Lam 2004)

Criterion	Personality Dimension	Team Performance*	R-Square	Significance
Team Leader	Information Gathering	Intuitive > Sensing	0.297	.0130
	Decision Making	Feeling > Thinking	0.297	.0130
System Analyst	Decision Making	Thinking > Feeling	0.482	.0038
Programmer	Interaction with the World	Extrovert > Introvert	0.505	.0020
Heterogeneity Leader-Member	Interaction with the world	Extrovert~Introvert	0.181	.0612
	Information Gathering	Intuitive~Sensing	0.595	.0001
Heterogeneity Member-Member	All dimensions	--	--	Not significant

* > means outperforms, ~ means difference.

Gorla and Lam (2004) suggest that a small team (or small shop), lacking a strong hierarchical political and leadership structure, must establish a team culture which carefully balances and accommodates personality types. In this sense, a small team needs personality heterogeneity between a team leader and other team members (Gorla et al. 2004: 82). Specifically, Gorla and Lam (2004) find that MBTI personality categories are vital to ensure personality heterogeneity in the team.

An equally important consideration in method selection is an individual team member's skill level; a small team in a small shop must rely on limited resources endemic to their team size and available man-hours to devote to a task (Boehm et al. 2004; Brooks 1995). For this reason, a software development methodology for the team in a small shop demands a greater amount of skill from individual team members (Boehm et al. 2004: 46). Cockburn (2002) classifies the degrees of individual software development methodology comprehension into three levels; each level indicates advancement in the ability to utilize a method effectively, adaptively and creatively. Boehm and Turner (2004) extend Cockburn's (2002) original levels to further stratify the first level (Table 11). In essence, both argue that a small team will operate effectively when

the degrees of freedom are high; a method with extensive planning and predictive requirements will not likely suit advanced users (those at a higher level).

Table 11 Levels of Software Method Understanding and Use (Boehm and Turner 2004)

Level	Characteristics of the Level
3	Able to revise a method (break its rules) to fit an unprecedented new situation.
2	Able to tailor a method to fit a precedented new situation.
1A	With training, able to perform discretionary method steps (changing the nature of a method's steps). With experience, can achieve level 2.
1B	With training, able to perform procedural method steps (sticking to the plan). With experience can master some level 1A skills.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.

Cockburn (2000) states: a methodology denotes "…everything about how a team repeatedly produces and delivers systems; whom they hire and why, what people expect from coworkers, the processes they follow, their conventions, work products, and even their seating arrangements" (Cockburn 2000: 65). Additionally, Cockburn (2000) suggests that methods entail "…techniques and drawing notations" for the activities, processes and techniques, skills and tools embedded within a software methodology (Cockburn 2000). While Figure 6 appears to place a team's values as an outlier in the model, it seems clear that team values in a small shop are related to personality mix, skill mix, experience, knowledge of standards, and effectiveness. Large-scale and heavyweight software development methodologies often seek to control these human aspects. However, in some cases practitioners may find comfort in the predictability, the prescriptions, the proscriptions and clarity embedded in the policies and procedures of process-heavy and heavyweight methods. This "production-line" environment, where each person's tasks are well-defined, may be preferable in cases where user skill is low (Boehm et al. 2004: 49).

2.4.3 Process Diversity in Small-Team and Small-Shop Software Development

Cockburn's (2000) framework for software development methodology selection can be used as a guide to understand process diversity. Process diversity promotes the idea that no single methodology will fit all software development project needs. Instead of focusing on what a particular methodology does, Cockburn's (2000) principles (Figure 6) allows a team to select multiple methodologies on multiple dimensions of the model. In this sense, method selection, adoption and adaptation embodies the environmental and structural aspects of team culture, team size, and problem characteristics. Often, a methodology is judged by the effectiveness of its outcomes than on the particulars of its processes and structural elements (Lindvall et al. 2000). In the end, a small team uses a methodology to produce a software artifact which pleases the customer – whatever methodology which accomplishes this is a successful methodology.

While it is desirable to gauge the utility of a methodology for small team software development on its ability to facilitate a viable and successful software development process, an aspect of that viability is in team development. A method could be ill-suited to the team but capable of producing a viable product. Thus, while creating a working product which makes the customer happy is clearly paramount, adopting methods which enhance team development and learning is perhaps equally important. It is important to distinguish process from methodology as a small team's software process can contain a number of methodologies to facilitate the delivery of successful software. While Cockburn's (2000) framework suggests a means for arriving at a methodology selection, the dynamics of the problem space and the dynamics of the team may suggest hybrid approaches. Any method for small-team software development should

not confine and restrict that team's process and, moreover, the selected method or methods should promote team building and learning.

In 2000, two special issues of IEEE Computer focused on matters relevant to small-team software development methods and processes: one issue on process diversity (Cockburn 2000; Florac et al. 2000; Johnson et al. 2000; Rising et al. 2000; Sutton 2000; Williams et al. 2000b); and one issue on software engineering in-the-small (Dybå 2000; Nunes et al. 2000; Rajlich 2000; Russ et al. 2000). These special issues are notable for their content and timing. These special issues were published as agile software development methodologies were on the cusp of wider demand and also the popularization in increased demand for a dynamic and commerce-driven World Wide Web.

2.4.4 Critical Factors and "Home Grounds" for Small-Team Software Development

Boehm and Turner (2004) offer five critical factors for the selection of a suitable small-team software development methodology. Table 12 depicts these factors and their implications for small teams and large teams.

Table 12 Small Team and Large Team Methodology "Home Grounds" (Boehm and Turner 2004)

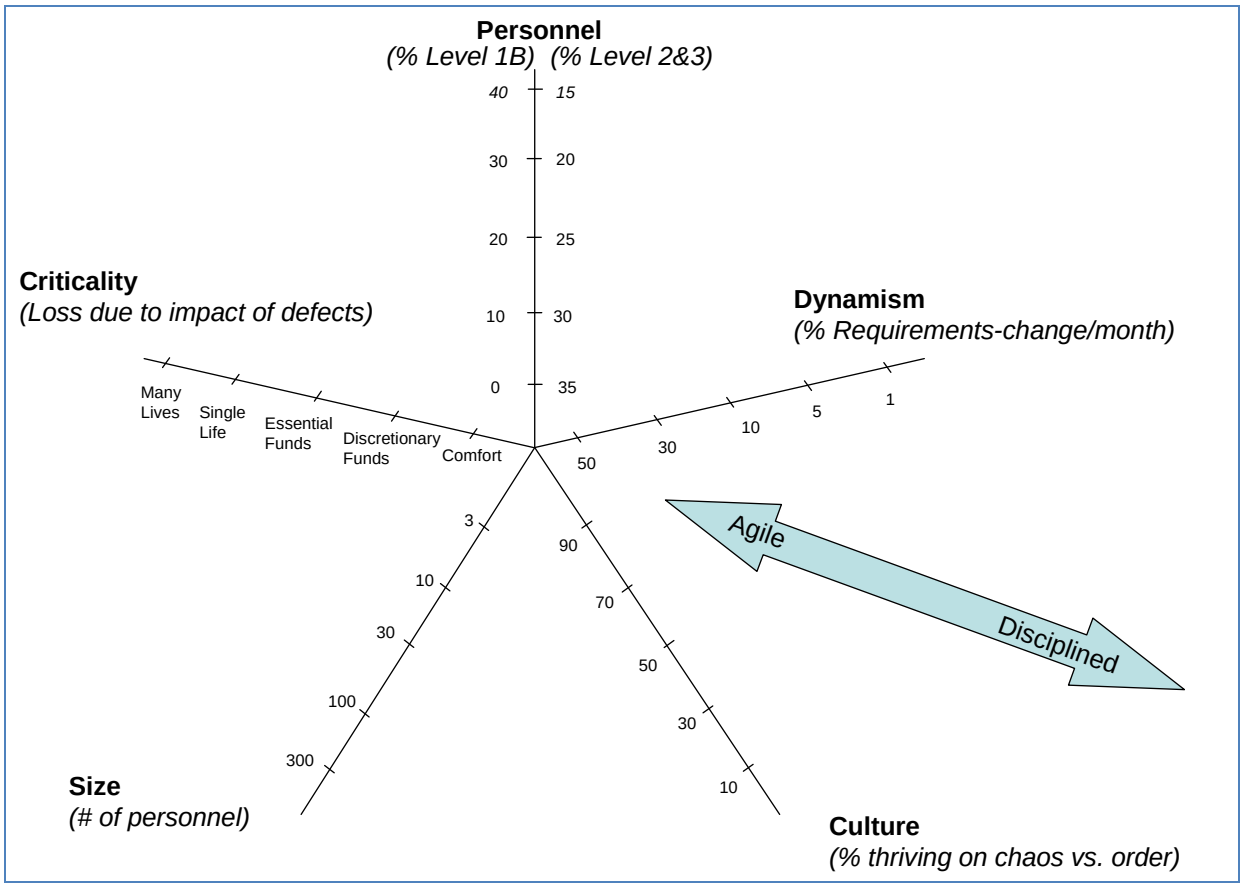
Characteristics	Small Team Oriented	Large Team Oriented
Application		
Primary Goals	Rapid value; responding to change	Predictability, stability, high assurance
Size	Smaller teams and projects	Larger teams and projects
Environment	Turbulent; high change; project-focused	Stable; low-change; project/organization focused
Management		
Customer Relations	Dedicated on-site customers; focused on prioritized increments	As-needed customer interactions; focused on contract provisions
Planning and Control	Internalized plans; qualitative control	Documented plans, quantitative control

Communications	Tacit interpersonal knowledge	Explicit documented knowledge
Technical		
Requirements	Prioritized informal stories and test cases; undergoing unforeseeable change	Formalized project, capability, interface, quality, foreseeable evolution requirements
Development	Simple design; short increment; refactoring assumed inexpensive	Extensive design; longer increments; refactoring assumed expensive
Test	Executable test cases define requirements, testing	Documented test plans and procedures
Personnel		
Customers	Dedicated, collocated CRACK* performers	CRACK* performers, not always collocated
Developers	At least 30% full-time Cockburn level 2 and 3 experts; no Level 1B or -1 personnel**	50% Cockburn Level 2 and 3s early; 10% throughout; 30% Level 1B's workable; no Level -1s**
Culture	Comfort and empowerment via many degrees of freedom (thriving on chaos)	Comfort and empowerment via framework of policies and procedures (thriving on order)
* Collaborative, Representative, Authorized, Committed, Knowledgeable ** These numbers will particularly vary with the complexity of the application		

Figure 11 depicts relationships between the dimensions in Table 12. While the original diagram in Boehm and Turner discusses agile vs. plan-driven methodologies (discussed in the next section of this chapter), these characteristics are approximately true for small vs. large-scale software development as agility and adaptation are among the more valuable traits for a useful small-team software development methodology. Thus a small team will best utilize methods when developers are skilled individuals who are adaptive to change and possess the intuition and

Figure 11 Dimensions Affecting Method Selection (Boehm and Turner 2004)

judgment to adopt and adapt a methodology as is required in the problem space.



We can conclude this section by stating that a successful software development methodology for a small team will balance factors on personnel, dynamism, culture, criticality and size. Thus, there is no single methodological aspect which guarantees small team success. As Frederick Brooks intimates in his seminal paper "No Silver Bullets":

> *…The central question of how to improve the software art centers, as it always has, on people… We can get good designs by following good practices instead of poor ones. Good design practices can be taught. Programmers are among the most intelligent part of the population, so they can learn good practice. (Brooks 1995: 202)*

In the case of selecting a software development methodology for small teams, it is human factors which will most likely determine the degree to which a software development method will enable or disable team effectiveness.

2.5 Comprehending Agile Software Development Processes

This section is concerned with the following fundamental questions related to agile methods: What are agile methods? How are agile methods similar to and different from existing systems development methods? Are agile methods better than existing methods and, if so, why? Lastly, in what settings and for what purposes would agile methods be most appropriate? In developing possible answers to these questions, this section proceeds in the following manner. First, an overall introduction to the agile methods movement is offered. This introduction will use, as much as is possible, the language, thoughts and ideas of practitioners and researchers most responsible for creating the agile methods movement. Next, agile methods are contrasted to other software development methods such that a general taxonomic understanding of software and systems development methods is developed. We then progress to a discussion on the *prima facie* merits and limitations of agile methods. Next, the conditions under which agile methods would be most suitable are discussed. The section will then conclude with final thoughts concerning the future of agile methods and possible new directions for the agile movement.

2.5.1 Background on Agile Methods

The agile software development movement arose out of dissatisfaction with traditional software engineering methods (Martin 2003). Although the literature on agile methods reveals a curious retention of the language of software engineering, apparently framing these methods in the metaphor and paradigm of engineering, the literature also makes it clear that agile methods

constitute a break from the dominant paradigm of engineering. This paradox is a primary motivator of this research.

Early examples of agile methods are Scrum, Crystal Clear, Extreme Programming, Adaptive Software Development, DSDM and others (Boehm et al. 2004). All are considered to be faster and more people-centric than the more traditional waterfall model of the SDLC. With agile methods, customers are satisfied by rapid and continuous delivery of software in the short term. Time scales are in units of days and weeks rather than months and years. Changes to requirements are always welcome due to the iterative nature of the methods. Agile methods are people-centric as they value close and daily cooperation between business people and developers; this daily cooperation is usually conducted face-to-face. The motivated individual, and self-organizing teams of motivated individuals, typically finds the most success with agile methods as their focus on the individual emphasizes inter-personal trust over contracts and other legal inducements.

If agile can be summarized in a phrase, that phrase would be "people over process." In this sense, as a framework, agile methods provide principles, patterns and practices for the software developer to follow. However, according to Alistair Cockburn, an authority on agile methods, "… process and technology are second-order effects… The first-order effect is people" (Martin 2003: 1). Thus the goal of most agile development methods is to foster the development of collaborative and self-organizing teams.

Table 13 Manifesto for Agile Software Development (Fowler et al. 2001)

Manifesto for Agile Software Development
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value: • Individuals and interactions over processes and tools

- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more (Fowler et al. 2001).

Table 14 Principles behind the Agile Manifesto (Fowler et al. 2001)

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

Many of the most popular and accepted agile software development methods were created and perfected by the authors of the Agile Manifesto in Table 13 and the principles behind the manifesto in Table 14. Some of these methods authored by these practitioners include SCRUM, Crystal, Feature Driven Development, Test Driven Development, Adaptive Software Development and, most significantly, Extreme Programming (XP).

2.5.2 Selecting an Agile Method for Small-Team Software Development

Each of the commonly-recognized agile methods has a different organizational scope, SDLC focus and set of constraints (Boehm et al. 2004). Table 15 provides a brief list and description of the more popular agile and iterative methods. Table 15 provides background on several of the more popular agile methods and provides a rationale for selecting the agile method traits best suited to small teams.

Table 15 Comparison of Agile and Iterative Methods

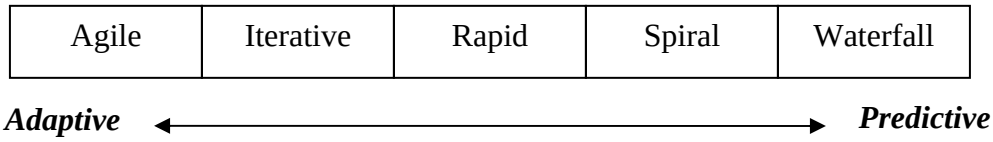
Agile Method	Method Synopsis	Applicability to Small-Team Development
Scrum (Schwaber et al. 2002)	Generally a project-management approach which borrows from a rugby metaphor. Emphasizes 30-day cycles and daily 30-minute meetings. Agility is implied in limited planning no further than the scrum cycle.	Indirect
Adaptive Software Development (Highsmith 2000)	An evolution from Rapid Application Development which provides for a speculate-collaborate-learn cycle which specifies continuous learning and adaptation.	Indirect and omits individual developers.
Lean Development (Womack et al. 1991)	A project-management approach based on risk management and lean manufacturing.	None
Crystal (Cockburn 2002)	A framework of related methods that address the variability of environment and the specific characteristics of projects. Methods vary by colors indicating team size and criticality of the project.	Direct in the case of single teams.
Extreme Programming (Beck 1999)	The most widely-recognized and used agile method. Value based approach focusing on communication, simplicity, feedback and courage. Very principle-driven and closely related to the agile manifesto.	Direct and tailored to small teams and individuals.
Dynamic Systems Development Method (Stapleton	A large framework of methods with a large European user base. Has a strong process management emphasis operating through a	Direct in the case of single teams.

Agile Method	Method Synopsis	Applicability to Small-Team Development
1997)	five-phase process	
Rational Unified Process (Jacobson et al. 1999)	Oriented towards Rational/IBM's Unified Modeling Language. Streamlines predictive methods using risk-driven spiral processes. Focus on a four-phase life cycle with exit criteria and phase milestones. Risk management techniques are emphasized.	Direct in the case of single teams.
Team Software Process (Humphrey 2000)	A configurable plan-driven development process for teams. Templated and focuses on concrete roles and scripts. Clings to plan-driven approaches.	Strong emphasis on single team.
Feature-Driven Development (Palmer et al. 2002)	Simple processes, efficient modeling and short iterative cycles focusing on the customer. Has an architectural emphasis on eliciting the best design upfront.	Weak – Emphasis on Multi-teams.
Capability Maturity Model Integration (Ahern et al. 2001)	A high-level model on which the Capability Maturity Model for Software is built. Provides and extensible framework for methods. Includes systems engineering, supplier selection and integrated process and product development. Provides a reference model rather than a method.	While not a method, does provide some direction for single teams.
Capability Maturity Model for Software (SW-CMM) (Paulk et al. 1995)	Serves as a checklist/roadmap for maturing software development processes. Specifies five levels of maturity and tends to result in heavyweight processes. Distills a wide range of best practices for software development	Direct in the case of single teams. Not entirely agile.
Personal Software Process (Humphrey 1995)	Provides an operationalization for individual developers involved in SW-CMM. Seeks improvement for individual programming skills. Specifies four levels of improvement.	Direct in the case of the Individual.
Cleanroom (Prowell et al. 1999)	Uses mathematical proofs for verification and reliability certification. The object is to develop defect-free code upfront. Specifies a complete discipline across the entire SDLC. Creates high standards which are difficult to achieve.	Direct. Focuses on single teams and individuals.

2.5.3 Positioning Agile Methods

Agile methods can be situated within a wider spectrum of software development methods in order to understand where agile methods "fit." One common theme emerges whereupon agile methods were often developed in response to perceived deficiencies inherent within the prevailing software engineering methods. The "predictive"[9] software engineering methods were increasingly viewed as inappropriate in face of rapid and constant change. The literature suggests that agile methods reside on the adaptive end of a spectrum running back towards predictive methodologies (Abrahamsson et al. 2003; Fowler 2005; Williams et al. 2003). This spectrum is represented in Figure 12 below where software development methods are classified as either adaptive or predictive in nature (Abrahamsson et al. 2003).

Figure 12 Classifying Software Development Methodologies from Predictive to Adaptive (Abrahamsson et al. 2003; McConnell 2004b)



The predictive-adaptive continuum could give the impression that agile methods represent step-wise refinement in a natural progression from older methods to newer methods. In this sense, new methods will arise in response to changes in the environment which render older methods less effective. The literature on agile methods support this concept both explicitly and implicitly (Cockburn et al. 2001; Fowler 2005; Fowler et al. 2001; Highsmith 2002; Lindstrom et al. 2004; Nerur et al. 2005; Newkirk 2002; Subramaniam et al. 2006; Williams et

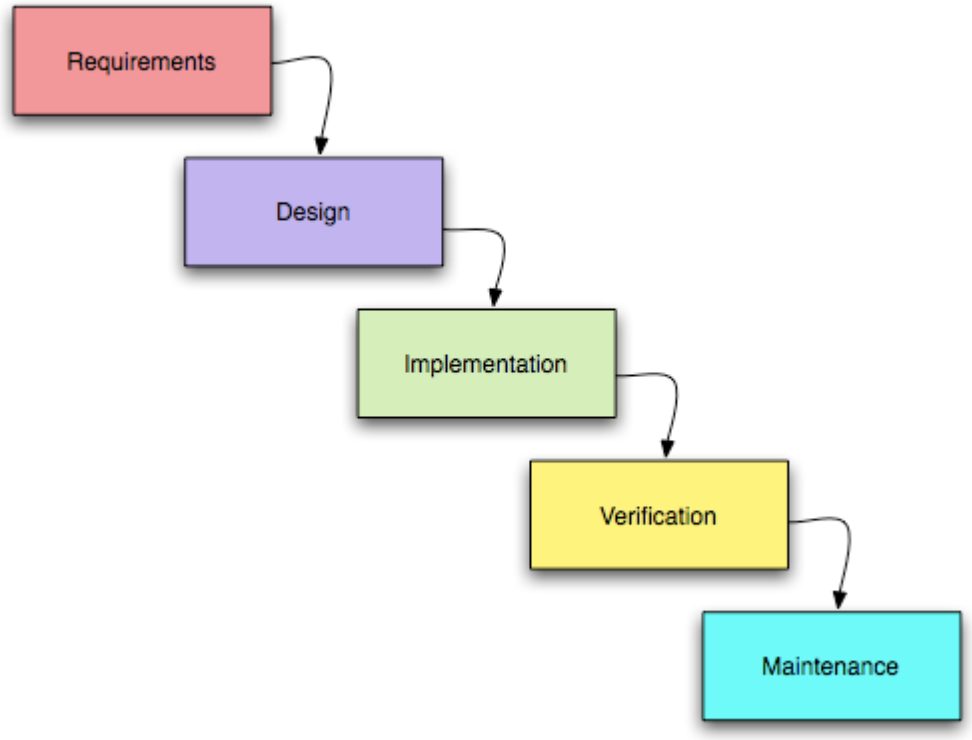
[9] The waterfall model is predictive as it seeks to manage risk and costs to mitigate for the effects of change

al. 2003). This progression supports the premise that agile methods represent a paradigm shift away from engineering and process-driven methods towards an emergent adaptive paradigm.

It would require several volumes to recount the history of systems and software development methodologies; such an effort would be beyond the requirements of this literature review. However, we must situate agile methods within the larger scope of systems development methods if we are to appreciate the advantages and disadvantages which agile methods have to offer. In this section, the impetus for the arrival of adaptive and agile processes is traced and presented as a paradigmatic shift born of the necessity of practice. The work of Fred Brooks is also used to reflect on the reasons why methods which worked very well in the 1970s had themselves become challenged by the 1990s (Brooks 1995:264).

Scholars such as Boehm and Brooks offer rich insight into the nascent era of systems development throughout the 1950s, 1960s and into the 1970s. They chronicle the development of engineering-oriented methods which addressed the "software crisis" when several software development projects had experienced atrocious cost overruns and delays. At the time, process and engineering-oriented solutions were adopted such that software projects were more maintainable (Boehm 1996). These plan-driven and engineering-oriented solutions entailed discrete project milestones and phases, well expressed within the waterfall software and systems process model, which provided a systematic means of managing software development projects. Figure 13 depicts the familiar and common steps of the waterfall software and systems development process model which is exemplary of engineering-oriented software development processes.

Figure 13 The Waterfall Model of the SDLC



The waterfall model was generally sufficient for many years subsequent to its introduction in the 1970s (Boehm 1996). The waterfall process model expresses a software life-cycle where each discrete step leads towards the completion of a subsequent step culminating in a finished product. With the metaphor of a product life-cycle, revision and refinement would require re-entry into the entire waterfall process model in a linear fashion (Boehm 1996).

Of course, few projects in reality are as linear as the waterfall model would suggest – therefore adjustments to the basic tenets of the model began to arise (Boehm 1996). The steps suggested by the waterfall model may, in practice, run parallel to each other or require sub-processes which themselves utilize the steps of the waterfall model. Over time, the waterfall model was modified in a series of compromises motivated and wrought by the empirical experience of practice.

The spiral model is one such compromise which was developed to address the deficiencies of the abstractions of the waterfall model (Boehm 1988; Boehm 1996). In a cyclical fashion, we can observe that changes and challenges in the environment of practice led to the initial development and adoption of the process-driven, plan-driven and engineering models for software and systems. If the environment of business and practice were fairly stable and static, the waterfall model would have continued to provide a "…sequence of milestones around which people could plan, organize, monitor, and control their projects" (Boehm 1996: 73). However, "…just as the waterfall model was becoming fully elaborated, people were finding that its milestones did not fit an increasing number of project situations" (Boehm 1996: 73). We can interpret this to mean that the context and frame, or the understanding of the context and frame, under which the waterfall method had been developed was changing. We can trace this trend throughout the software and systems development literature of the 1970s, 1980s and 1990s as evidenced by the many hybrid variations on the waterfall model which introduced the idea of iterations between and among the discrete steps of the waterfall model (Boehm 1996).

2.5.4 Cracks and Fissures in the Old Paradigm

Agile methods reject the prevailing paradigm of software engineering (which is founded in the Positivist ontology, theory, models, epistemology and paradigmatically reinforced in the Positivist social ontology) as inadequate and inappropriate in some, but not all, cases. In this sense, agile methods do not exclude the prevailing paradigm entirely, but rather illustrate where the prevailing paradigm is less important to the emerging paradigm of agility. This begs the question: is the advent of agile software development methods an instance and example of Kuhn's paradigmatic revolution? There are various reasons to answer this question in the

affirmative and the negative. One possible reason why agile methods do not represent a Kuhnian revolution is that the inertia and dominance of the engineering metaphor overshadows the agile metaphor. In this sense, agile methods are often used within the software engineering framework and paradigm. However, there is also ample reason to consider agile methods as representative of a new paradigm more representative of agile processes, values and principles.

In many ways, the agile methods movement has all the hallmarks of a Kuhnian paradigm shift whereupon new ideals, goals and values have arisen as an older paradigm fails to explain and predict the experience of practice. In this sense the agile methods movement has characteristically rejected an entire class of *heavyweight* processes in favor of *lightweight* processes. These *lightweight* processes espouse the values of iterative production, interactions with customers, rapid prototyping and agile responses to change (Fowler et al. 2001; Lindstrom et al. 2004:42). Moreover, the principles informing the agile movement did not appear overnight; the conditions supporting this movement are the result of reactions, accumulated over time, to the use of traditional software development methods (Larman et al. 2003).

Fred Brooks, in revisiting his seminal book *The Mythical Man Month*, (1995) provides many after-the-fact insights as to why the process-driven, plan-oriented models such as the waterfall model, had become increasingly inadequate in practice (Brooks 1995). In a retrospective and closer examination, Brooks highlights a number of cracks and fissures which appear in process-oriented and plan-driven software and systems development approaches. When "…existing institutions have ceased to adequately meet the problems posed by an environment that they have in part created" (Kuhn 1996:92), a pre-requisite to a paradigmatic shift presents itself. Kuhn would also explain why many modifications to the waterfall model did not break away entirely from the prevailing paradigm of plan-driven and process-oriented

methodologies. Whereas feedback in practice suggested that modifications to the waterfall model were cause to question the fundamental premises of the engineering paradigm, most adjustments were made within the prevailing paradigm inherent in the waterfall model. Thus, the fundamental paradigm wasn't challenged and new action strategies arose from unchanged beliefs. However, a parallel series of ideas, entirely contrary to the plan-driven and process-oriented engineering methods, also came to prominence in the 1990s; those being categorized as agile methods today.

Brooks (1995) summarizes "…the basic fallacy of the waterfall model" as a problem of flexibility and adaptability (Brooks 1995:266). In the waterfall model, it is assumed that

> *…one builds a whole system at once, combining the pieces for an end-to-end system test after all the implementation design, most of the coding, and much of the component testing has been done. (Brooks 1995:266)*

Brooks, and others, have since realized that while the abstract dependencies between the waterfall model's steps might be true in concept, actual software development experiences iterations within and between these steps, and, in general, non-linear movement within and among these steps. Brooks (1995) had come to recognize that iterative sub-cycles within the development cycle are a reality; that regular and iterative builds, such as those advocated in agile methods, are appropriate (McConnell 2004a); and that the various incremental-build/rapid-prototyping software and systems development approaches met with greater successes than the waterfall model (Brooks 1995).

Lastly, Brooks (1995) concludes his revisit of changes in software and systems development over a 20 year period from 1975 to 1995 with the realization that methodologies will continue to address the same problem domain regardless of the particulars of any single

method or changes in the environment. Brooks indicates that "…the distinctive concerns of software engineering are" (Brooks 1995:288):

- "How to design and build a set of programs into a system"
- "How to design and build a program or system into a robust, tested, documented, supported product"
- "How to maintain intellectual control over complexity in large doses"

Thus, Brooks suggests that the fundamental aims of software and systems development methodologies will remain the same, despite paradigmatic shifts in method use. This section has established the fertile ground necessary for a paradigmatic shift towards agile methods. The next section will discuss agile methods as a philosophical embodiment of a number of adaptive and iterative methods.

2.5.5 The Emergence of Adaptive and Iterative Methods

Larman and Basili (2003) trace the roots of agile development in a digest of 70-plus years of Iterative and Incremental Development (IID) and highlight key moments in the development agile methods. Larman and Basili (2003) make it clear that the waterfall method was never intended to provide a strict guideline and formula for software development: The waterfall model is an ideal which abstracts the software and systems development process for management rather than for developers (Parnas et al. 1986). For decades, researchers and practitioners of software and systems development have been acutely aware that "the picture of the software designer deriving his design in a rational, error-free way from a statement of requirements is quite unrealistic" (Parnas et al. 1986). In this sense, it is unfair to castigate the waterfall model as the sole hindrance of progress: process-oriented approaches serve a purpose and should be

used when appropriate. Despite this, the dichotomy between IID and waterfall-oriented methods is very real and has created an atmosphere ripe for a statement such as the Agile Manifesto.

Abrahamsson et al. (2003) provide a thorough comparative analysis of IID methods and how they came to be called "agile" methods. Their work is valuable as it provides advice to software developers and software project managers on when and why agile methods should be used. If a manager or developer asks themselves "are agile methods just a fad or will agile methods provide substantial guidance over and above existing and accepted methods?" Abrahamsson et al.'s (2003) comparative analysis provides substantial responses to this question.

As is made clear in the Larman and Basili (2003) paper, the family of methods now known as agile methods each arose at different points in time, evolved at different rates, and arose under different circumstances and for different reasons. As agile methods progressed and developed, a single and consistent theme arose in their use: the "…document-driven single-pass sequential life cycle" had somehow failed (Larman et al. 2003:55). In a 1998 study of 23,000 software development projects, the "…top reasons for project failure… were associated with waterfall practices"; the report goes on to mention that "…IID practices tended to ameliorate the failures" which arose from over-reliance on the waterfall model (Larman et al. 2003:54). Again, this is not a failure of the waterfall process model so much as it is an industry relying too heavily on simplistic and parsimonious conceptions of the software and systems development process.

The period during which the majority of IID methods emerged and blossomed was during the 1990s. Adaptive Software Development, Agile Modeling, Crystal Family, Dynamic Systems Development Method, Extreme Programming, Feature-driven Development and Internet-speed Development, among others, all emerged and took shape in the 1990s. Again, it is quite likely

that the explosion of computer users brought on by the popularization of the Internet via the World Wide Web has a key reason for the emergence of IID methods. Figure 14 presents an evolutionary map of these methods (Abrahamsson et al. 2002; Abrahamsson et al. 2003).

Figure 14 Evolutionary Map of Agile Methods (Abrahamsson et al. 2003)

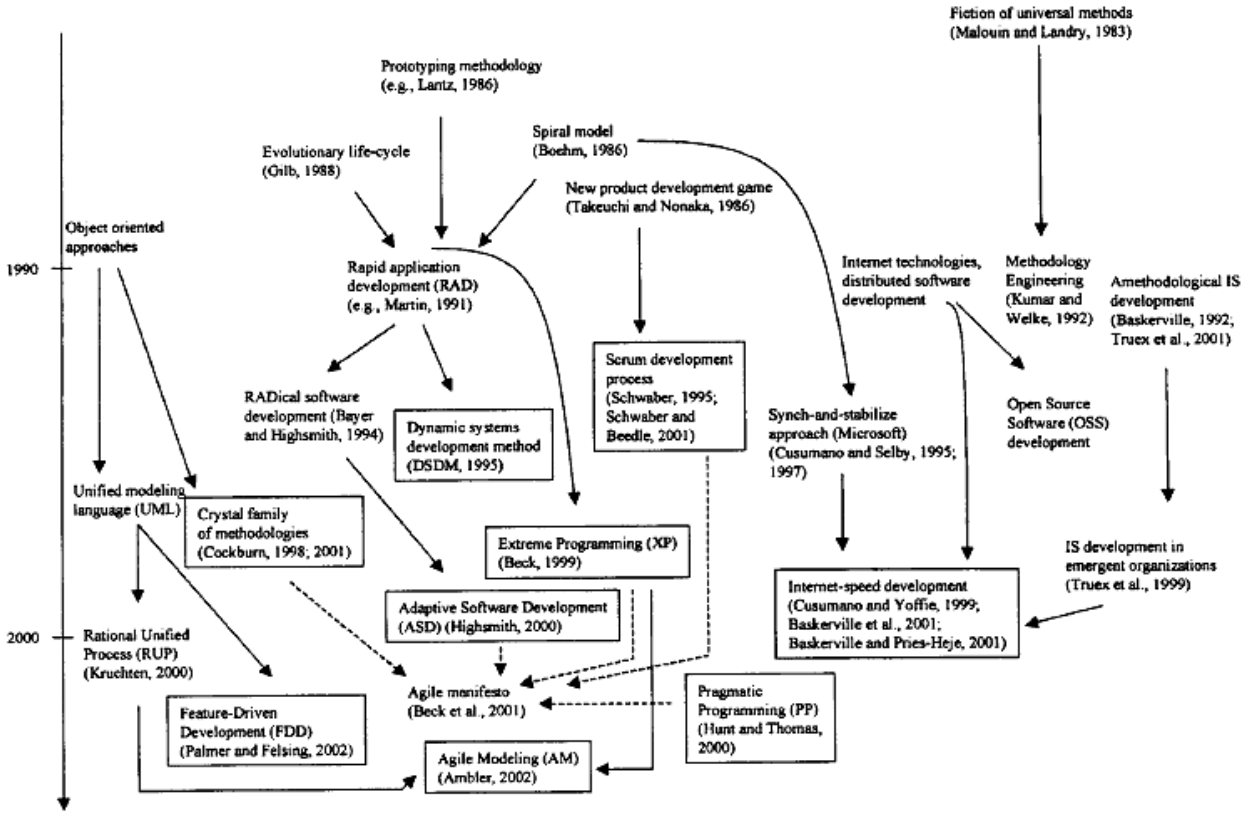


Figure 14 should make it quite clear that agile methods did not suddenly appear in a cataclysmic and violent revolution against traditional methods. In the Kuhnian (1996) sense, the emergence of these iterative and incremental development methodologies arose in incremental steps as reactions to inconsistencies in the prevailing paradigm. Furthermore, agile methods are a collection of ideas and activities from a community of scholars and practitioners; their activities are "pre-paradigmatic" in nature. It is difficult to determine if agile methods represent a paradigm shift, however, the Abrahamsson et al. (2003) illustrate how the formal declaration of

agile methods in the Agile Manifesto was preceded by many subtle methodological challenges which have emerged during what was otherwise a period of "normal science." (Kuhn 1996) Moreover, Kuhn gets to the crux of the agile-as-paradigm question with respect to a community of science and practice by asking: "What do its members share that accounts for the relative fullness of professional communication and the relative unanimity of their professional judgments" (Kuhn 1996:183)? Kuhn's own answer to this is that this community shares a disciplinary matrix of symbols, models and values. In the case of the agile software development community, it is their shared values that are most indicative of paradigm shift as these values have been concretely expressed and attested to by many of those cited in Figure 14.

The real challenge for the agile paradigm is the degree to which the agile movement provides new methods and concepts which do a better job than the methods and concepts of the prevailing paradigm. Although Einsteinian concepts of physics have supplanted Newtonian concepts, it is still easier, in some circumstances, to use what was correct about Newtonian concepts (Kuhn 1996:102). In this sense, Abrahamsson et al. (2003) provide five analytical lenses for analyzing agile methods:

- **Software development life-cycle** – The advantage of referring to a life-cycle for software and systems development exists within the life-cycle's ability to demarcate discrete phases and steps which allow developers to locate their progress. Whether the life-cycle is pursued iteratively or sequentially does not change the usefulness of the life-cycle concept.

- **Project management** – All software projects require some form of management – this is the case for very large projects and small one-or-two person teams. Along with the life-cycle, project management allows for progress to be located and optimized.

- **Abstract principles vs. concrete practices** – To what degree does a method prescribe and proscribe practice? Are tangible steps offered or are values and ideals expected to translate into action?

- **Universally predefined vs. situation appropriate** – Perhaps this is the inverse to the previous analytic lens. It would seem that the more proscriptive and prescriptive a method is, the more universally predefined and applicable it would be. However, one could construe the purpose of a principle as providing a flexible framework for acting in a situationally appropriate manner.

- **Empirical evidence** – how many times has this method been demonstrated to work in a real-world setting?

This analysis, grounded in the literature, is useful for the purpose of assessing agile methods. Rather than focusing on Abrahamsson et al.'s (2003) analysis of each method, their summary assessment of an aggregate of agile methods illuminates what agile methods have and have not done thus far. If agile represents a new paradigm, then the period of "normal science" such that representative methods can be honed and improved has been a relatively short one or has not yet emerged. Wider adoption and use of agile methods should serve to not only iron these methods out, but also expose weaknesses in the methods which will either be resolved within the agile paradigm or serve as the basis for further paradigmatic shift. Table 16 below summarizes Abrahamsson et al.'s (2003) findings.

Table 16 Results of Abrahamsson et al.'s (2003) Comparative Analysis

Perspective	Description of the results	Implications
SDLC	Methods, without rationalization, cover different phases of the lifecycle.	Life-cycle coverage needs to be explained. Life-cycle phases not covered need to be clarified.
Project Management	While most methods appear to cover project management, true support is missing.	Conceptual harmonization is needed. Project management cannot be neglected.
Abstract principles vs. concrete guidance	Abstract principles dominate the literature and developers' minds.	Emphasis should be placed on enabling practitioners to utilize the suggestions made.
Universally predefined vs. situation appropriate	Universal solutions dominate the literature.	More work on how to adopt agile methods in different development situations is needed.
Empirical support	Empirical evidence is limited; most of the research is at conceptual level.	More empirical, situation-specific, experimental work is needed; results need to be publicly available.

The Abrahamsson et al. (2003) comparative analysis makes clear the fact that agile processes are not a panacea. It would seem that a question certainly enters into the mind of a project manager, developer or even customer with respect to agile methods: are agile methods right for my project? Such questions are only answered when the bounds of agile processes are known. The next section addresses the strengths and limitations of agile processes.

2.5.7 Strengths and Limitations of Agile Processes

As is often the case with any technique, process or method, strength can also be a weakness; strengths and weaknesses also underscore fundamental questions regarding suitability.

When strengths and weaknesses are known, a method or technique is more bounded and better understood. On this topic Fowler (2005) has the following to say:

> *One of the open questions about agile methods is where the boundary conditions lie. One of the problems with any new technique is that you aren't really aware of where the boundary conditions are until you cross over them and fail. Agile methods are still too young to see enough action to get a sense of where the boundaries are. This is further compounded by the fact that it's so hard to decide what success and failure mean in software development, as well as too many varying factors to easily pin down the source of problems.*

Generally, Fowler (2005) also suggests that some motivation and proclivity towards agile development, on the part of a development team, on the part of the customer and perhaps inherent within the nature of the project itself, is required in order for agile to be a worthy "experiment." A general acceptance of change, possibly in a radical sense as opposed to an incremental sense, is required in order for any impacting benefits of agile to be evident (Fowler 2005; Orlikowski 1992).

Turk et al. (2002) offer insight into the limitations of agile processes. Limitations are best understood when the original premise of a technique or method is known and explicit. The assumptions of agile processes are stated thusly (Turk et al. 2002):

- Customers are co-located with developers and readily accessible
- Developers are co-located and readily accessible
- Documentation is not essential
- Software requirements will evolve
- Dynamically adaptive processes will create better software
- Developers have the experience and professionalism required to define an adapt their processes
- Timely and frequent product delivery is a primary success factor
- Quality assurance is possible through frequent informal reviews and testing
- Reusability and generalizability are ancillary goals of application and domain-specific software
- The cost of change is fairly constant
- Software can be developed incrementally
- Design does not need to change as change is addressed in refactoring

While the advent of the World Wide Web and popularization of the Internet offers some rationale behind these assumptions, it should be immediately obvious that these assumptions and conditions cannot be met in all circumstances. This would suggest that agile methods are not suited for all project types and all products. In light of these assumptions, Turk et al. (2002) outline some limitations of agile processes and suggest that agile processes are not suitable for all projects (Table 17).

Table 17 Limitations of Agile Methods (Turk et al. 2002)

Limitation	Description
Limited support for distributed development environments	The assumption that customers and other developers are readily accessible for face-to-face meetings is challenged by increasingly distributed development environments. Offshoring to India is one example of this. Developing web-oriented applications for a customer in another location is yet another example of this. However, the same technologies that allow for distributed development can also provide opportunities for rich interaction using information and communication technology as a substitute for face-to-face communication. The work of Lee (1994) and Ngwenyama and Lee (1997) support this.
Limited support for outsourcing/subcontracting	If an agile project is light on specification and light on documentation, the basis for agreements related to subcontracting is sparse. As the manifesto stresses relationships over contracts – this would open up many legal issues which subsume well-intended values.
Limited support for reusable artifacts	With a focus on rapid application development and shorter development cycles, the high-level, long-range planning often required to create appropriate re-usable modules may be lost. A seasoned developer may not have trouble "seeing" where reusability is worthwhile, but an inexperienced developer might not see "the forest for the trees."
Limited support for large teams	A large degree of specification and formalization allows larger teams to work together with the brevity of communication required for coordination and control. With a large project/team, a point of diminishing returns is inevitable when face-to-face and informal communications are the norm.
Limited support for developing high levels of quality assurance and safety	When software safeguards against direct injury to humans or excessive financial loss are required, the informal quality-control mechanisms of agile processes have yet to be proven. While the test-first orientation, early/frequent iterations of working code, and pair-programming component of some agile methods can be effective towards quality control, these approaches lack a formal specification which can be tied to

Limitation	Description
	legal consequence. This problem can be addressed, but some applications are better suited to process and plan orientation.
Limited support for developing large, complex software	While the Larman and Basili (2003) history of Iterative and Incremental Development (IDD) provides evidence to the contrary, there is a category of complex software which does not lend itself well to code refactoring as the principle component of architectural change management. Large and complex projects have so many structural interdependencies that laissez-faire and agile change management techniques could be disastrous.

This assessment of the strengths and limitations of agile methods informs this question: under what circumstances does an adaptive and agile approach make sense as compared to a plan-driven approach? Boehm and Turner (2003, 2004) offer guidance which generalizes the appropriate features of both plan-driven and agile processes by conducting risk analysis. When the risks surrounding each development approach are known, these risks can be extrapolated to the particulars of a given organizational and systems developmental context. Generally, three categories of risk are analyzed such that a balanced decision for or against any given methodological approach is possible (Boehm et al. 2003; Boehm et al. 2004).

Table 18 Risk-based Analysis of Methodologies (Boehm and Turner 2004)

Risk Type	Risks
General Environmental Risks	• Technology uncertainty • Diverse stakeholders • Complex systems
Risks of a plan-driven approach	• Emerging requirements • Constant change • Need for rapid results • Staff skills
Risks of an agile approach	• Scalability • Criticality • Design simplicity • Staff turnover/continuity • Staff skills

The Boehm and Turner (2004) analysis provides a useful framework for undertaking a serious assessment of which method to pursue; in some cases, these approaches are not incommensurate. Another use of the Boehm and Turner (2004) risk framework is to identify which aspects of a project most are sympathetic to a particular software development method. Table 19 approaches the Boehm and Turner (2004) risk analysis categories in a more prescriptive manner.

Table 19 Prescriptions for Method Use (Boehm and Turner 2004)

Use agile methods in case of:	Use plan-driven methods in case of:
• Low criticality • Senior developers • High requirements change • Small number of developers • Culture amenable to chaotic conditions	• High criticality • Junior developers • Low requirements change • Large number of developers • Culture requiring order

As is the case with research methods, systems and software development methods have optimal conditions to which they are most suited. It is also possible to consider, as is the case with Information Systems research, mixed-method approaches where some elements of agile methods and some elements of plan-based methods are combined. A great deal of skill and experience may be required in order to undertake a mixed-method approach successfully, yet the Larman and Basili (2003) historical analysis suggests that mixed-method approaches have been successfully implemented for decades.

Software development methods are not the sole basis for success in software development; their benefits are accrued as governing principles and philosophies in order to ensure progress and success. Methods for systems and software development suggest ways of seeing and not seeing much as is the case with the methods and methodologies that scholars use to research various phenomena. This idea allows for an entirely different assessment of plan-

oriented and agile methodologies. A scholar of ISD or SE might also approach software development methods from a philosophical, epistemological and methodological perspective.

2.5.8 Philosophical Reflections on Agile Methods

Thus far, this dissertation has proposed that the agile methods movement has every appearance of being an example of a Kuhnian paradigmatic shift. We can better understand the role that agile methods serves for practitioners and scholars if we consider the philosophy (ontology, epistemology and methodology) guiding these methods. As "…all research is based on some underlying assumptions about what constitutes valid research and which research methods are appropriate" (Myers et al. 2002:5), there also similar underlying assumptions in the use of agile methods. We can also trace the path to a philosophy for agile methods in epistemology. For example, in Information Systems research, there are qualitative and quantitative approaches to research methods which are guided by an underlying epistemology. Thus, just as methods for research are guided by an underlying epistemology, agile methods are also guided by epistemology. Thus, an epistemology holds underlying "…assumptions about knowledge and how it can be obtained" (Myers et al. 2002:5).

Lee (2004) suggests that a guiding philosophy is comprised of an ontology, epistemology and methodology: An ontology is the "…foundational beliefs about the empirical world" which the members of a community of science or practice share and an epistemology is a "…a broad and high-level outline of the reasoning process by which…" a school of thought or practice "…performs its empirical and logical work" (p.6). It would be hard to argue that software and systems development are NOT both logical and empirical work. Lastly, Lee (2004) refers to a

methodology as "…a more specific manner in which to do empirical and logical work" (p.6). While these concepts and terms were not intended for a community of practice, these concepts provide a means from which the agile methods movement can be understood as a paradigm.

The practitioner literature on agile methods contains numerous value-laden statements concerning agile methods in a philosophical and epistemological manner. This implies that, as is the case with communities of science, the community agile methods practitioners socially construct their knowledge. This socially-constructed knowledge is "…not immutable but under our power as a community… to question, amend, correct and improve" (Lee 2004:7). In the development of agile methods, the community of agile practitioners has undoubtedly questioned, amended, corrected and improved upon engineering and plan-based methods. In doing so, agile practitioners have constructed a new disciplinary matrix. If the values and beliefs held in the community of agile practitioners are part of their disciplinary matrix, it then falls on a community of scholars to develop "…post hoc logical reconstructions of actual logics-in-use" (Lee 2004:7) in order to theorize on the paradigmatic nature of agile methods.

Scholars of software engineering have already begun to theorize on the success of agile methods in search of a philosophical grounding for agile methods (Hazzan et al. 2003; Hazzan et al. 2004a; Hazzan et al. 2004b; Nerur et al. 2007; Nerur et al. 2005; Socha et al. 2006; Tomayko et al. 2004). There is an emerging recognition that the collective work of Argyris and Schön (1974, 1978, 1996) and Schön (1983, 1987), grounded in the philosophies of pragmatism and phenomenology, provides an adequate theoretical and conceptual basis for agility in design as an emergent epistemology of practice (Nerur et al. 2007:79; Rajlich 2006). Thus, understanding agile method success in small shop software development will require a closer examination of

the epistemology guiding the agile phenomenon. This theorizing has only recently begun in earnest and this dissertation is intended to move this process of understanding forward.

Nerur and Balijepally (2007:81) suggest that agile methods represent a new metaphor for design founded in disciplines such as architecture, where flexibility and responsiveness are perhaps more important than optimization. This focus on design as a philosophical and epistemological reference for software and systems development is gaining recognition amongst scholars of Information Systems, Computer Science, and beyond (Hevner et al. 2004; Lee 2007; March et al. 1995; Schön et al. 1996; Simon 1996). Therefore, in order to further theorize on these epistemological underpinnings, and in order to provide a theoretical basis for the Reflective-Agile Learning Model and Method, the work of Argyris and Schön, and Schön warrants further discussion.

2.6 Reflection, Reflective Practice and the Reflective Practitioner

This dissertation has, as among its objectives, a goal to promulgate a reflective practitioner approach to the professional practice of small-team software development. The reflective practitioner approach is outlined by Donald Schön as an epistemology of practice conducive to systematic learning in two books: *The Reflective Practitioner* (1983) and *Educating the Reflective Practitioner* (1987). The overarching intent and purpose in much of Schön's work has been theorizing on learning, reflection and change. Schön, along with Chris Argyris, theorize on the theories of action for *Organizational Learning* to account for how change occurs at the personal, organizational and societal levels (Argyris et al. 1978; Argyris et al. 1996; Argyris et al. 1974). Schön's work on *Organizational Learning* was preceded by work

concerning change and the need for systematic and ongoing learning for adaptation (Schön 1967; Schön 1973). Others have theorized on organizational change and *Organizational Learning* (Keen 1981; Senge 1994; Simon 1991).

2.6.1 Antecedent Work and Thinking

Donald Schön characterizes the impetus for *Organizational Learning* in *Beyond the Stable State* (1973): "…our society and all of its institutions are in continuing processes of transformation. We cannot expect new stable states will endure even for our own lifetimes" (p.30). Thus, Schön suggests a need for consistent systems for ongoing learning: "…We invent and develop institutions which are 'learning systems,' that is to say, systems capable of bringing about their own continuing transformation" (p.30). This imperative for continuous *Organizational Learning* is often challenged by dynamic conservatism: the natural resistance a system has to change as change presents a threat to the actors within a system (Schön 1973: 51). Dynamic conservatism is important as it is a disruptive force which confounds change management and learning. Rather than hope to eradicate dynamic conservatism, a learning system accommodates and minimizes the effects of dynamic conservatism:

> *A learning system… must be one in which dynamic conservatism operates at such a level and in such a way as to permit change of state without intolerable threat to the essential functions the system fulfils for the self. Our systems need to maintain their identity, and their ability to support the self-identity of those who belong to them, but they must at the same time be capable of transforming themselves. (Schön 1973: 57)*

Schön presents a model for how innovations can be used in a learning organization which differs from a product-centric and static concept of learning through innovation diffusion (Table 20).

Table 20 Diffusion of Innovation in Traditional and Learning Systems (Schön 1973)

Classical models for the diffusion of innovations	Learning systems' models around the diffusion of innovations
The unit of innovation is a product or technique	The unit of innovation is a functional system
The pattern of innovation is center to periphery	The pattern of diffusion is systems transformation
Relatively fixed, central and hierarchical leadership	Flexible and ad hoc leadership
Signaled as a stable replication of a centralized message	Evolving message from a team of shared ideas
Scope limited by the resources and will from the center; emanates via "spokes" from the center to the periphery	Scope limited by the infrastructural technology to carry the change message; increases in connectivity increase the scope
Feedback from the periphery to the center and back	Feedback operates equally at local and universal levels throughout a system's network

Schön's model of diffusion in a learning system clearly presages the connectivity made possible by the Internet. Thus, the required elements of a learning system, which diffuses the innovations of learning, are networks, flexibility, feedback mechanisms and organizational transformation (Smith 2001). A learning system places importance on individual learning and systemic learning where learning is a social act among members of the learning system: "…A social system learns whenever it acquires new capacity for behavior, and learning may take the form of undirected interaction between systems…" (Schön 1973: 109). Many of the interactions within and between systems would facilitate learning at the periphery (among the actors) rather than from center to periphery and back (Schön 1973: 165). Furthermore, a learning system would inform the center as a result of learning at the periphery; aggregate wisdom is collected from the experiences of actors within the system rather than by axiomatic and rational norms and standards issued from the center. Thus, the center is the facilitator and legitimizer of change and not the tutor (Schön 1973: 166).

Schön's early work on learning systems transitioned naturally into work with Chris Argyris on theories of action for *Organizational Learning*. Important themes which emerge later in the epistemology of *Reflective Practice* are first examined in Schön's work with Argyris (1974, 1978, 1996). A learning system relies on the ability of actors within the system to learn and, subsequently, transmit this learning within the system. Thus, in taking action, actors within a learning system work from mental maps also called "tacit knowing" (Polanyi 1983: 9). Argyris and Schön's work focuses specifically on the tacit knowledge of held by professional practitioners and how this knowledge results in and informs action. While a practitioner offers espoused theories (informed by professional training, social environment and other indoctrinations) explaining their action, Argyris and Schön argue that action is mostly informed by a tacit *Theory-in-use* (Argyris et al. 1974: 29). These are the two "theories of action" which influence *Organizational Learning*.

Implicit and tacit theories-in-use are influenced by the practitioner's beliefs and values concerning a set of governing variables which drive their action planning and action taking (Argyris et al. 1974). Differentiating between *Espoused Theory* and *Theory-in-use* is vital not only to understand and affect a learning system, but to also understand how professionals think in action. Argyris and Schön (1974, 1978, and 1996) also propose that a practitioner's theory of action influences mechanisms for *Organizational Learning* called *Double-Loop* and *Single-Loop Learning*. In detecting and correcting errors evident in the consequences of action, a practitioner and/or organization will either choose to reconsider and question their beliefs and attitudes concerning governing variables or they will continue to seek alternative action-strategies without any changes in attitudes and beliefs concerning their governing variables. To question attitudes and beliefs concerning governing variables as the result of the consequences of action-taking is

to engage in *Double-Loop Learning*. A failure to question governing variables is to engage in *Single-Loop Learning*; learning which does not result in a learning system (Argyris et al. 1978; Argyris et al. 1996).

This antecedent thinking and theorizing occurred in the 1960s and 1970s; by the 1980s, Schön had developed new thinking on learning systems which centered on the individual professional practitioner and professional practice. As professional practice is where the "rubber meets the road," a focus on professionals and professional institutions is a natural progression in theorizing on learning systems. Schön's work in the 1980s on professional practice, collected primarily in *The Reflective Practitioner* (1983) and *Educating the Reflective Practitioner* (1987), serves as the cornerstone to Schön's overall program on learning and change. In these books, Schön introduces an epistemology of *Reflective Practice* and its importance in reflective action and effective professional practice. The following section outlines Schön's epistemology of reflective professional practice.

2.6.2 Reflecting on the Reflective Practitioner

In *The Reflective Practitioner* (1983) and *Educating the Reflective Practitioner* (1987), Schön outlines a reflective epistemology of practice in contrast and opposition to the positivist epistemology of practice, *Technical Rationality*, and underscores the importance of reflective practice for the development of learning systems in the professions and throughout society. As an epistemology is an aspect of philosophy (usually philosophy of science), a quick review of epistemology is warranted. Whereas an ontology comprises the foundational beliefs about the empirical "real" world amongst a community of practice or science, an epistemology is "…a

broad and high-level outline of the reasoning processes by which a school of thought performs its empirical and logical work" (Lee 2004: 6). From this definition we can determine that an epistemology assists in how we reason about truth and how truth is justified through reasoning.

2.6.2.1 Technical Rationality: The Positivist Epistemology of Professional Practice

Technical Rationality is Schön's characterization of the positivist epistemology of practice. *Technical Rationality* holds that "…professional activity consists in instrumental problem solving made rigorous by the application of scientific theory and technique" (Schön 1983: 21). *Technical Rationality* presents a system for regarding the truth and verity of professional knowledge and professional practitioners' knowledge base. In *Technical Rationality*, a professional's knowledge base is specialized, firmly bounded, scientific and standardized. This knowledge base has three principle components:

- **Basic Science** - An underlying discipline or basic science component upon which practice rests or from which it is developed.
- **Applied Science** - An applied science of "engineering" component form which many of the day-to-day diagnostic procedures and problem solutions are derived.
- **Body of Knowledge** - A skills and attitudinal component that concerns the actual performance of services to the client, using the underlying basic and applied knowledge.

Thus, the epistemological view of *Technical Rationality* is hierarchically embedded in the institutional context of professional life; it is embedded in the distinctions between research and practice, in professional education and in attitudes towards learning and change. From the perspective of *Technical Rationality*, research and practice are separate activities where researchers provide basic and applied scientific knowledge from which practitioners derive their

techniques and knowledge for practice. In turn, practitioners furnish a problem environment from which researchers learn. Any art or craft which might be a functional part of professional practice is omitted or controlled in favor of knowledge developed from scientific and rigorous hypothetico-deductive hypothesis testing.

Schön critiques and challenges *Technical Rationality* as it focuses on professional practice as an endeavor of problem-solving rather than problem-setting. Schön presents this distinction as being problematic: "…in real-world practice, problems do not present themselves to the practitioner as givens. They must be constructed from the materials of problematic situations which are puzzling, troubling and uncertain" (Schön 1983: 40). According to Schön, professionals "…are coming to recognize that although problem setting is a necessary condition for technical problem solving, it is not itself a technical problem" (p. 40). Rather, "…problem setting is a process in which, interactively; we name the things to which we will attend and frame the context in which we will attend to them" (p.40). Thus, whereas "…*Technical Rationality* depends on agreements about ends… a conflict of ends cannot be resolved by the use of techniques derived from applied research…" (p.41). In *Reflective Practice,* Schön favors the non-technical processes related to naming, framing, seeing-as and metaphor which govern problem setting. Problem setting is a necessary pre-cursor to a positivist learning system of *Technical Rationality*, "…it is through the non-technical process of framing the problematic situation that we may organize and clarify both the ends to be achieved and the possible means for achieving them" (p.41).

Schön describes problem-setting as fraught with uncertainty, uniqueness, instability and value conflicts of the sort that *Technical Rationality* is ill-equipped to handle. Thus, an alternative epistemology which allows for problem setting is required in order to establish a

learning system. Problems cannot be set from a known constellation of scientifically vetted truth; problems are dynamic and require a dynamic response. Thus, if the ability to define a problem's setting is somewhat of an art, then any artistic and creative approach not codified in the central knowledge of the positivist tradition would be considered non-rigorous: this becomes another means of framing the rigor vs. relevance debate. Schön describes the situation thusly:

> *In the varied topography of professional practice, there is a high, hard ground where practitioners can make effective use of research-based theory and technique, and there is a swampy lowland where situations are confusing 'messes' incapable of technical solution. The difficulty is that the problems of the high ground, however great their technical interest, are often relatively unimportant to clients of the larger society, while in the swamp are the problems of greatest human concern. Shall the practitioner stay on the high, hard ground where he can practice rigorously, as he understands rigor, but where he is constrained to deal with problems of relatively little social importance? Or shall he descend to the swamp where he can engage the most important and challenging problems if he is willing to forsake technical rigor? (Schön 1983)*

A rather tepid reaction to this dilemma is to mold the situation of practice to fit the constraints of technical knowledge: models are changed to suit the data; systems are built irrespective of their consequence; failures are cast off as "affect" and thus marginalized. A variation on this behavior is to mold the situation of practice around available and sanctioned techniques rather than develop techniques which reflect the truth of the problem setting. Thus, Schön summarizes the degree to which *Technical Rationality* (the positivist epistemology of professional practice) raises a dilemma:

> *It seems clear, however, that the dilemma which afflicts the professions hinges not on science per se but on the Positivist view of science. From this perspective, we tend to see science, after the fact, as a body of established propositions derived from research. When we recognize their limited utility in practice, we experience the dilemma of rigor or relevance. (Schön 1983: 49)*

In response, Donald Schön offers an alternative epistemology of professional practice which embodies the implicit, tacit, artistic and intuitive processes practitioners bring to problem setting.

2.6.2.2 Elements of Reflective Practice

Schön's epistemology of *Reflective Practice* focuses on the situatedness of action and the knowing-in-action which governs the decision-making and action-taking of most professionals in their daily work. This knowing-in-action is mostly tacit and implicit in patterns of action and evident when practitioners "think on their feet." Therefore, the practitioner's mental maps, his or her theories-in-use, are the product of *Reflection-in-action*, where "tacit knowing" is "in" the action (Polanyi 1983; Schön 1983) . Thus, a sketch of Schön's epistemology begins to take shape in the following terms and concepts:

- **Tacit knowing** – *These are actions, recognitions, and judgments which we know how to carry out spontaneously; we do not have to think about them prior to or during their performance.* (Schön 1983: 54)

- **Knowing-in-action** – *Our knowing is ordinarily tacit, implicit in our patterns of action and in our feel for the stuff with which we are dealing… our knowing is in action.* (Schön 1983: 49)

- **Knowing-in-practice**: … *[A practitioner] develops a repertoire of expectations, images and techniques. He learns what to look for and how to respond to what he finds.* (Schön 1983: 60)

- **Overlearning**: *As long as practice is stable, in the sense that it brings the same types of cases, [the practitioner] becomes less and less subject to surprise… Knowing-in-practice tends to become increasingly tacit, spontaneous and automatic, thereby conferring the benefits of specialization*. (Schön 1983: 60)

  …As practice becomes more repetitive and routine, and as knowing-in-practice becomes increasingly tacit and spontaneous, the practitioner may miss important opportunities to think about what he is doing. (Schön 1983: 61)

- **Reflection-in-practice** – *A practitioner's reflection can serve as a corrective to over-learning. Through reflection, he can surface and criticize the tacit understandings that have grown up around the repetitive experiences of a specialized practice, and can make new sense of the situations of uncertainty or uniqueness…* (Schön 1983: 61)

- **Reflecting-in-action** – *If common-sense recognizes knowing-in-action, it also recognizes that we sometimes think about what we are doing. Phrases like 'thinking on your feet,' 'keeping your wits about you,' and 'learning by doing' suggest not only that we can think about doing but that we can think about doing something while doing it. Some of the most interesting examples of this process occur in the midst of a performance.* (Schön 1983: 54)

- **Reflecting-on-action** – *Practitioners do reflect on their knowing-in-practice. Sometimes, in the relative tranquility of a postmortem, they think back on a project they have undertaken, a situation they have lived through, and they explore the understandings they have brought to their handling of the case.* (Schön 1983: 61)

- **Naming-and-framing –** *When phenomenon at hand eludes the ordinary categories of knowledge-in-practice, presenting itself as unique or unstable, the practitioner may surface and criticize his initial understanding of the phenomenon, construct a new description of it, and test the new description by an on-the-spot experiment. Sometimes he arrives at a new theory of the phenomenon by articulating a feeling he has about it. When he finds himself stuck in a problematic situation which he cannot readily convert to a manageable problem, he may construct a new way of setting the problem – a new frame which … he tries to impose on the situation.* (Schön 1983: 63)

- **Frame-experiment** – Practitioners ...*encounter a problematic situation whose reality they must construct. As they frame the problem of the situation, they determine the features to which they will attend, the order they will attempt to impose on the situation, the directions in which they will try to change it. In this process, they identify both the ends to be sought and the means to be employed. In the ensuing inquiry, action on the situation is integral with deciding, and problem-solving is a part of the larger experiment in problem setting.* (Schön 1983: 165)

- **Repertoire** –*…examples, images, understandings and actions… (which) includes the whole of his experience insofar as it is accessible to him for understanding and action.* (Schön 1983: 138)

- **Seeing-as and generative metaphor** – *When a practitioner makes sense of a situation he perceives to be unique, he sees it as something already present in his repertoire*. (Schön 1983: 138)

  Once a new problem is seen to be analogous to a problem previously solved…, then metaphor is being used to generate new conceptions from experience. *When the two things seen as similar are initially very different from one another, falling into what are usually considered different domains of experience, then seeing-as takes a form [called] 'generative metaphor.'* (Schön 1983: 183)

  …Seeing A as B where A and B are initially perceived, named and understood as very different things – so different that it would ordinarily pass as a mistake to describe

> *one as another. It is the restructuring of the perception of the phenomena A and B which enables us to call 'metaphor' what we might have otherwise called 'mistake'.* (Schön 1983: 185)

- **Theory-in-action** – A learning system uses reflection to reflection on the governing variables within a problem setting.

These elements of Schön's epistemology fit together in the following summary: A practitioner uses tacit knowledge and knowing-in-action when problem setting; generally, a practitioner's repertoire will cover day-to-day situations and extend expertise. However, novel situations call for the use of reflection-in-practice (reflection in and on action) in order to conduct frame experiments whereby novel situations are comprehended and added to repertoire; By approaching a new situation using generative metaphor, and by using reflection-in-practice, a reflective practitioner is able to contribute to a learning system.

2.6.3 Acceptance and Critique of Reflective-Practice

Schön's epistemology of *Reflective Practice* has received considerable attention and traction within several disciplines: *Education* (Bryant et al. 1997; Freidus 2002; Garlan et al. 1997; Hazzan 2002; Hazzan et al. 2004c; Mathiassen et al. 2002; Smith 1994), *Management* (Daudelin 1996; Heiskanen 1995; Jones 2002; Lalle 2003; Rosier 2002; Ulrich 2000; Vince 2002) , *Nursing* (Jarvis 1992), and *Industrial and Engineering training* (Cheetham et al. 1996; Garlan et al. 1997; Kuhn 1998), to name a few. Several researchers and scholars in areas of design have also recognized the utility of *Reflective Practice* (Anderson et al. 1993; Atwood et al. 2002; Dorst et al. 1995; Fallman 2003; Hazzan et al. 2004a; Hill et al. 2002; Louridas et al. 2000; Stempfle et al. 2002; Vessey 2006; Wakkary 2005). In nearly all cases, the popularity of *Reflective Practice* is in mature disciplines with a strong professional identity and a focus on

education. *Reflective Practice* has resonated with areas of professional education largely due to Schön's 1987 follow up book, *Educating the Reflective Practitioner*, which focuses on techniques for implementing *Reflective Practice* in educational settings. This focus on "pedagogy" is among the bases for many of the criticisms of *Reflective Practice*.

While there are many criticisms of *Reflective Practice*, Smith (2001) summarizes the three major criticisms:

- **Distinguishing Reflection "in" and "on" practice** – Some feel that the possible scope for reflective practice is extremely limited when time is tight. Schön (1983) does address this on page 275 where he asserts that reflection, regardless of time allotted, need not inhibit action: a tennis player would learn to reflect on moments past and upcoming moments when planning a shot.

- **Relating reflective practice to praxis** – Some complain that reflective practice it not prescriptive (and/or proscriptive) enough to have meaningful impact on praxis. Such a critique comes from a reader who may not have been able to suspend disbelief long enough (presumably due to positivist indoctrination) to realize that "…The ability to draw upon a repertoire of metaphors and images that allow for different ways of framing a situation is clearly important to creative practice and is a crucial insight" (Smith 1994: 142-145).

- **Lack of a rigorous analysis of the epistemology** – Bryant et al. (1997) summarize thusly: "…what we do not find in Schön is a reflection by him on his own textual practice in giving some kind of account of that he does of *Reflection-in-action* and the reflective practicum… He does not interrogate his own method."

2.6.4 Situating the Utility of Reflective Action

Donald Schön's epistemology of *Reflective Practice* significantly influenced professional training and education for teachers, nurses and engineers (Bryant et al. 1997; Hazzan et al. 2004a; Jarvis 1992; Tomayko et al. 2004). In the field of education, there is a sense that *Reflective Practice* has become a part of that discipline's canon; it demonstrates frequent utility

for educators in a variety of professions (Bryant et al. 1997). However, it is important to note that in most uses, practitioners are encouraged to "apply" *Reflective Practice,* in a prescriptive manner, to their own situations and experiences (Smith 1994). Schön seems to have intended *Reflective Practice* not as a method to be applied but as an epistemology, or methodology, which provides a framework for situated and reflective action.

It is equally important to keep in mind that *Reflective Practice* is very connected to Argyris and Schön's work on *Organizational Learning*, learning organizations and learning systems. It is argued that it is Donald Schön's work on learning systems that still provides the most thorough theoretical treatment on the matter (Smith 2001). *Reflective Practice* is a means to harness individual practitioner repertoire which contributes to a learning system: *Reflective Practice* is not a recipe or formula to be applied. In *Educating the Reflective Practitioner* (1987), Schön does provide some procedural and methodical hints for adopting *Reflective Practice*.

2.7 Synthesizing the Literature: Emerging Themes

The reviewed literatures reveal certain patterns which can serve as a basis to justify the legitimacy of the research questions: the professional practice of software development is still nascent and evolving; small teams do have unique needs and require software development methods suited to their size; there are multiple approaches available for assessing software development methodologies; agile methods represent a paradigmatic break from software engineering's past and are well-suited to small-team software development. Among these themes, the critical element is Schön's epistemology of *Reflective Practice*. The literature

review concludes with a summarization of the importance *Reflective Practice* with respect to the research questions and objectives of this dissertation.

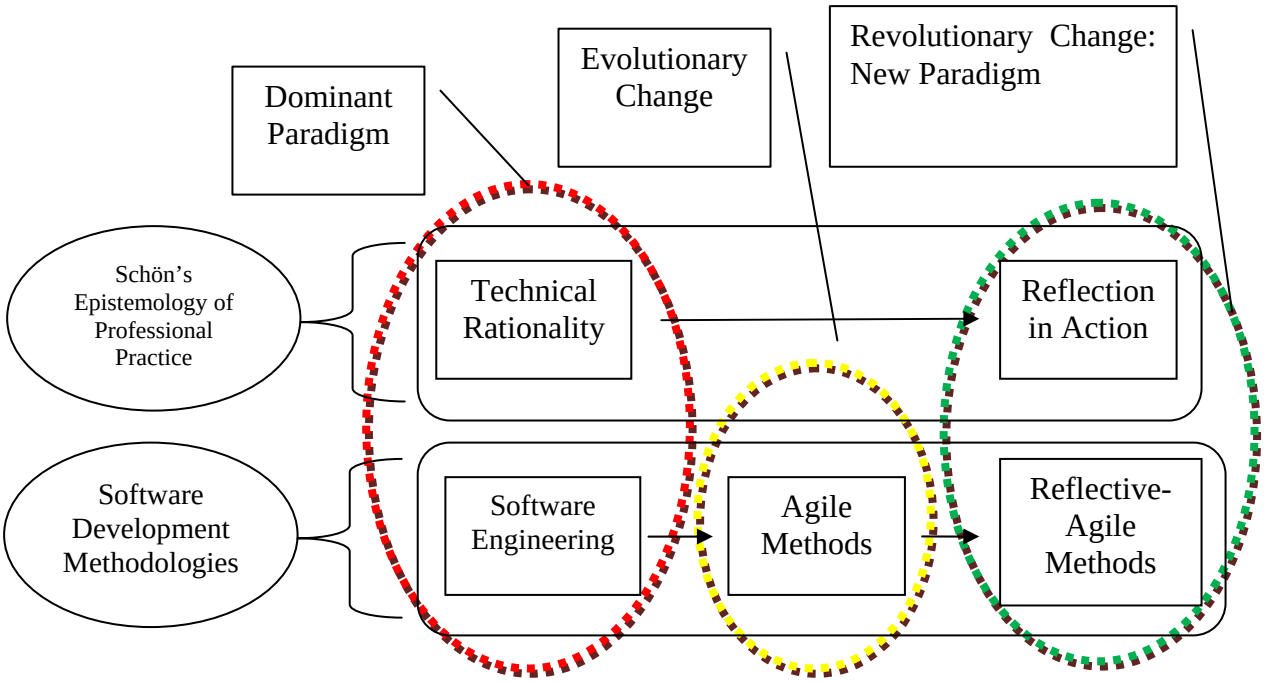
2.7.1 Understanding Agility through Reflective Action

It is evident in the literature that the agile methods movement came to a head as a response to disconnect between the experiences of practice and the guidance of the Software Engineering paradigm and metaphor. Throughout the 1990s and up to the present, rapid change, wrought by advances in information technologies and user expectations arising out of these advances, have created discord between daily experience and the guidance of traditional software development methods. Thus, parallels between *Technical Rationality* and *Reflective Practice* and between Software Engineering and agile development methods are palpable. Each shows how the experience of practice questions generalized knowledge in the dominant paradigm. Whereas a natural response to this dilemma would be to update theory to reflect reality, Kuhn suggests that such transitions are not so simple; the inertia the Positivist epistemology of *Technical Rationality* is hard to overcome. This is what Schön (1973) described as *Dynamic Conservatism.*

Kuhn (1996) also tells us that it is often a revolution which creates the total change necessary to bring theory into alignment with a critically and fundamentally changed environment of practice. In developing RALMM in a Dialogical AR Partnership, this dissertation theorizes on how the software development has been permanently changed by the Internet and the World Wide Web and how this affects small software development shops. Figure 15 below suggests the relationship between *Technical Rationality* and software

engineering. Figure 15 also suggests a connection between Schön's *Reflective Practice* and the agile methods movement and illustrates the frame conflict between engineering and artisanship in agile methods. If agile methods are an evolutionary step, we can use *Reflective Practice* as a theoretical lens to suggest new models for IT practice.

Figure 15 Progression Towards a Craftsmanship Model of IT Practice



2.7.3 Considering the Artisan Frame

One possible professional frame and metaphor for agile methods is that of the artisan. In his epistemology of *Reflective Practice*, Schön (1983) characterizes the reflective practitioner as one who draws as much as, if not more, from art as he or she does from science. Whereas the concept and viability of artisan as skillful master of a trade has been supplanted by the science,

engineering, and Taylorism (Chau et al. 2003; Melnik et al. 2004), Schön proposes that the epistemology of *Reflective Practice* reconsider art and craft:

> *Among philosophers of science, no one wants any longer to be called a Positivist, and there is a rebirth of interest in the ancient topics of craft, artistry and myth – topics whose fate Positivism once claimed to have sealed. It seems clear, however, that the dilemma which afflicts the professions hinges not on science per se but on the Positivist view of science. From this perspective, we tend to see science, after the fact, as a body of established propositions derived from research. When we recognize their limited utility in practice, we experience the dilemma of rigor and relevance… Let us search, instead, for an epistemology of practice implicit in the artistic, intuitive processes which some practitioners do bring to situations of uncertainty, instability, uniqueness, and value conflict.(Schön 1983:49)*

It may be that newer forms of communication and organizing, made possible by information technology, have modified frames and metaphors for the professional practice of software development towards reflective and artful craft. Thus, the question may be asked: can the agile methods movement promote new a understanding of software development, from the frame and metaphor of the artisan rather than the engineer? This question is a prime motivator of this dissertation. The development of the Reflective-Agile Learning Model and Method will, among other things, provide empirical evidence to support this postulation.

While the traditional concept of the artisan is that of the skilled manual worker, what about the skilled knowledge worker? As artisans were the dominant producers of goods before the Industrial Revolution, what of the skilled knowledge worker in the Information Age? An artisan creates artifacts based on skills mixed with intuitive and creative thinking: so too does the knowledge worker. Thus, the focus on *Reflective Practice* as the theoretical stimulus in this research may open up argument that tomorrow's skilled information technology worker may be understood as an artisan and a professional. Thus, the professions are so tied up in the dogma of

Positivism that very concept of "the professional" may hinder advancement in the professional practice of software development.

2.7.4 Using Reflective-Agile as a Generative Metaphor for Small Teams

In appropriating *Reflective Practice* as a theoretical lens, this research relies heavily on the concept of metaphor as a structuring force of design. Metaphors are often used as a device to understand the unknown from the perspective of known frameworks. Metaphor is also a communications technique which is helpful in managing change. In this sense, metaphors become "… central to the task of accounting for our perspectives on the world: how we think about things, makes sense of reality, and set the problems we later try to solve" (Schön 1979). We use metaphors to "frame" phenomena by classifying them where these frames cast a new perspective on phenomena. Metaphors are used as a short-hand for understanding and also to cast new light on a subject; revealing according to the metaphor and hiding what the metaphor does not account for[10]. Metaphor is also abstraction: the details of new phenomena are often subsumed by the already-accepted lessons of *a priori* experience which illuminate the new phenomena and anchor them to an existing frame. Carrying over an existing frame to another domain of experience is often used to "tame" this new domain and subsume it into an existing philosophical, ontological, epistemological and methodological tradition (Barrett et al. 2001).

[10]Consider the story of the drunkard and the lamppost: "A policeman saw a drunk searching for something under a lamppost. 'What have you lost, my friend?' the policeman asked. 'My keys', said the drunk. The policeman then helped the drunk look for his keys and finally asked him: 'Where exactly did you drop them?' 'Over there', responded the drunk, pointing toward the dark street. The policeman then asked: 'Why are you looking here?' The drunk immediately replied: 'Because the light is so much brighter here.'

Software development began as the handiwork and intellectual craft of a select and gifted few practitioners privy to the esoteric computing systems available at the time. The importance of computing and data processing demanded more computer programming for which an artful and craft-oriented practice of computer programming was not ready. *Technical Rationality* had tamed problems in so many other domains and disciplines that the advent of computer science and software engineering are understandable responses. In order to "clean up" the mess that software development had become, engineering was proffered as a generative metaphor: a way of "seeing" software development in a new light: the applied science of engineering. According to the Accreditation Board for Engineering and Technology[11], engineering is

> *…the creative application of scientific principles to design or develop structures, machines, apparatus, or manufacturing processes, or works utilizing them singly or in combination; or to construct or operate the same with full cognizance of their design; or to forecast their behavior under specific operating conditions; all as respects an intended function, economics of operation and safety to life and property.*

These are concepts and notions bounded by rationality; thus, we can consider engineering as an instantiation of *Technical Rationality*.

Engineering has a central mission of problem-solving. Certainly problem-solving is a goal-seeking activity which is congruent to the purposes of computers and computer programs: we want that these artifacts to work on our behalf. It is less clear that the practice of software development is entirely bound by the rational problem-solving of engineering. The actions of an effective software development practitioner may go beyond those prescribed by the engineering model. Once all of the problems of the software developer are cast in the generative metaphor of engineering, the engineering becomes the only legitimate and validating measure of success.

[11] http://www.abet.org

Thus, Schön suggests that a generative metaphor casts a "spell" under which certain things become "good" and others "bad." (Schön 1979) Software engineering was "generated" out of the engineering metaphor and used to create the discipline of software engineering anew.

This dissertation, in using *Reflective Practice* as a theoretical lens, is also concerned with the metaphors used to understand the professional practice of small-team software development in the small-shop environment. Agility is an obvious metaphor guiding agile methods, and reflection is an obvious metaphor guiding Schön's epistemology of *Reflective Practice*. Thus, the conjunctive, Reflective-Agile, is used as a generative metaphor as the paradigmatic implications combining *Reflective Practice* and agile methods to explore a learning system in small-team and small-shop software development.

CHAPTER 3 Description of the Baseline Reflective-Agile Method

Dialogical AR is an interpretive research approach used in this dissertation to design and develop a design science artifact starting from a baseline *Reflective-Agile* method for small-team software development in the small-shop environment. An action research approach promises to "…produce highly relevant research results, because it is grounded in practical action, aimed at solving an immediate problem situation while carefully informing theory" (Baskerville 1999). Further, action research well-suited for researching IS phenomena as action research is "… one of the few valid research approaches that researchers can legitimately employ to study the effects of specific alterations in system development methodologies" (Baskerville et al. 1996).

The designed artifact developed in this research, the Reflective-Agile Learning Model and Method, is a synthesis Extreme Programming (XP) and Don Schön's (1983, 1987) epistemology of *Reflective Practice*. XP is used as a baseline agile method which is iteratively modified to include *Reflective Practice.* The Dialogical AR cycle provides an empirical basis to specifying learning for the both the practitioner and researcher in the research outcomes and provides feedback to the theoretical foundations of the interventions.

The remaining sections of this chapter are presented as follows. First, criteria for the selection of XP as the baseline agile method are examined. Next, each of the major elements, XP and *Reflective Practice,* of the baseline Reflective-Agile method are discussed, followed by a description of the final synthesized approach. Next, the theoretical bases of the proposed

interventions are discussed. Lastly, the baseline Reflective-Agile learning approach for small-team and small-shop software development is outlined and described.

3.1 Criteria for the Selection of Extreme Programming

In considering an agile method for small-team software development, a selection based on team size is logical given the proposed Dialogical AR setting. From the list in Table 15 in the previous chapter, the Extreme Programming (XP), Team Software Process (TSP), Personal Software Process (PSP) and Cleanroom agile software development methods are best suited as they specifically and directly address small teams and individuals. Crystal, DSDM and RUP hold promise yet lack an explicit emphasis on small teams and individuals based on the literature (Boehm et al. 2004). With the selection of any agile method, it is important to consider the degree to which the individual programmer is addressed as a smaller team will focus at individual level.

Beyond a focus on the individual, an agile software development method for small teams should also widely address the SDLC. Whereas Extreme Programming, PSP and Cleanroom each address the individual and/or the small team, XP also covers a wide range of the SDLC (Boehm et al. 2004). Thus, candidate methodologies for a baseline Reflective-Agile method are now narrowed to XP, PSP and Cleanroom. Of these choices, XP most applicably fits due to its organizational emphasis and breadth of life-cycle coverage. According to Boehm and Turner (2004) the most agile method is that which places the fewest constraints on method adoption. However, as methods guide activity and practice, a method necessarily constrains through its

guidance and in its use. Boehm and Turner's (2004) definition of an agile method as a method most free from constraints would favor Scrum, Adaptive Software Development and Lean Development; however, the least-constrained of the small-team agile methods is Extreme Programming. By way of this logic, it follows that Extreme Programming meets several criteria making the method a valid candidate as a baseline for the Reflective-Agile method for small-team software development:

- XP has an emphasis on individuals and small teams
- XP addresses a wide range of the SDLC
- XP is not overly constraining and, thus, according to Boehm and Turner (2004), XP is agile
- XP is widely adopted and has been the subject of extensive practitioner use
- XP has been well critiqued and evaluated by way of extensive practitioner use
- XP has been proven as an appropriate method for small teams and individuals

Thus, this Extreme Programming was selected as a baseline method for the Reflective-Agile method. Also, this research secondarily considers TSP and PSP if there are problems with the use of XP. While these methods are less agile, they are rigorous and directly address the concerns of small-team software development. At best, TSP and PSP would provide critical balance to XP in much the same way that *Reflective Practice* will augment and enhance XP in the development of the Reflective-Agile method. The following section further explores and examines the XP methods.

3.2 Examining the Baseline Reflective-Agile Method

The baseline Reflective-Agile method for small-team software development involves the introduction of the epistemology of *Reflective Practice* to Extreme Programming. Thus, at its core, the baseline Reflective-Agile method is Extreme Programming. The baseline Reflective-Agile method appropriates modifications and activities suggested by Hazzan (2002, 2004) and Hazzan and Tomayko (2004) to introduce the reflective practitioner approach to agile software development. The use of Dialogical AR will allow for iterative and empirical evaluation the interventions related to *Reflective Practice* and will guide appropriate changes (if any) to XP.

3.2.1 The Agile Element: Extreme Programming Examined

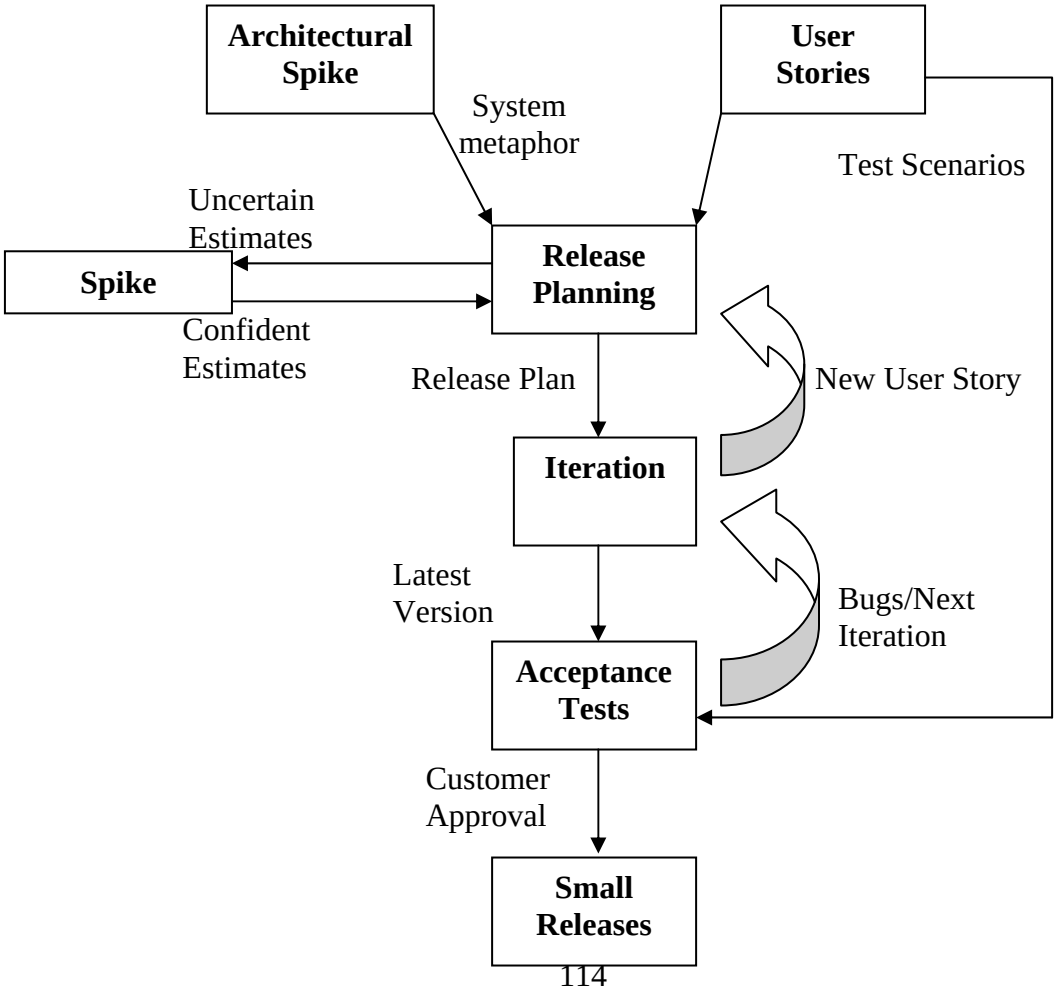
XP has enjoyed a high profile among agile methods and is among the most controversial of agile methods (Boehm 2002a). XP is actually based on 12 practices, Table 21 examines these practices.

Table 21 Extreme Programming Practices (Beck 1999; Wake 2002)

XP Practice	Practice Description
Planning	
User stories	As requirements often change over time and as a project grows, user stories are mnemonics which capture short phrases which the customer gives to express intentions. The developers also record a short phrase and reconcile this with the customer.
Planning Game	At the beginning of each iteration, the customer and the developer choose features required for the next iteration.
Short Cycles	Delivers working software every two weeks.
Iteration Plan	Developers set a budget for what can be accomplished during an iteration and obtain agreement with the customer partner.
Release Plan	A calendar/map of upcoming iterations towards completion and acceptance.
Designing	
Simple Design	Design follows simple user stories created for a given iteration.
Use a system metaphor	This creates an overall sense of what the software is in abstract terms which allows for conceptual connections to be quickly established. This has a lot to do with naming and framing.

XP Practice	Practice Description
Refactor often	Systems and software are living things which are subject to change. Reuse and modularity are principles which at times are at odds with the evolution necessary to keep up with a dynamic system. When developers remove redundancy, eliminate unused functionality, and rejuvenate obsolete designs, they are refactoring
Coding	
Customer Team Member	The customer is an integral part of the team
Pair Programming	Production code is created by two programmers working at the same work station
Continuous Integration	As cycles are short, new versions and fixes should be integrated into the working software quickly.
Testing	
Test-Driven Development	All production code is written to previous and new unit tests ensuring that rapid feature additions does not break existing code

Figure 16 Elements of Extreme Programming (Wells 2000)



The general steps and flow of an XP development project are seen in Figure 16 as modeled by Wells (2000). XP has a clear emphasis and applicability to small-team software development. Furthermore, XP addresses a significant portion of the Software Development Life Cycle within its methodological steps.

The general elements of the XP method in Figure 16 correspond with the XP practices explicated in Table 21. Each phase of the method describes discrete actions and specifies meaning for these actions. The following sections provide a step-wise narrative describing the general flow of the XP method as specified in Figure 16.

3.2.1.1 Specifying the System

The general flow of the XP method begins with *User Stories*, which are written by the customer in order to describe the things that the customer system needs to do for them. Generally, these are in the form of a few sentences which are free from jargon and technical language. Thus, *User Stories* specify the requirements which the system/software must satisfy.

As *User Stories* are collected, *Architectural Spikes* are used to demonstrate solutions to technical problems based on the *User Stories* and for prototyping. *Architectural Spikes* are prototypes which may not end up in a final project, but which test feasibility and the technical capability to achieve the system requirements as specified in the *User Stories*. This activity reduces the risks associated with technical hurdles. The prototyping phase is also the time to specify a *System Metaphor*. It is thought that a *System Metaphor* creates object and naming

consistency. Essentially, this is a naming and framing exercise which encourages cohesion in architectural modeling.

3.2.1.2 Planning the Releases

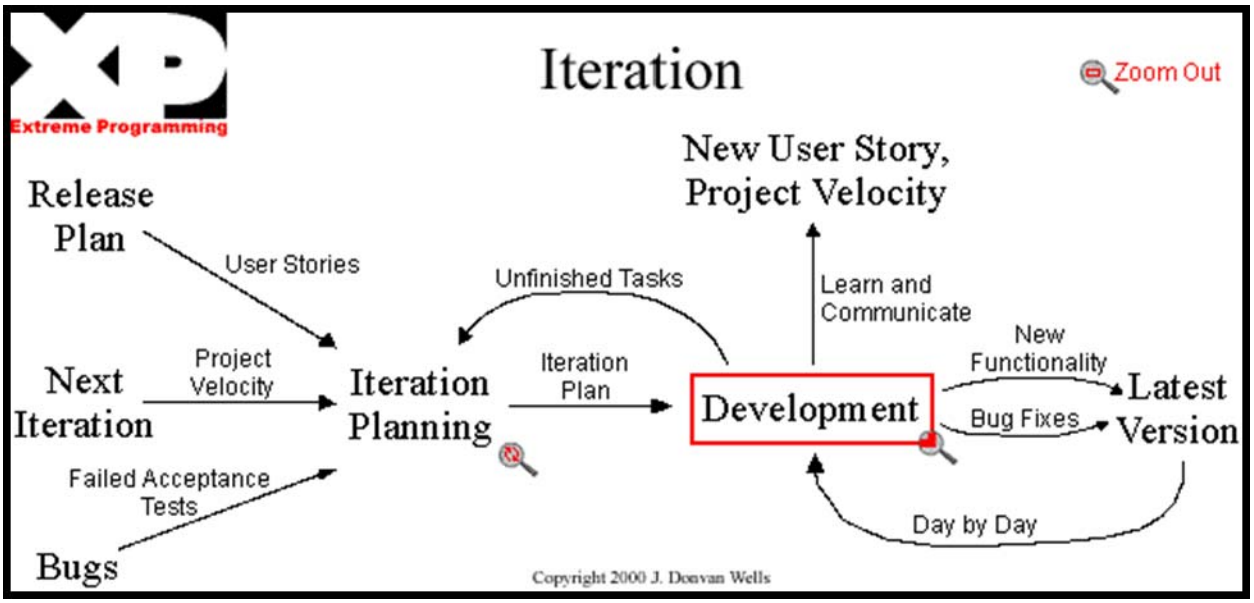
User Stories and *Architectural Spikes* lead to a *Release Plan* which lays out the overall structure of the project. From the *Release Plan* several *Iteration Plans* will emerge. *Release Planning* allows each stakeholder to make decisions appropriate to their role and responsibility in the project: developers make technical decisions and customers make business/domain decisions. *Release Planning* accounts for each *User Story* and estimates the amount of time required to program each story. These time estimates assume, aside from testing, complete devotion to the completion of that *User Story*. Typically, *User Stories* are usually written on index cards and arranged, using the *Planning Game* technique, during *Release Planning*. *Project Velocity* describes the rate at development activities which development and implement *User Stories* requirements are completed. New *Spike Solutions* may emerge during *Release Planning* as dependencies emerge between User Stories. Time estimates may be uncertain during *Release Planning*, such that new *Spike Solutions* are required in order to make confident estimates.

3.2.1.3 Iterations

The *Iteration* is a phase whereupon *User Stories* are developed into working software. XP revolves around *Iterations* as *Iterations* are at the heart of this method and most other agile methods. *User Stories* from the *Release Plan*, known bugs from failed *Acceptance Tests*, deferred *User Stories* and *Project Velocity* all influence *Iteration Planning*. The *Iteration Plan*

specifies and schedules development for an *Iteration*, if any tasks are not completed during the *Iteration*, these tasks are carried over to the next *Iteration Plan*. Given the close and frequent customer contact encouraged in XP, it is possible for *User Stories* to change such that new *User Stories* emerge during the *Iteration*. The outcome of *Iterations* tests the new functionality which contributes to *Small and Frequent Releases of Working Software*. Figure 17 highlights elements of the *Iteration* phase.

Figure 17 Iteration Phase of XP (Wells 2000)

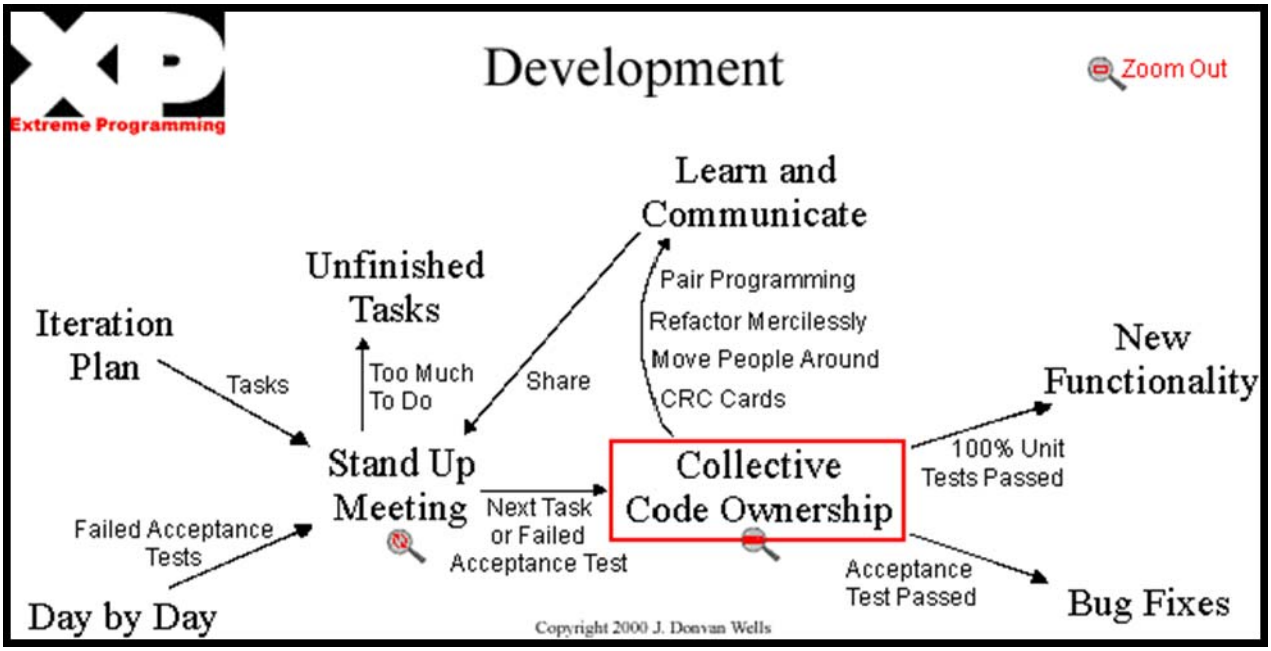


3.2.1.4 Development

An important part of XP is face-to-face communication. This communication happens within an XP team and between the team and the customer. XP calls for a *Daily Standup Meeting* among developers which fosters open communication of problems, ideas, solutions and new directions. More importantly, the *Daily Standup Meeting* facilitates learning and promotes the refactoring which is such an essential element of the XP approach. A unique and

controversial approach prescribed in the XP method is *Pair Programming*. *Pair Programming* suggests that all production code is to be created with two people working together at a single workstation to increase software quality. Many consider this approach is unconventional and it is a source of controversy (Aiken 2004; McBreen 2002a). Figure 18 relates the principles phases of the development phase.

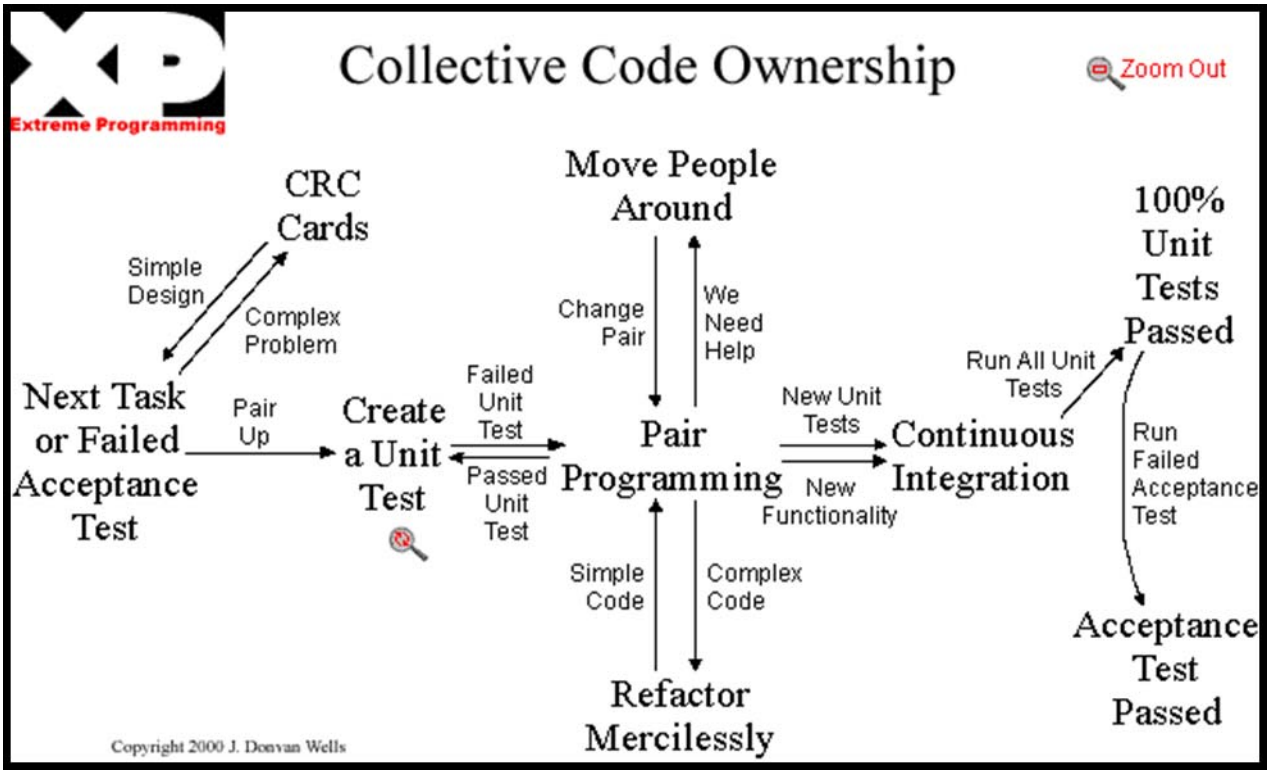
Figure 18 The Development Phase of XP (Wells 2000)



Several XP development practices are peripheral and orbital to *Pair Programming*: *Unit Tests*, team rotation, *Refactoring*, and continuous integration; Figure 19 demonstrates the relationship among these practices. *Unit Tests* are written by the XP team developers as a testing framework for the modules of functionality within the project. *Unit Tests* give immediate feedback concerning the viability of a module. This is a test-driven and test-first philosophy which permeates through many of the agile software development methods. Team rotation suggests that it is easier for the entire team to get a sense of the project if people are rotated among different development roles. For XP this means changing pairs and seeking help.

Refactoring is activity which takes complex (yet working) code and seeks to simplify this code as often as possible. *Refactoring* looks for patterns of redundancy, unused functionality, and obsolete design and prunes them from the codebase. Lastly, the development stage specifies *Continuous Integration* of new working code into the working code from the previous iteration. This *Continuous Integration* is made possible by subjecting functionality to *Unit Tests* and *Acceptance Tests*. If all *Unit Tests* and *Acceptance Tests*, both from the current iteration and all prior iterations passes, then the current iteration's code becomes a part of working software which is then used by the customer.

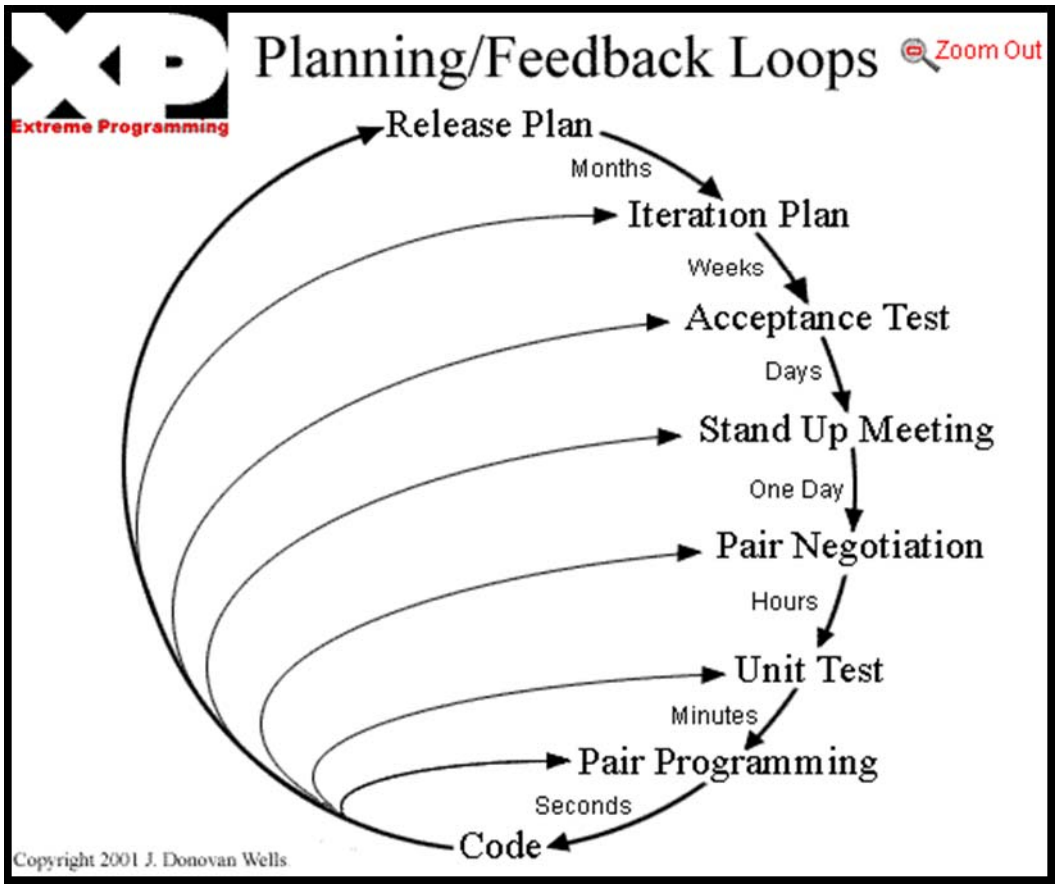
Figure 19 Programming Practices within XP (Wells 2000)



3.2.1.5 Acceptance and Small Release Cycles

An important part of XP is the concept of *Customer as Team Member*, where the customer is directly involved throughout the project. In XP, the customer does not review the "final" software after months or years of development and give a binary answer concerning acceptance; XP specifies that the customer is a part of the process. The initial *Release Planning* allows the customer and developer to elicit requirements through the *User Stories* and *Release Planning*. *Release Planning* does not imply a negotiation for a final project delivery date; Release Planning involves setting a timetable and expectations for continuous integration of working software into the customer's organization. New features go operational as they are ready rather than waiting for a single monolithic application at project conclusion. Quality assurance for each incremental release comes from *Unit Tests* and *Acceptance Tests*. *Acceptance Tests* are derived directly from *User Stories* to remain consistent with the original intent of the system. *User Stories* are not complete until all *Acceptance Tests* are passed. This processes serves as a reminder that agile methods are adaptive methods. Part of this adaptability requires that *User Stories* and all tests used to validate quality against these stories can be changed at anytime should the environment or the customer require it. XP, as is the case with many of the agile methods, is designed to iteratively accommodate change. Figure 20 demonstrates the importance of iterative cycles throughout the XP development cycle.

Figure 20 The Iterative Nature of XP (Wells 2000)



3.2.1.6 Reflecting on Extreme Programming

On closer inspection, XP looks to be an appropriate method for small-team software development as it was specifically designed for small teams (Beck 1999). XP's iterations keep a developer on task and coordinate the efforts of a software team. However, as the goal for this research is a Reflective-Agile learning method for small-team software development, there are aspects of XP which may or may not be required. For instance, how important or vital is *Pair Programming*? Many have advocated for and or argued against this element of the XP method as practitioners and researchers have come to grips with XP.

3.3 Introducing Elements of Reflective Practice

The principle strategy for developing the proposed Reflective-Agile learning method is a synthesized approach which matches elements of Extreme Programming (XP) with Schön's epistemology of *Reflective Practice*. This synthesis has been proposed by Hazzan and Tomayko in numerous papers and books (Hazzan 2002; Hazzan et al. 2003; Hazzan et al. 2004a; Hazzan et al. 2004b; Hazzan et al. 2004c; Hazzan et al. 2005; Tomayko et al. 2004) and implied by others (Highsmith 2000; Nerur et al. 2007). The initial Reflective-Agile methodology for small-team development is derived from Extreme Programming and Hazzan and Tomayko's work which introduces Schön's reflective practice (Schön 1983) to Extreme Programming (Beck 1999) and software development in general. By introducing *Reflective Practice* to an agile software development method, this research can specify the impacts of this introduction on small-team software development, agile methods and the Reflective Practitioner framework in general.

The interventions, related to *Reflective Practice*, to be introduced in the Dialogical AR setting consist of: daily use of journals (web logs) for *Reflection-on-action* (Mathiassen 1998; Mathiassen et al. 2002); a team wiki for *Organizational Learning* and *Reflection-on-action* (Leuf et al. 2001); and the use of the *Ladder of Reflection* to facilitate *Reflection-in-action* (Tomayko et al. 2004). Each of these interventions is grounded in the epistemology of *Reflective Practice* and design.

3.3.1 The Ladder of Reflection

Hazzan and Tomayko (2004) offer a method for introducing *Reflective Practice* to Extreme Programming by utilizing Schön's *Ladder of Reflection*. (1987, p.115) Schön (1987) describes the *Ladder of Reflection* as is a process by which a reflective dialog on designing, and with the materials of designing, transpires. The rungs of the reflective ladder are as follows (movement is from the bottom to the top and back down as necessary):

Table 22 Rungs on the *Ladder of Reflection*

Rungs on the Ladder of Reflection
Reflection on the Reflection on the Description of Designing (reflecting on the dialog itself) [Meta-level to reflection]
Reflection on the Description of Designing (What does the description mean?) [Meta-level to description]
Description of the Designing (appreciation, critique, advice)
Designing (*Reflection-in-action*)

The reflective ladder works by moving "up" from designing (at the bottom) through to reflection on reflection on description of designing (at the top) as a means of eliciting theories-in-use and the tacit understandings of practice (Hazzan 2002; Hazzan et al. 2003; Hazzan et al. 2004a; Schön 1987; Tomayko et al. 2004). While Schön (1987) extensively illustrates the reflective ladder using an architecture design studio example, Tomayko and Hazzan (2004) illustrate the use of the reflective ladder with Extreme Programming examples. For many, techniques such as the reflective ladder appear as over-simple and non-rigorous, however, the purpose of the technique is to discover the tacit understandings inherent in action. Table 23 demonstrates one

possible use of the reflective ladder during the Extreme Programming activity of pair programming.

Table 23 A Ladder of Reflection: The Case of Pair Programming (Tomayko et al. 2004)

Reflective Ladder Rungs	Pair Programming Dialog
***Designing* (*Reflection-in-action*)**	Programmer A: "I'm going to use a stack here. Does this make sense?"
***Description of the Designing* (appreciation, advice, criticism, etc.)**	Programmer B: "Good question. Let's explore the nature of the algorithm. Do you remember that in the last retrospective session we discussed a similar problem? What was it about?"
***Reflection on the Description of Designing* (reflection on the meanings behind the original description of designing).**	Programmer A: "You are right. I'm trying to recall. We started by comparing the nature of two projects and concluded that the project we discussed in that retrospective session is similar to what we developed last year. After that, we did a lot of reuse."
***Reflection on Reflection on description of designing* (each in the dialog reflects on the content of the dialog)**	Programmer B: "And, I remember more clearly that following this retrospective session, we decided to change the format of our retrospective session. But more specifically, on the code level, we decided to change the design. Let's try to think in these directions: redesign and reuse. I guess they will save us a lot of time eventually."

Hazzan and Tomayko (2005) describe further uses for the reflective ladder within the Extreme Programming method and also within areas of customer and team interaction. This reflective ladder technique is an operationalization and instantiation of Schön's (1983, 1987) *Reflection-in-action* and *Reflection-on-action.* The reflective ladder is used to elicit an explication of a practitioner's knowing-in-action; the practitioner's *Theory-in-use* becomes clear and enhances understanding of action and beliefs regarding governing variables. Theories and models of

learning and action are all critical to the successful adoption, use and iterative refinement to the baseline Reflective-Agile method. In this sense Argyris and Schön (1974), Schön (1983; 1987) and Senge (1994) each underscore the importance of an adaptive learning strategy within an organization; this dissertation will support that this is no different for organizations engaged in small-team software development. The following sections discuss appropriate models and patterns for learning which will assist in the introduction and development of the Reflective-Agile method.

3.3.2 The Learning Organization

In order to successfully introduce the reflective ladder to XP, a small team must be willing to accept that *Reflective Practice* will directly result in an improvement of their development processes. Tomayko and Hazzan (2005) suggest a learning organization (Argyris et al. 1974; Schön 1983; Schön 1987) as an environment which best supports and promotes *Reflective Practice*. A learning organization acknowledges the importance of knowledge and information within the organization and the need to manage these assets (Senge 1994; Tomayko et al. 2004). A learning organization values the professional development of each member of the organization, realizing that effective people will make an effective product. A learning organization also values the accumulation of individual repertoire for *Double-Loop Learning*. Thus, according to Senge (1994), a learning organization provides a working environment in which learning is an integral part of everyday work. In essence, the introduction of *Reflective Practice* should facilitate a learning system where the particulars of a given methodological element are not as important as an ability to adapt and develop repertoire appropriate to the team.

3.3.3 Models and Theories for Organizational Learning and Agility

Argyris and Schön (1974) present a set of widely-accepted and adopted views on how professionals think and behave in a given problem setting. According to Argyris and Schön (1974) there are two theories of action which govern the mental maps practitioners use to guide their actions: *Espoused Theory* and *Theory-in-use*. An *Espoused Theory* is the manner in which a practitioner outwardly expresses what it is that he or she does in action. *Espoused Theories* are embedded in the language we use to describe what it is we do to others or what it is we' d like others to think it is that we do. A *Theory-in-use*, however, consists of the mental maps and strategies that implicitly and tacitly guide action. Put more simply, these two theories of action (*Espoused theory* and *theory-in-use*) represent a disconnect between theory and action in professional behavior.

It would seem that explicating a *Theory-in-use* would be the most helpful by-product of using Schön's (1987) reflective ladder. Argyris and Schön (1974) identify three critical elements for effective learning:

- **Governing Variables**: These are goals and ideals which the practitioner, through satisficing,[12] keeps within acceptable parameters (i.e. "Goal-setting," "Maximize winning and minimize losing," "Minimize generating or expressing negative feelings," "Be rational") (Argyris et al. 1974:66). Governing variables are personal exhortations to "stay on task, on message and on plan."

- **Action Strategies:** Actions and plans for keeping governing variables within an acceptable range of possibilities or states (i.e. "Design and Manage environment unilaterally," "own and control task," "unilaterally protect others," "unilaterally protect

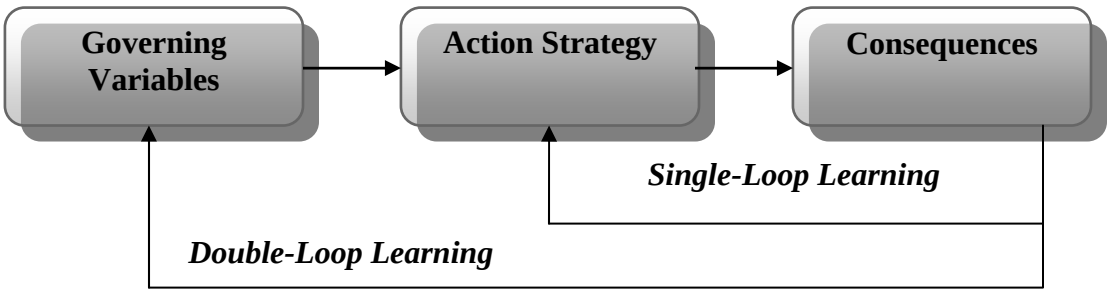
[12] This is a word coined by Herbert Simon as a portmanteau of "satisfy" and "suffice." This represents attempts to reach an adequate balance between optimal solutions and constraints.

self").

- **Consequences:** The intended and unintended results of action for the practitioner and others.

Argyris and Schön (1974) map a relationship to determine whether a practitioner uses the consequences of action to reexamine their action strategies or reexamine both their action strategies and their beliefs regarding governing variables. The distinction between these two approaches to regarding the consequence of action is inherent in what Argyris and Schön (1974) call *Model I* and Model II behavior. *Model I* behavior is exemplified by the exclusive use of S*ingle-Loop Learning*: learning whereupon only action strategies are changed as a result of consequences to action (Argyris et al. 1974:18). Model II behavior incorporates *Double-Loop Learning*, where both action plans and governing variables may be adjusted as the result of the consequences of action. The relationship between *Double-Loop Learning* and *Single-Loop Learning* is illustrated in Figure 21.

Figure 21 Single-Loop and Double-Loop Learning (Argyris et al. 1974)



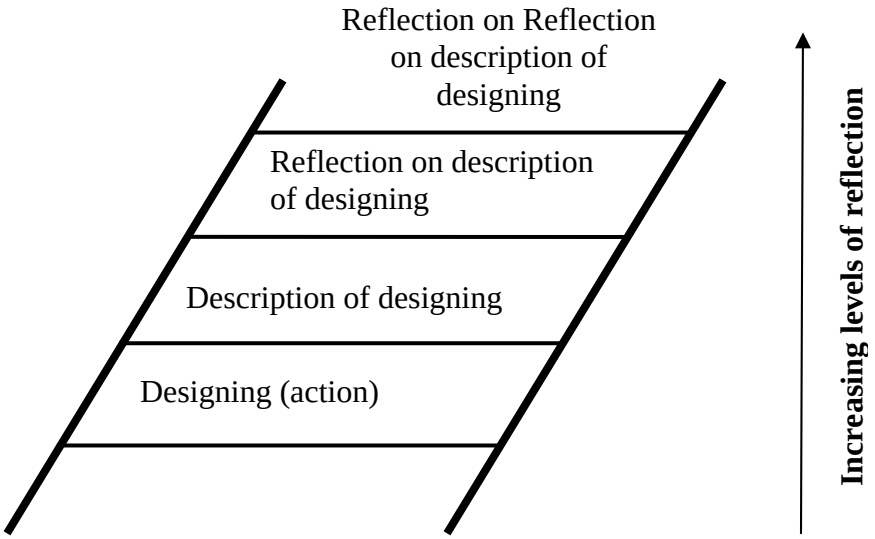
These theories of action and learning illustrate how a theoretical framework concerning change, learning and action is critical to the introduction of *Reflective Practice* to XP. The reflective ladder will not be effective if an environment receptive to learning and change is not

also established. Without an ongoing reflective dialog, it is possible for a small team to be lulled into false assumptions gathered from working closely and informally.

3.4 Summarizing the Elements of the Reflective-Agile Methodology

The preceding sections of this chapter have described the components of a baseline Reflective-Agile method for small-team software development. This section will summarize each of the elements of the baseline Reflective-Agile method in order to tie them together in a comprehensive manner. Figure 16 shows the general steps of the Extreme Programming method and demonstrates areas where the ladder of reflection can be applied. Figure 22 reviews Schön's (1987) reflective ladder:

Figure 22 The Ladder of Reflection (Schön 1987)



The practitioners will utilize the ladder during *Pair Programming* in order to explicate tacit understanding and reflect on theories-in-use. Thus, the reflective ladder enables a reflective conversation with designing and the materials of design. Many of the steps in the XP process can be enhanced by use of the reflective ladder, but this research will focus on *Pair Programming*. *Pair programming* is particularly interesting as the technique practically demands reflection and discussion between the pair (Beck 1999; Boehm et al. 2004; Tomayko et al. 2004). Design is also an area of XP which may benefit from the reflective ladder. *Architectural Spikes*, *User Stories*, the *Daily Standup Meeting* and a *System Metaphor* are all XP activities would benefit from reflective collaboration amongst the team. The *System Metaphor* should capture tacit understanding and theories-in-use from the development team and the customer.

All of the elements of the baseline Reflective-Agile method are now in place. The general structure and requirements for the method are listed below in Table 24.

Table 24 The Baseline Reflective-Agile Software Development Method

Elements of the Baseline Reflective-Agile Software Development Method
1. Learning organization: Establish and commit to a learning environment
2. XP: Adopt and use the Extreme Programming method
3. Reflective Practice: adopt and use reflective ladders in at least the following stages of XP: **a.** User stories **b.** Architectural spikes **c.** System metaphor **d.** Release planning **e.** Iteration planning **f.** Daily Standup Meeting **g.** Pair programming

CHAPTER 4 Research Methodology

The dissertation utilizes a mixed-method research approach which combines Lee's (2007 Design Science and Action Research (DSAR) framework and Dialogical action research (Dialogical AR) (Mårtensson et al. 2004). Additionally, Straus and Corbin's (1998) Grounded Theory is used as a mode of analysis for collecting, analyzing and interpreting the dialogical evidence in order to evaluate the outcomes of the theoretical interventions introduced in the Dialogical AR Partnership. The mixed-methods research approach consists of three parts:

- **Dialogical Action Research** – This is a variant of action research explored and proposed by Mårtensson and Lee (2004) which focuses on reflective dialog between a practitioner and researcher in order to mitigate the dichotomous nature of theory-based knowledge and practitioner knowledge and to balance rigor and relevance

- **Design Science** – A research paradigm grounded in engineering and design conducted to theorize in the development of artifacts. Design science is concerned with problem-solving strategies, practices and techniques related to activities of design: analysis, design, implementation, management and use (Hevner et al. 2004)

- **Interpretive Coding of Qualitative Data** – In order to specify learning in the Dialogical AR cycle, qualitative evidence will be collected and analyzed throughout the dialogs and interventions. The Strauss and Corbin (1998) Grounded theory mode of analysis is used to collect and analyze qualitative evidence from dialogs in the Dialogical AR Partnership

As to design science, March and Smith (1995) offer a simple prescription for conducting design science research: (1) develop and build artifacts and (2) justify and evaluate these artifacts. Also, March and Smith (1995) and Hevner et al. (2004) have suggested validation criteria for artifact building, evaluation and justification. Additionally, Lee (2007) illustrates a framework which promotes the benefits of both action research and design science such that each supports the other in developing and/or testing theories in Information Systems research in a rigorous and

relevant manner. The designed artifact will be evaluated according to the Hevner et al. (2004) criteria as well as the March and Smith (1995) criteria.

The remaining sections of this chapter are as follows: first, there is an introduction to Dialogical AR and its background and applicability in information systems research; this is followed by an outline of the Lee's (2007) framework for coordinating design science and action research and the use of Dialogical AR as a method of inquiry. Next, a justification is offered for the mixed-methods approach taken, which conducts design science research in the interpretive mode of Dialogical AR. Last is a brief discussion on the generalizability of research outcomes using DSAR and Dialogical AR.

4.1 Investigating IS Phenomenon with Dialogical Action Research

In any research effort, there are bound to be questions regarding the rigor and relevance of research methods used versus the real-world applicability of the research outputs (Applegate 1999; Applegate et al. 1999; Davenport et al. 1999; Lee 1999). Action research is a research method which aims to include both rigor and relevance. In action research, the researcher works jointly with practitioner(s) to address the practitioners' problems. Together they analyze the situation at hand, evaluate the issues and reframe the problem and the solution space in an iterative manner. Dialogical AR offers a means to integrate the knowledge of the organizational actor (referred to as *praxis*) (Mårtensson et al. 2002: 5; Mårtensson et al. 2004) and the scientific knowledge of the outside researcher (referred to as *theoria*) (Mårtensson et al. 2002: 6) to promote both knowledge heterogeneity and knowledge contextuality in their mutual problem

solving. Dialogical AR also offers the added advantage of improving the knowledge and expertise of the researcher and the practitioner while addressing a real world problem.

Dialogical AR goes beyond Canonical AR by recognizing the importance and the role of knowledge heterogeneity and contextuality in the research partnership with the practitioners (Mårtensson et al. 2004). Knowledge heterogeneity recognizes that the practitioner's knowledge and the scientist's knowledge are two disparate domains of knowledge and are both equally important. Knowledge contextuality suggests that knowledge loses its meaning when detached from its social context (Mårtensson et al. 2004).

4.2 The Dialogical Action Research Cycle

For the purposes of this dissertation, the Dialogical AR method consists of two principle phases which include some of the steps of the Canonical AR cycle (Baskerville 1999). First, there is a diagnosis and planning phase used to determine the nature of the practitioners' problems. In this phase, the researcher seeks to identify the practitioners' *Espoused Theories* and consequently derive and speculate upon their *Theory-in-use* (Argyris et al. 1978; Argyris et al. 1974). The modus operandi for achieving this goal is to thoroughly transcribe, review and analyze the dialogical evidence and field notes. The transcripts and field notes comprise the qualitative evidence the researcher uses to interpret and understand the practitioners' *Espoused Theory*. The transcription process also allows the researcher to reflect on the tone of the dialogs in order to develop better interpretations. The practitioner's *Theory-in-use* cannot be estimated by direct inquiry, *Theories-in-use* are constructed from the researchers observations and

interpretations of the practitioner's behavior and the transcribed evidence (Argyris et al. 1978; Schön 1983). .

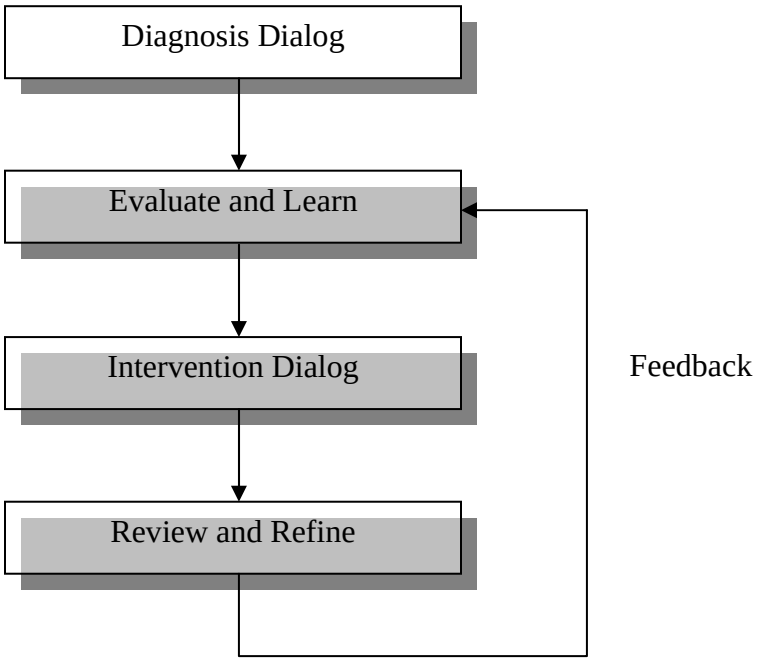
The second phase of Dialogical AR uses the results from diagnosis iteratively in the Canonical AR cycle. The Canonical AR cycle is also augmented with the Dialogical AR concepts of knowledge heterogeneity and knowledge contextuality. The outcome of this phase serves the dual purpose of learning for both the practitioner and the researcher and the development of solutions for the practitioners' problem(s). During this process, the practitioner tries theory-driven actions and interventions based on theories the researcher feels are appropriate to the problem. The evaluation phase of the Canonical AR cycle is also used by the researcher to test the validity of the theories used and thereby augment learning. The learning process provides the researcher an opportunity to improve and strengthen the selected theory and offer progress and learning back to his scientific community. In order to specify this learning, the researcher must identify the assumptions that envelop the practitioner's theories-in-use and then use an established scientific theory from his community of science to interpret the practitioner's problem in her own context. Success in this process perpetuates the learning cycle and potentially augments existing information systems theory. Repeated dialogs with the practitioner are therefore essential for constructive and continuous evaluation of the problem domain: in this sense, the action research cycle should be traversed a few times for the method to be truly effective.

This section is intended to illuminate the positive outcomes possible for both the practitioner and the researcher when using Dialogical AR. Positive outcomes for the practitioner would be solutions to her problem and for the researcher, an opportunity to improve existing theories. Therefore, a researcher who plans to use Dialogical AR as a means to research

information systems phenomena must be prepared to gather in-depth and detailed information via intimate dialogs with the practitioner.

The principle steps involved in the Dialogical AR method (Figure 23), as used in this dissertation, are a based on two pilot studies conducted during and subsequent to Ph.D. seminar in action research given by Dr. Allen Lee in 2003 and 2004. Thus, the methodological steps used in this study are not exactly those described by Mårtensson and Lee (2004), but rather a post hoc operationalization of Dialogical AR developed as a result of the pilot studies. As these pilot studies were used as a pedagogical learning device, neither pilot study represents actual research. Nonetheless, the experience was useful in understanding Dialogical AR method for the purposes of this research. During this research, the processes developed during the pilot studies, the processes described by Mårtensson and Lee (2004) and the processes of Canonical AR were each used as methodological guidance during the field work and dialogs. Thus, Dialogical AR, as outlined by Mårtensson and Lee (2004), was used as the primary methodological guide throughout the study whereupon this study self-identifies with the Dialogical AR method and its philosophical assumptions.

Figure 23 Steps in the Dialogical AR Process



4.3 The Rigor and Relevance of Dialogical Action Research

The following question often follows when one is first exposed to action research (Straub et al. 1998): isn't this just consulting? If we frame Dialogical AR as a variant of action research, which aims to balance rigor and relevance in research outcomes, then it appears that the more difficult of the two concepts to "prove" would be the method's rigor. Questions related to the design, development, deployment and use of an IT artifact are certainly topics worthy of investigation by IS scholars (Benbasat et al. 2003); however, the question is: is Dialogical AR the appropriate vehicle?

As it has been suggested that IS research demonstrate more relevance to practitioners (Benbasat et al. 1999), the IS researcher is left to sort out which methods are appropriate such that research outcomes balance rigor and relevance: clearly the practitioner prefers research outcomes that they can use immediately. The use of action research in information systems research is certainly not a new proposal (Baskerville et al. 1998; Baskerville 1999; Baskerville et al. 1996; Mårtensson et al. 2004; Myers 1997; Straub et al. 1998); the specific issue addressed in this dissertation is whether Dialogical AR offers sufficient methodological rigor.

4.3.1 Why Action Research is an Appropriate Research Method

Information Systems is an applied discipline similar in nature to professions such as architecture. Thus, it would follow that practical outcomes are desirable and expected of research in information systems. While a Positivist research approach informed by the natural science model minimizes the impact and influence of the researcher, realizing relevant research outcomes may call upon information systems researchers to directly effect change in an organization (Baskerville et al. 2004: 329; Mårtensson et al. 2004: 515). In this sense, information systems researchers would purposefully induce “Hawthorne Effects” whereby the actions of the researcher are meant to positively influence outcomes for the subjects of research. If the introduction of an information system is meant to change the organization for the better, it may be necessary for the researcher to “dirty” their hands.

If we accept that information systems research should encourage transformative effects in an organization as a research outcome, then action research is indeed an appropriate research

method. In this regard, Baskerville and Myers (2004) make a compelling argument for why action research is appropriate for investigating information systems:

> *…It is strongly oriented toward collaboration and change involving both researchers and subjects. Typically it is an iterative research process that capitalizes on learning by both researchers and subjects within the context of the subjects' social system. It is a clinical method that puts IS researchers in a helping role with practitioners. (Baskerville et al. 2004)*

If benefit to organizations is seen as a key outcome of information systems research, then the organization can be considered as a unit of analysis in most information systems research endeavors (Scott 2003); in most cases, action research uses the organization as a unit of analysis. The origins of action research can be traced to advances in social psychology and operations research (Emery 1997; Lewin 1947). Social Psychology is concerned, to a degree, with the effect the organizational actor has on the organization itself. From the Tavistock socio-technical perspective, the organizational actor is included as a part of a wider system which encompasses both a social and technical system (Emery 1997). If we accept this socio-technical antecedent for action research, then we would necessarily distinguish between the "technological system" and the "social structure," which consists of occupational roles and their institutionalization within the organization (Emery 1997: 4).

As action research is a means to produce rigorous and relevant research outcomes when investigating socio-technical systems, Baskerville and Myers (2004) suggest four key premises of action research which underscore why action research is an appropriate information systems research method. The four premises of action research are: consequences define human concepts; practical outcomes embody truth; the logic of controlled inquiry; the social context of action. We can also relate these premises to key principles of socio-technical systems analysis where an information system is viewed as an emergent structure resulting from the interactions

between the people and the technology within the system. We now examine these four premises from the socio-technical perspective.

4.3.2 The Consequences of Experience

A pragmatic premise of action research suggests that it is only through the consequences of experience (first hand or derived) that we conceive of reality (Baskerville et al. 2004). We can only understand the technical and social elements in an information system if the purpose of each component is fully rationalized. Thus, we must conceive of each part as being meaningful as the interrelation of the parts in an information system is clearer when each individual part has meaning (Emery 1997). In keeping with the principle of the Hermeneutic Circle, action research aims to understand the emergence of the whole, as a result of studying the "parts" of the information system (Baskerville 1999: 2). We can also simplify this premise by accepting Baskerville's (1999) position that "action brings understanding."

4.3.3 Truth in Practical Outcomes

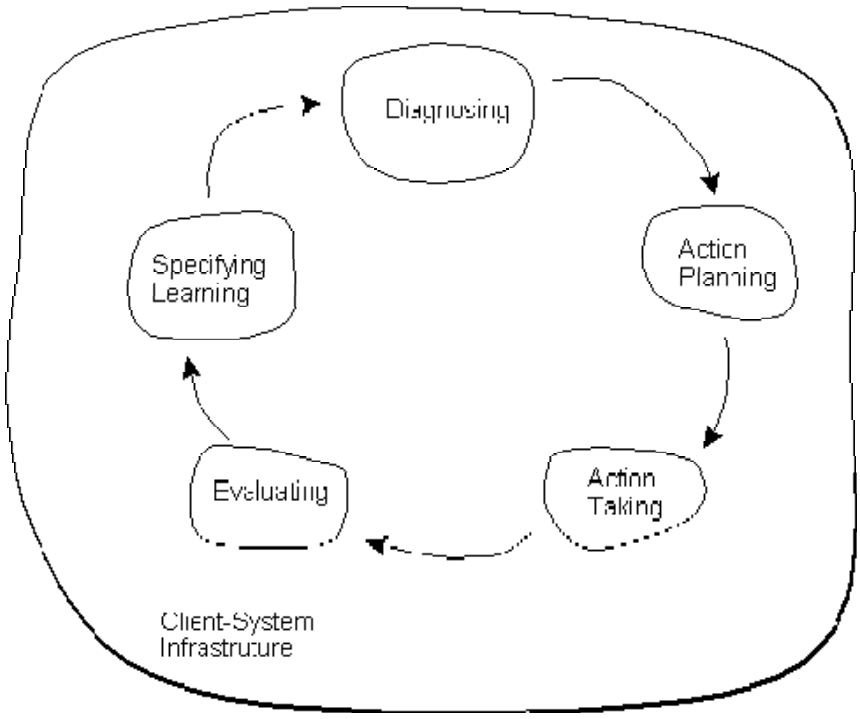
Another premise of action research suggests that we derive truth in accordance with the practical and observable outcomes of action. If theory is a process used to approximate and predict real behaviors and outcomes, then research methods advocating direct action, such as action research, are certainly conducive to theory building (Weick 1995). We can appreciate an innate human quest for truth from a psychological perspective and propose that action is the only proven means to realize truth which is generalizable and widely accepted. As an information

system is an emergent system, the process of direct investigation used in action research is a logical choice for deriving truth in research outcomes.

4.3.4 The Logic of Controlled Inquiry

Action research holds as one of its central tenets that the researcher must specify learning which contributes to a body of knowledge held by a community of science. Learning can be considered the primary means by which the social and technological structures in an information system are created (Baskerville et al. 2004: 332). John Dewey's principles of controlled inquiry, the process by which we establish the facts that govern our operating model of the world, has influenced the steps in the action research cycle. Thus, it is through controlled inquiry that research is able to specify the learning which builds and sustains the elements of an information system. Dewey's principles of controlled inquiry are: (1) intermediate situation; (2) problem development; (3) find a solution; (4) reasoning; (5) operationalization of facts (Dewey in Baskerville et al. 2004: 332).

Figure 24 Action Research Cycle (Baskerville 1999)



When Dewey's principles of controlled inquiry are contrasted against the action research, we can see the similarities between Baskerville's (199) stages of the action research cycle (Figure 24) and Dewey's controlled inquiry process.

4.3.5 Social Context of Action

The last of Baskerville and Myers' (2004) premises of action research is directly related to Dialogical AR. This premise suggests "…that any human behavior that elicits a response from another individual constitutes a social act" (Mead in Baskerville et al. 2004: 332). Therefore, this premise holds that it is social interactions which create the context from which all actions are based. As Dialogical AR focuses on knowledge contextuality and heterogeneity in

the practitioner/researcher partnership, the actions and interventions undertaken in the partnership consider the social and historical context of emergent solutions to the practitioners' problems (Mårtensson et al. 2004: 517).

4.4 Dialogical AR is both Scientific and Rigorous

Baskerville and Myers' (2004) four action research premises are offered as justification for using action research as an appropriate method for investigating information systems. However, the justification for the method's scientific rigor has yet to be made. To this end, Baskerville (1999), Baskerville and Wood-Harper (1996), Myers (1997) and Straub and Welke (1998) have each offered explanations for how action research is a scientific and rigorous research method. The aims of Dialogical AR are no different than any other scientific endeavor: the method examines responses to experimental stimulus in real-world situations in order to derive evidence to confirm or disconfirm scientific theory and also provide remedies for the "real world" problem (Mårtensson et al. 2004: 3). Thus, it is the reflective and iterative nature of Dialogical AR which facilitates a scientific process of inquiry (Baskerville 1999; Mårtensson et al. 2004). A key criticism levied against action research is that the method can easily be confused with consulting; this is likely due to the involved and applied nature of the method. Mårtensson and Lee (2002, 2004) provide the following defense against any such accusations in asserting that Dialogical AR:

- Specifies self-learning in contribution to a community of science while consulting does not
- Dictates a partnership whereas a consultant is a service provider

- Encourages learning from negative results in an iterative fashion until leaning and remedy can be achieved

Dialogical AR is distinguished from "normal" Canonical AR based on a few key philosophical perspectives (Mårtensson et al. 2004: 6-7):

- Dialogical AR differentiates between the scientific attitude (using a theoretical body of knowledge [theoria] and a systematic manner of reasoning) and the natural attitude of everyday life (practical common sense and tacit knowledge [praxis])

- Dialogical AR recognizes the role of the social and historical context within an organization and seeks to understand this context in how the organization is structured

This focus on knowledge contextuality is central to the Dialogical AR approach as scientists need awareness of the organizational context in order to properly theorize and prescribe remedies for the practitioners' problems (Mårtensson et al. 2004: 8).

Ultimately, the justification for Dialogical AR's scientific rigor can be made on philosophical and logical grounds with respect to the philosophy on science. Adopting a phenomenological position of Schutz and Berger and Luckmann, Lee argues in Mårtensson and Lee (2004) that context holds the key for any rigorous scientific method. This assertion is in reference to the fact that scientific endeavor, outside the context of a particular community of science, is neither scientific nor rigorous. Concepts such as truth, rigor and science are social constructions to which members of a community of science or practice accede through socialization (Berger et al. 1967). Plainly, this principle is similar to the colloquialism: one man's trash is another man's treasure. The assertion is made that science, outside the social context of a community of scholarship and its attendant discipline, can hardly be called as such (Berger et al. 1967; Mårtensson et al. 2004: 9; Schutz 1967).

Mårtensson and Lee (2004) provide seven points (Table 25) which defend the rigorousness of Dialogical AR and demonstrate how a single-site case study can be scientific, rigorous and generalizable (Lee et al. 2003; Mårtensson et al. 2004).

Table 25 Supporting the Scientific Rigor of Dialogical AR (Mårtensson and Lee 2004)

Justification	Description
Phenomenology	Dialogical AR is rooted in a phenomenological conception of science where the practitioner holds first-level constructs: sense-making and tacit knowing. The scientist holds second-level constructs: scientific theorizing based on observation of the practitioner's first-level constructs. Whether the domain is the social sciences or the natural sciences, the concept of second-level constructs (theorizing on that which is observed) is logically consistent in all scientific endeavors. Studying actors within in a system in action is just as scientifically valid through strictures of logic and empirical testing.
Contextuality of knowledge	The contextuality of science, as a social construction, ensures that the scientific attitude will not corrupt the natural attitude of everyday life in the conduction of Dialogical AR. Theoria cannot enter into the context of practice until this knowledge is appropriated by the practitioner on her own terms and understanding.
Weltanschauung	Theoria and Praxis present two different cultural worlds that are distant enough such that "contamination" is never possible while appropriation is and would constitute learning.
First and Second-level constructs	The practitioner, being from another "culture" can't truly perceive of the researcher's actions and interventions as experimental stimulus. This is also true of the researcher.
Building shared context	Team building occurs through the dialog in order to establish shared context. The meetings form a partnership whereby the scientist is allowed to interpret and diagnose and the practitioner is afforded some assistance with her problems.
Culturally distinct	As members of separate cultures, the scientist and practitioner each have their own language, logic and terminology.
Knowledge heterogeneity	Dialogical AR recognizes that theoria has no dominance over praxis in importance: both collude to benefit and learn from each other.

On the face of logic, the justifications for the scientific rigor in Dialogical AR suggest that, given the proper scientific attitude and conduct, Dialogical AR can be used in a rigorous manner. As with any research, the quality and rigor rest in the manner with which a method is used as a tool of scientific inquiry.

4.5 Validating the Designed Artifact using Design Science and Action Research

In the iterative development of a Reflective-Agile learning method, this dissertation is concerned with software development methods as a design process more so than a design product. While the qualities of the designed artifact are important, this dissertation is concerned with how a small-team software development learns in their use of the designed artifact. This raises the question: in what way is a method important to a small software development shop? Is the benefit in the process or product? Thus, the Dialogical AR method is used to iteratively illuminate the effect *Reflective Practice* has on a small team's learning processes as they use XP. The utility of introducing *Reflective Practice* is that *Reflective Practice* describes a process of learning and reflection.

There is growing recognition that Information Systems is a discipline centrally focused on IT artifacts and the design thereof (Benbasat et al. 2003; Hevner et al. 2004; March et al. 1995; Orlikowski et al. 2001). The act of designing is meant to produce artifacts, these artifacts may include constructs, models, methods and instantiations (Hevner et al. 2004). Hevner et al. (2004) emphasize that building design artifacts provides a means of validating design artifacts. Whereas Hevner et al. (2004) suggest that we seek to prove the utility of a design rather than the behavioral-science goal of truth, there is a link between truth and utility in a design such that each influences and informs the other. Moreover, many of "wicked problems" related to technology-induced change arise out of human factors in the problem. It is these "wicked problems" that Hevner et al. (2004) propose design science research should address. Thus, the designed artifact in this research is both a model and method intended to provide utility to practitioners in addressing the very human goal of *Organizational Learning*. In order to assess

the correctness and utility of a design science artifact, Hevner, et al. (2004) offer seven guidelines and criteria for the purpose of assessing the model as an IT artifact (Table 26).

Table 26 Guidelines for Design Science Research (Hevner et al. 2004)

Guideline	Description
Guideline 1: Design as an artifact	The designed artifact must be formally stated and address construct, model and method.
Guideline 2: Problem Relevance	The designed artifact must yield utility. The need for this model is demonstrated by the emergence of the Agile Software Movement.
Guideline 3: Design Evaluation	A method for demonstrating the utility of the model is required. Likely choices are Observational and Descriptive design evaluation methods.
Guideline 4: Research Contributions	The problem which the designed artifact addresses must be demonstrated as valid and pressing. The designed artifact should be innovative or at least be efficient and effective in contrast to existing models.
Guideline 5: Research Rigor	The development designed artifact must be distinguished from the common practice of and established as design research via formal and coherent means
Guideline 6: Design as a Search Process	In designing the artifact, the problem space must be illustrated such that the designed artifact can be recognized as a valid solution.
Guideline 7: Communication of Research	The designed artifact must be clearly conveyed in a manner which could be useful to practitioners, researchers and managers.

Further to Hevner et al.'s (2004) guidelines, above, is a general design science research framework suggested by March and Smith (1995). Figure 25 demonstrates the relationship between design science research activities and design science research outputs. The cells in this model represent research activities which allow for both relevance and rigor.

Figure 25 Research Framework for Design Science (March and Smith 1995)

Research Activities

Research Outputs	Build	Evaluate	Theorize	Justify
Constructs				
Model				
Method				
Instantiation				

The utility of design science is its research outputs: models, methods and instantiations. As is the case with action research, the researcher using design science intervenes in the problem setting. Thus researcher does not act as an impartial and unbiased observer but rather actively participates and takes an active role in order to demonstrate utility in efficient and effective designs (Lee 2007:48).

As evidenced in Table 26, Hevner et al. (2004) also emphasize the importance of evaluating a design after its construction. On the subject of evaluation, Hevner et al. (2004) exhort "…The selection of evaluation methods must be matched appropriately with the designed artifact and the selected evaluation metrics" (Hevner et al. 2004:86). With intent to demonstrate the "…goodness and efficacy of an artifact…," Hevner et al. (2004: 86) suggest the researcher carefully selection an evaluation method such that the quality of the artifact can be demonstrated. Measurement of quality is somewhat challenging as "…the measurement of style lies in the realm of human perception and taste" (p. 86). Therefore, efforts to measure the quality of a methodology are used to inform the "theorize and justify" columns of March and Smith's research framework for design science (Figure 26). To the end of evaluation Hevner et al.

(2004) suggest various evaluation methods (Figure 26); these evaluation methods are various and attempt to cover a wide range of artifacts.

Figure 26 Design Science Research Evaluation Methods (Hevner et al. 2004)

1. Observational	Case Study: Study artifact in depth in business environment
	Field Study: Monitor use of artifact in multiple projects
2. Analytical	Static Analysis: Examine structure of artifact for static qualities (e.g., complexity)
	Architecture Analysis: Study fit of artifact into technical IS architecture
	Optimization: Demonstrate inherent optimal properties of artifact or provide optimality bounds on artifact behavior
	Dynamic Analysis: Study artifact in use for dynamic qualities (e.g., performance)
3. Experimental	Controlled Experiment: Study artifact in controlled environment for qualities (e.g., usability)
	Simulation – Execute artifact with artificial data
4. Testing	Functional (Black Box) Testing: Execute artifact interfaces to discover failures and identify defects
	Structural (White Box) Testing: Perform coverage testing of some metric (e.g., execution paths) in the artifact implementation
5. Descriptive	Informed Argument: Use information from the knowledge base (e.g., relevant research) to build a convincing argument for the artifact's utility
	Scenarios: Construct detailed scenarios around the artifact to demonstrate its utility

As a Lee's (2007) DSAR framework will be used to iteratively develop the Reflective-Agile learning method, and since Dialogical AR is rooted phenomenology, then an observational and descriptive evaluation method would be most appropriate for the designed artifact in this research.

4.5.1 Appropriating Lee's Design Science and Action Research Framework

The commonality between action research and design science is that both research methods focus on balancing rigor and relevance. This balance can be attributed to the "real world" focus espoused in each research approach (Benbasat et al. 1999; Lee 2007). Relevance is thought to be the degree to which the outcomes of research have direct and evident applicability to the problems faced by individuals, organizations and societies (Lee 2007:44). In facilitating relevant outcomes, action research requires that the researcher enters directly into the problem setting such that direct observation and intervention is possible. Similarly, design science provides relevance in that the IT artifacts designed and implemented are required to address a relevant and nontrivial problem. In both methods, there is no ambiguity in the provision for relevance; both methods have relevance embedded in their fabric making these two research methods complementary (Lee 2007).

Lee's (2007) DSAR framework builds on the March and Smith (1995) design science research framework and enhances this framework with action research. In using the DSAR framework, the designed artifact will be iteratively designed and evaluated using Dialogical AR. Lee's (2007) DSAR framework fuses both research approaches in a manner appropriate to this dissertation. The principle argument Lee makes in proposing the DSAR framework is his assertion that action, itself, is a product and process of design as designing is a human activity. When engaged in designing are the result of artificial processes; our designs and designing are conscious actions taken to produce that which does not naturally exist. Thus, action is an artifact as it is human-made; action and artifacts, wrought by design science and action research,

are by-products of willful and intentional human intervention to achieve goals and solve problems (Lee 2007:49).

Using Lee's DSAR framework, the phases of the action research cycle specified in Figure 24 are used to conduct and guide the design science research activities in the March and Smith (1995) framework shown in Figure 25. Lee's (2007) DSAR framework acts as a direct extension and "actuator" of the March and Smith (1995) design science framework. Lee (2007) suggests that complementary fit between these two approaches is possible in that they have similar aims and goals: to present relevant solutions to "real world" problems. Thus, action research is a means of operationalizing the March and Smith (1995) design science framework in a methodological sense.

Figure 27 Lee's Design Science and Action Research Framework (2007)

			Design-Science Research Activities in Action-Research Cycle N			
			Build	**Evaluate**	**Theorize**	**Justify**
Action Researcher	***Design-Science Research Outputs***	Constructs				
		Model				
		Method				
		Instantiation				
Practitioner	***Design-Science Research Outputs***	Constructs				
		Model				
		Method				
		Instantiation				

Figure 27 depicts Lee's (2007) DSAR framework for integrating design science and action research. Each of the design science research outputs can be divided between the practitioner and researcher during an action research cycle. This is necessary as action requires that progress and/or learning is specified for both the practitioner and researcher. Furthermore, each member of the Dialogical AR Partnership brings with them different yet complementary

skills which makes the division of labor across the design science research outputs and activities practical. This superimposition of both research approaches is mutually beneficial: action research assists the specific problem domain of design research and design science is methodologically operationalized by action research.

For the purposes of this dissertation, it is apparent that Lee's DSAR framework is well suited to the iterative development and use of the designed artifact. By all accepted accounts on design science, the Reflective-Agile Learning Model and Method is indeed an artifact (Hevner et al. 2004; Lee 2007; March et al. 1995; Simon 1996).

4.5.2 The Benefits of using an Action Research Partnership in Design Science

The provisional DSAR framework shown in Figure 27 is incomplete as it does not describe which roles required of the researcher and practitioner. Lee (2007) recommends a division of labor which is appropriate to expertise of the participant; Lee divides this expertise as theoria and praxis (Lee 2007; Mårtensson et al. 2004). Accordingly, the DSAR framework (Figure 27) divides design science research outputs between the practitioner and the researcher. Similarly, it is not necessary that the practitioner and researcher to participate in all of the research activities and/or outputs in the DSAR framework. The research outputs are the purview of the researcher and are of no concern to the practitioner. Also, there are activities related to instantiation that the researcher need not take part. The areas of the DSAR framework most appropriate for either the practitioner or researcher are evident in Figure 28.

Figure 28 Adjusted DSAR Framework (Lee 2007)

			Design-Science Research Activities in Action-Research Cycle N			
			Build	**Evaluate**	**Theorize**	**Justify**
Action Researcher	***Design-Science Research Outputs***	Constructs				
		Model				
		Method				
		Instantiation				
Practitioner	***Design-Science Research Outputs***	Constructs				
		Model				
		Method				
		Instantiation				

Figure 28 demonstrates the nature of the partnership between researcher and practitioner in the DSAR framework. In the Dialogical AR Partnership, the researcher takes greater responsibility in specifying theory-based interventions for research outputs related to constructs, models and methods, whereas the practitioner would be concerned with areas of method and instantiation. This division of labor focuses the researcher on matters pertaining to *theoria* and focuses the practitioner on matters pertaining to *praxis*.

A noteworthy aspect of Lee's (2007) framework is manifested in the equivalence between action, artifact, intervention, treatment and stimulus. The actions taken by the practitioner partner are treated as artifacts just as the software development method used is treated as an artifact. The practitioners' action, awareness of their actions (which would be enhanced by *Reflective Practice*) and the researcher's observations of the practitioner's actions each become artifacts subject to the phases and steps of the DSAR framework (Lee 2007:54).

4.6 Using the Design Science and Action Research Framework

This section describes how the DSAR framework is used to iteratively develop, implement and refine the designed artifact. The Dialogical AR team will develop constructs, models and methods which the practitioner then instantiates using the researcher's suggested interventions for guidance. The researcher bases interventions on theory-driven constructs, models and methods. Lee (2007) suggests several possible data collection and analysis techniques which can be used to evaluate the artifact(s). Yin's (1994) case study research techniques could be used to build constructs. The open, axial and selective coding in Strauss and Corbin's (1998) Grounded Theory could also be utilized to elicit constructs. The categories which emerge from Grounded Theory's coding activities could then be used as the constructs described by March and Smith (1995). Lee (2007) further describes how a Grounded Theory mode of analysis can be used to evaluate constructs. In Grounded Theory, a category is "saturated" when no further coding can contribute further to a category. Thus, when "…no new properties, dimensions, conditions, actions/interactions, or consequences are seen in the data…" a category is said to be saturated (Strauss et al. 1998:136); thus, a saturated category could be a thoroughly evaluated construct.

A researcher using the DSAR framework may want to develop a model which relates the various constructs. This model can be likened to theory: An action researcher will prescribe theory-driven interventions with the intention of improving upon, modifying or amending this model (Lee 2007:55). Whether the researcher uses existing theory or builds and develops new theory, there are well-accepted theory validation criteria for those assuming a qualitative,

quantitative or critical perspective. Among the possible choices, Lee (2007) suggests using Lee (1991) or Yin (1994).

The method is a design science research output which guides the practitioner's instantiations. This is not the research method the researcher uses to do his or her research; this is a method which the practitioner uses to develop the artifact. A method is usually a set of steps and techniques to reach a goal or desired end-state: to build an instantiation. In the case of Dialogical AR, the action-planning phase of the action research cycle would be appropriate for method development. In their dialog, the researcher and practitioner plan the specific "moves" which the practitioner will use during the action-taking instantiation phase of the DSAR framework (Lee 2007:56; Schön 1983:158).

The remaining DSAR activities are the responsibility of the researcher: to theorize and justify. Given the construct and model-building stages, the action researcher would react to the practitioner's instantiation and determine the implications the instantiation has for the constructs and model (i.e. the implications for theory). Subsequently, the action researcher specifies learning in two ways: (1) in diagnosis and suggested remedies in the next action-taking phase; and (2) in offering explanations regarding what did and did not work in the previous iteration and suggesting how the constructs, models and methods might change in result. Generally, theorizing and specifying learning (the justify phase in the DSAR framework) are needed as the consequences of action should help ot identify short-comings and failures in the model and constructs (Lee 2007).

4.6.1 Fitting the DSAR Framework to the Research Approach

The constructs influencing the model and methods are principally drawn from the work of Argyris and Schön (1978, 1996) and Schön (1983, 1987) concerning theories of action for *Organizational Learning* and how professionals think in and on action. These constructs are:

- **Learning organization**: An organizational environment committed to *Double-Loop Learning*
- **Reflective Practice**: A system for *Reflection-in-action* an *Reflection-on-action* which facilitates *Double-Loop Learning*

These constructs will inform a model and method for small teams to learn in their use of one or more software development methods. A basic and initial sketch of the model suggests that the actors within a learning organization would engage in reflective practice, elucidated and supported by the reflective ladder and XP, in order to reflect in a regular and systematic manner, on the consequences of their actions. *Reflection-in-action* and *Reflection-on-action* are used to regularly consider the consequences of action with respect to their beliefs regarding their governing variables. Thus, the learning model and method will promote *Double-Loop Learning*. *Reflective Practice* promotes *Double-Loop Learning* as it exposes the *Theories-in-use* which connect individual action to beliefs regarding governing variables. Often, the practitioners' *Espoused Theories* of action reflect unrealistic or inappropriate attitudes concerning governing variables. Learning is confounded by discrepancies between *Theories-in-use* and *Espoused Theories* such that only *Single-Loop Learning* rather than *Double-Loop Learning* occurs.

Figure 29 Initial Model of Constructs for the Reflective-Agile Method and Methodology

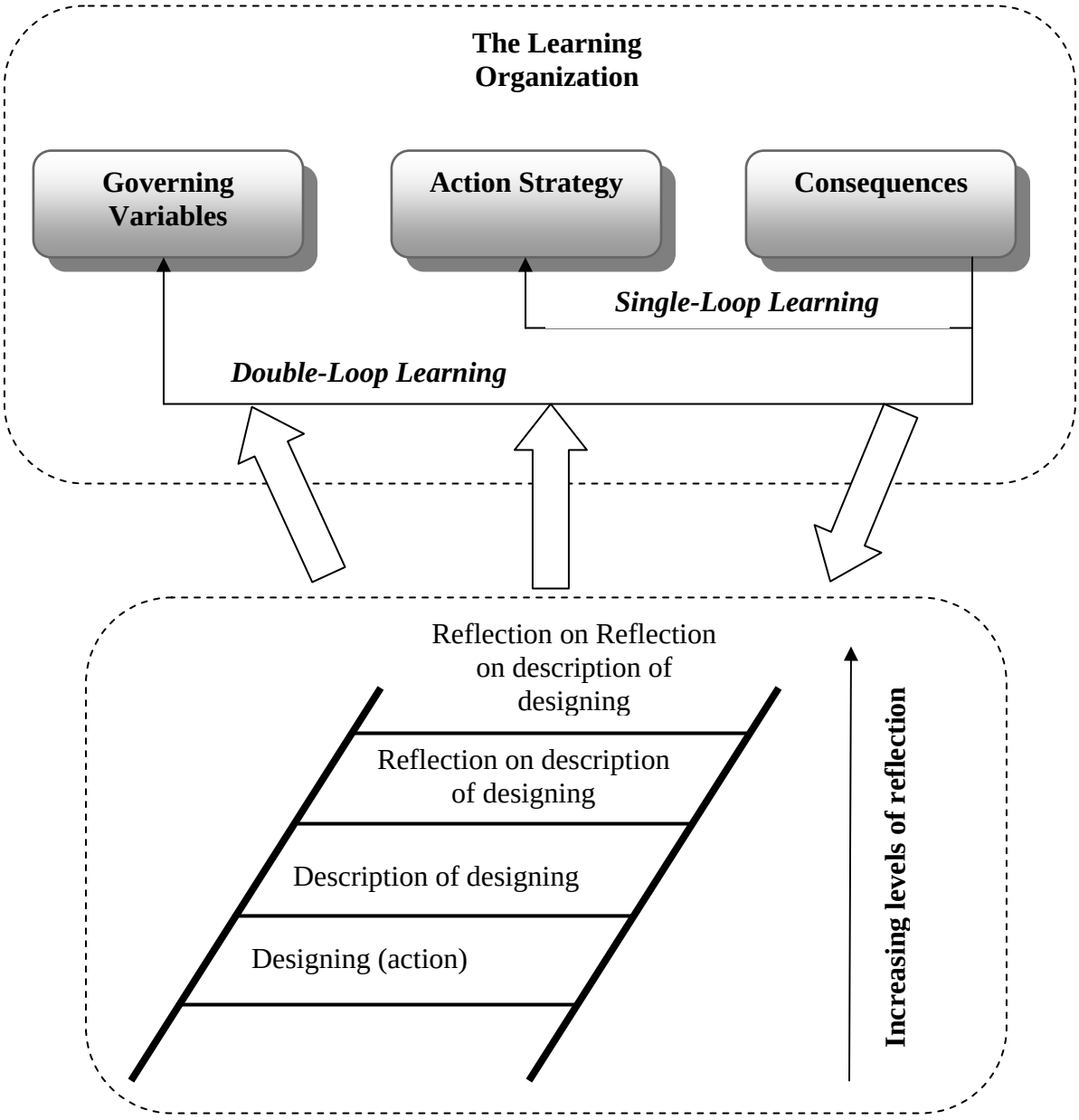


Figure 29 depicts an initial model which relates the theory-based constructs. In the Dialogical AR Partnership, the model will be applied as the practitioners adopt Extreme

Programming. During this process, the Dialogical AR Partnership will utilize the DSAR framework and develop constructs, models, methods and instantiations according to the DSAR matrix. Also, the Dialogical AR method allows for an iterative exploration of the constructs, model and method for continuous refinement. The action research cycle will be used to move the Dialogical AR Partnership through the DSAR framework until such time that the researcher can sufficiently prescribe learning for both practice and theory and the practitioners' problems are addressed; endless pursuit of the action research cycle is not practical for the practitioner nor is it practical for the researcher.

Figure 30 Filling in the DSAR Matrix

			Design-Science Research Activities in Action-Research Cycle N			
			Build	**Evaluate**	**Theorize**	**Justify**
Action Researcher	***Design-Science Research Outputs***	Constructs	A	E	E	
		Model	B	E	E	
		Method	C	F	F	
		Instantiation			F	
Practitioner	***Design-Science Research Outputs***	Constructs				
		Model				
		Method	C	F		
		Instantiation	D	F		

A. **Constructs**:
    a. Learning Organization
    b. Reflective Practice
B. **Model**: The working model of the reflective practitioner approach for the learning organization
C. **Method**: The reflective-agile method
    a. Extreme Programming
    b. Ladder of Reflection
D. **Instantiation**: The first introduction of the Reflective-Agile method
E. **Evaluation (Constructs and Model)**: A Strauss and Corbin approach to coding and

categorizing

F. **Evaluation (Method)**: Use of the Dialogical AR cycle to diagnose, act, specify, learn.

Figure 30 demonstrates the anticipated use of the DSAR framework matrix for this dissertation and describes the activities and outcomes to be pursued in the matrix. The "justify" column is grayed out as those activities are not within the scope of this research effort. As this is an initial effort in this area, the outcomes of this research should inform future research where one or more new practitioner settings could be used to justify the designed artifact. Thus, it may not be entirely appropriate to justify the designed artifact in the same Dialogical AR Partnership used to design and develop the artifact.

4.6.2 Taking the Interpretivist Mode of Inquiry for Evaluation

A hypothetico-deductive approach is commonly used to develop and test theory (Lee 1991). This approach is appropriate for quantitative research conducted in a Positivist tradition as well as qualitative research conducted in an Interpretive tradition (Lee 1991: 355). Using a hypothetico-deductive approach, where the researcher enters into research informed a priori by one or more theoretical constructs, raises questions as action research appears to be inductive. It would be possible to conduct action research from a Grounded Theory research perspective, where theory emerges as a result of the action research effort. Addressing this issue predicates on whether a Positivist ("knowledge is 'out there" awaiting discovery) or an Interpretivist ("knowledge is subjectively and socially constructed – knowledge is 'in here' awaiting experience) mode of inquiry is adopted.

This dissertation adopts an Interpretivist mode of inquiry. As Grounded Theory will be used as a mode of analysis to develop and review aspects of the designed artifact, this dissertation takes the position that interpretation is a form of deduction (Strauss et al. 1998:137). Therefore, "…although statements of relationship or hypothesis do evolve from data (… from the specific case to the general), whenever we conceptualize data or develop hypothesis, we are interpreting to some degree" (Strauss et al. 1998:137). Therein lies the importance of the evaluate, theorize and justify portions of the DSAR framework: we must constantly validate interpretations by "…constantly comparing one piece of data to another" (Strauss et al. 1998:137). Taking this position recognizes the necessity of the human element in interpretive analysis and the potential threat the human element poses with respect to distorted meanings and misinterpretation. Thus, rather than relying solely on induction, this dissertation uses *a priori* theoretical constructs from Argyris and Schön (1974, 1978, 1996) and Schön (1983, 1987) to develop and introduce theory-driven interventions in the Dialogical AR Partnership. In starting from theory, this research starts on a deductive footing based on extant theory. From this point, the iterative nature of action research will enable further deductive reasoning derived from the interpretation and analysis.

4.6.3 The Generalizability of the Research Outputs

Lee and Baskerville (2003) provide an extensive analysis of generalizability in IS research which can be used to determine the generalizability of the research outputs from this research. Lee and Baskerville (2003) provide an account of several classes of generalizability and an exposition on Hume's truism such that increases in sample size "… does not establish the generalizability of sample estimates to population characteristics, but can only establish the

generalizability of sample points to the sample estimate" (p. 235). In this sense, Hume's truism, "…a theory may never be scientifically generalized to a setting where it has not yet been empirically tested and confirmed" (p. 240), holds significance for this dissertation in terms of the generalizability of results arising from a single-site and longitudinal research effort.

One possible response to and critique of using Dialogical AR in a single-site setting would encourage further confirmatory work by adding additional case sites. However, according to Lee and Baskerville (2004), neither an increase in the sample size in a statistical study nor an increase in the number of sites in a multisite case study would be indicator of greater generalizability of a theory to new settings. Yin (1994: 37) instead encourages the researcher to "...generalize findings to 'theory,' analogous to the way a scientist generalizes from experimental results to theory." According to Lee and Baskerville's analysis of Hume's truism, "…there is only one scientifically acceptable way to establish a theory's generalizability to a new setting: It is for the theory to survive an empirical test in that setting" (p. 241). Thus, theorizing on designed artifact would provide an interpretive understanding which can be further utilized to develop a Positivist understanding or enhance a subjective understanding (Lee 2002).

Lee and Baskerville (2003) provide a generalizability framework whereby most forms of generalizing are discussed. Table 27 demonstrates the Lee and Baskerville (2003) generalizability framework where generalizing is from or to empirical statements (field work and observations), or, from or to theory.

Table 27 Lee and Baskerville's (2003) Generalizability Framework

	Generalizing to Empirical Statements	***Generalizing to Theoretical Statements***
Generalizing from Empirical Statements	**(EE)** Generalizing from data to description	**(ET)** Generalizing from description to theory
Generalizing form Theoretical Statements	**(ET)** Generalizing from theory to description	**(TT)** Generalizing from concepts to theory

According to the Lee and Baskerville (2004) framework, this dissertation would use type ET generalizing: generalizing from description to theory. The action learning phase in Dialogical AR involves generalizing form description to theory where, when using the DSAR framework, any theorizing done when the designed artifact does not meet expectations involves type ET generalization. Furthermore, Lee and Baskerville (2003) provide examples of ET generalizing from several sources in the literature cited in this dissertation (Klein et al. 1999; Strauss et al. 1998; Walsham 2002; Yin 1994). Yin (1994) encourages a case study researcher to aim for analytical generalizability where the researcher makes what he calls "level two" inferences from descriptions of the case results to theory. Thus, a basis for theorizing also exists within "thick" descriptions of the case setting (Walsham 2002), such as a detailed account and analysis of researcher-practitioner reflection (Lee et al. 2003: 236). In Grounded Theory, Strauss and Corbin (1998) suggest that theory should emerge entirely from observations and descriptions. Thus, the theorizing research activities within the DSAR framework have ample support in the literature as opportunities to generalize from description to theory.

CHAPTER 5 Evidence from the Action Research Partnership

If *Reflective Practice* and design are to be considered as a philosophical, epistemological and theoretical basis for the success of agile methods in the small shop setting, then the Reflective-Agile Learning Model and Method should be supported by empirical evidence. This chapter provides a qualitative description of the evidence from the Dialogical AR Partnership and study conducted with a small web-development team in a small shop in Central Virginia, USA.

The sections of this chapter are as follows. First, a description of the Dialogical AR setting is given. Next follows is a summary of the iterations of the Dialogical AR cycle. Subsequent sections present a broad account of the evidence from the Diagnosis and Action phases of the Dialogical AR cycle.

5.1 Description of the Dialogical Action Research Setting

The practitioners in the Dialogical AR team consisted of a software development team of four practitioners in a single-site small web development company in central Virginia, USA. Lee and Baskerville (2003) suggest that interpretive field studies, such as Dialogical AR, provide in-depth, rich and "thick" descriptions from which generalizations to theory can be made. As this research outlines progress towards a method and methodology for learning in small software development teams, which is aligned with the theoretical work of Argyris and Schön (1974,

1978, 1996), and Schön (1983 and 1987), regarding the nature of organizational and individual learning, then action research is a method of inquiry well-suited to this task.

The remaining subsections on this section discuss the particulars of the Dialogical AR setting and the Dialogical AR Partnership. As a necessary condition of diagnosis, a portrait of the practitioners' extant software development methods, habits and processes is developed and illustrated. These extant methods are used as contrast for the emergent themes and issues which arose during diagnosis.

5.1.1 The Practitioners and the Practitioner Setting

For the purposes of this report, I will refer to the company and team of practitioners as "SSC"[13] and refer to the author as "the researcher." SSC consists of two software developers, Johnny and Fred, a web designer/developer, Velma, and the company owner and lead developer, Daphne. Pursuant to Internal Review Board procedures on confidentiality, the identities of all companies and individuals have been changed to provide a degree of anonymity for the practitioners who participated in the Dialogical AR effort and to also protect the identities of SSC's clients.

SSC was started by Daphne as a graphics design company in 1998 and, in the ensuing decade, has steadily transformed into an IT solutions company which focuses on custom web applications and IT services and integration, which typically involves connecting back office and

[13] In the interest of confidentiality, the names of individuals and organizations discussed in this report are all pseudonyms

ERP software with a dynamic web front end. SSC evolved toward this business model in 2005 and has steadily acquired additional employees since that time: Velma in 2005 and Johnny and Fred in 2007. All of SSC's developers have professional and/or university training: Daphne holds a Bachelor's Degree in Graphic Design and a professional certificate in Information Systems and has started coursework on an MBA; Velma has community college training in Web Design; Fred holds both a bachelor's and master's degree in Information Systems and is trained in application and web development; and Johnny holds a bachelor's degree in Information Systems and is trained in application development. The combined work experience in web and applications development across the team is approximately 40 years.

SSC is a good case for investigating the research questions posed in this dissertation for the following reasons:

- **Small Team, Small Company**: SSC is a small team working in a small-shop environment. While there are small teams within large organizations, a small team operating within a small company determine and shape their organizational culture and learning in ways that are independent of the influence of a larger organizational culture present in larger companies. Thus, organizational and individual learning may be different in a small-team setting.

- **No Professed Method**: Daphne, who is a former student of the researcher, professed a desire for a formal method suited to the nature of her business and the size of her company and team in 2005. While SSC was using some method, no external (and thus externally valid) method was in use at SSC at the onset of this research.

- **Web Development**: The literature on web development and web engineering suggests that most web development is accomplished within small teams (McDonald et al. 2001b; Pressman 1998; Reifer 2000). Thus, SSC's business model is suited to an investigation of small-team software development. With the increasing demand for web application development, a better understanding of methods for small team is warranted (Ginige et al. 2001).

Understanding Daphne's desire for a formal and valid external method, the researcher suggested the suitability of agile methods and Extreme Programming in particular. To this end, the researcher suggested the possibility of forming a Dialogical AR Partnership using the Dialogical AR method. Upon her consent, the researcher clarified and explained the practitioner's role in the researcher/practitioner partnership. With this understanding in place, the Dialogical AR team was formed and the research took place over a period of nine months from July 2008 to February 2009.

5.1.2 Description of the Dialogical Action Research Team

With the ethics of full disclosure in mind, it should be clear that each practitioner was a student in at least one of the researcher's courses at some point. Furthermore, the researcher coached two of the practitioners, Johnny and Daphne, when they, as students, were competitors in a national student software development competition.[14] During the course of this research, none of the practitioners were students in any of the researcher's courses; this should clarify any ethical questions related to power. The researcher obtained signed statements of informed consent from all practitioner participants at the onset of the research partnership and, at the onset of the research effort, further clarified each party's respective roles in the Dialogical AR team. This section will now focus on the steps taken to develop the Dialogical AR team with an emphasis placed on establishing an attitude of equality amongst the practitioner and researcher.

[14] Johnny and Daphne did not compete on the same team, nor did they compete in the same year against each other

Daphne, as the company owner and team lead, was of particular interest and focus during the Dialogical AR. Mårtensson and Lee (2004) remind the researcher who investigates phenomenon using Dialogical AR to maintain an equitable partnership such that, through dialog, the practitioner and researcher work together to explore the issues facing the practitioner. Furthermore, this arrangement must take care to accept and respect the role of social and historical context inherent within the practitioner's problems (Mårtensson et al. 2004).

Thus, it was incumbent upon the researcher to establish and elaborate on the principles of Dialogical AR as early as possible during the research effort. Toward this end, Daphne was apprised of these matters prior to the start of the research and she, once duly informed, consented to the arrangement.

5.2 Iterations in the Dialogical Action Research Process

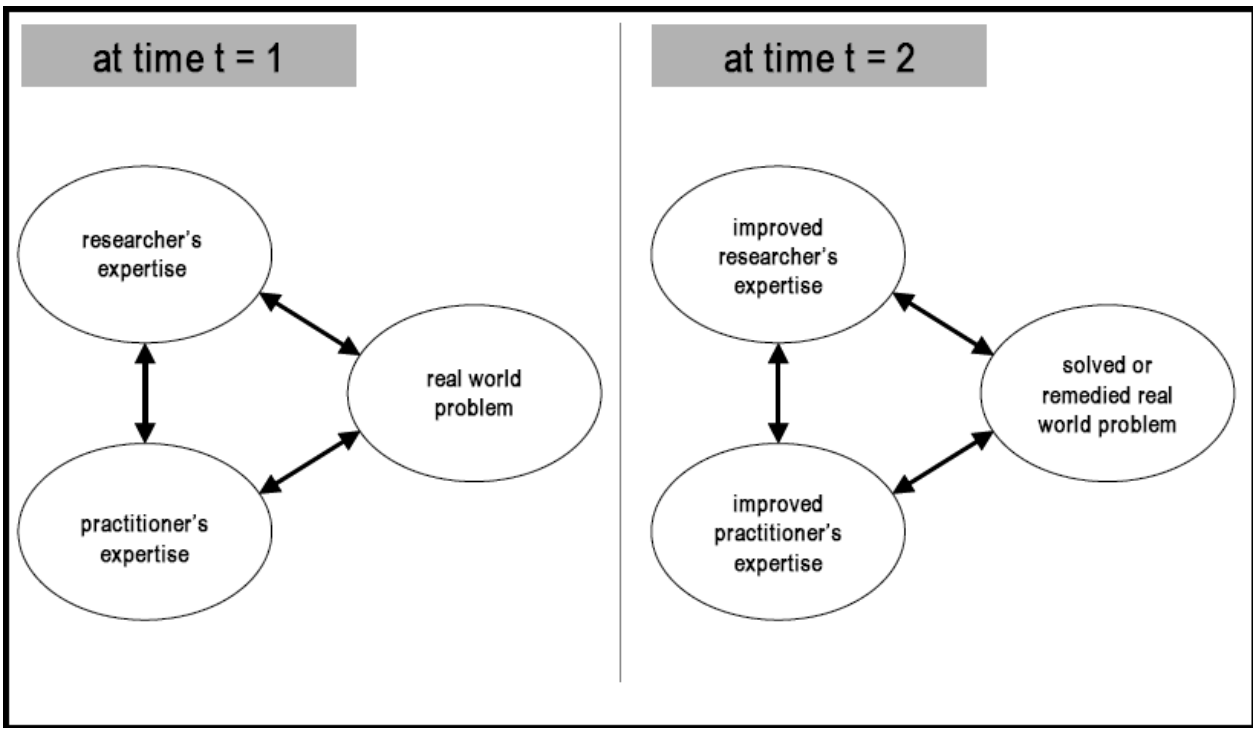
During the nine-month Dialogical AR Partnership, the researcher worked regularly with the practitioners in their everyday setting and, at times, met with the practitioners outside of their everyday setting. In this section, a chronological account of the iterations of the Dialogical AR cycle is given. Next, a rationale for grouping the deep and descriptive details of the evidence into the steps of the Dialogical AR cycle is discussed.

5.2.1 Timeline of the Dialogical Action Research Iterations

Several Dialogical AR cycles were undertaken during the course of the Dialogical AR Partnership (see Figure 24). Most of these cycles lasted for about a month although the duration varied with some cycles lasting a month and some cycles lasting a week or two. The initial

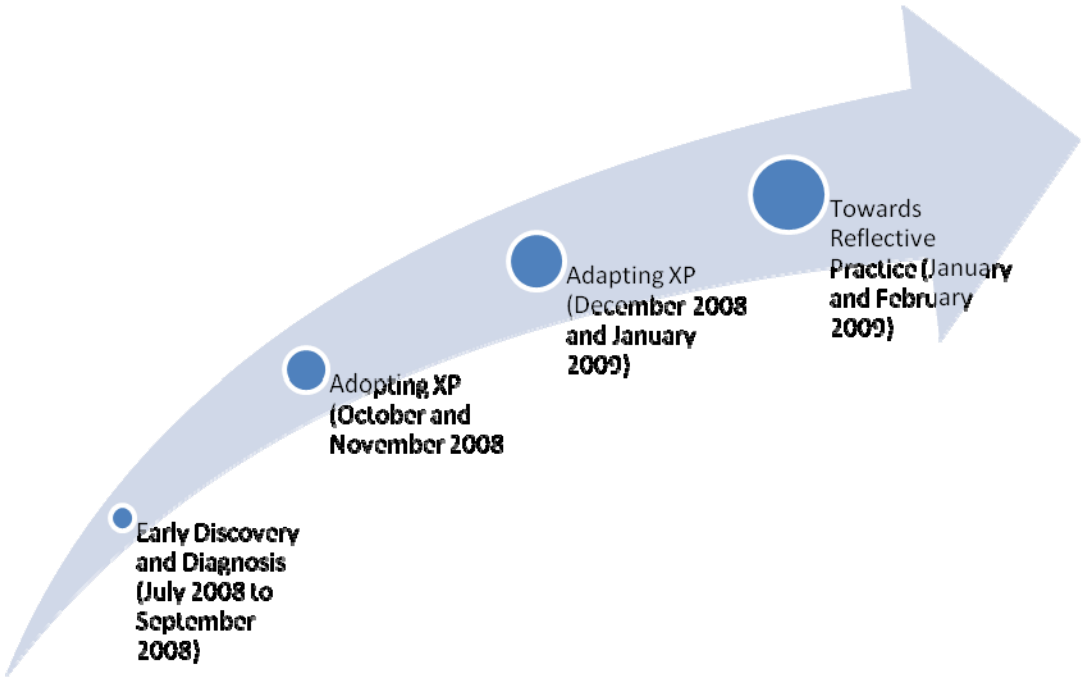
diagnosis phase, lasting for about two months, was the longest phase. Among the goals of the diagnosis phase was to develop a sketch of the company, the team, the practitioners and the extant methods in place. Subsequent phases of the Dialogical AR cycle use the outcomes of the previous cycle as inputs to the current cycle.

Figure 31 Iterative Improvements in Expertise (Mårtensson and Lee 2004)



In Figure 31, Mårtensson and Lee (2004) demonstrate the emergence achieved overtime as the researcher and practitioner explore the problem space over time. The following sub sections organize the overall timeline of the Dialogical AR cycles over the course of the study. Figure 32 shows a timeline of the four major Dialogical AR cycles.

Figure 32 Timeline of the Dialogical AR Iterations



5.2.1.1 Discovery and Diagnosis in the Early Period

The initial diagnosis phase of this research can arguably be traced back several years to discussions between the researcher and Daphne as she started her own business. Thus, the researcher has been somewhat privy to some of the issues Daphne faced at the time she decided to start a web software development company. Given the informal nature of these discussions, taken in the natural attitude of everyday life, these discussions were not included as evidence.

In July of 2008, shortly after receiving Internal Review Board approval, the researcher entered into the Dialogical AR Setting. During this month the researcher began collecting evidence via observations during SSC's weekly company meetings. The researcher also observed the practitioners' daily routines and consulted with Daphne as questions arose

concerning the nature of SSC's work. The field notes from this period are included in the evidence, and analysis thereof, none of the discussions or meetings during this period were recorded or transcribed. Thus, the activities of July 2008 were devoted to developing a profile of SSC and the practitioners on the team.

During August and September of 2008, Daphne and the researcher began their first set of recorded dialogs. These dialogs were transcribed and coded to promote a richer and fuller understanding of the social and historical context of SSC and their extant methods came into focus. These initial diagnoses and subsequent plans for action found both the researcher and practitioner in agreement that an agile method would be worth consideration. Moving forward, the researcher presented Extreme Programming (XP) as the most viable choice for SSC. Thus the action plan was to adopt XP.

5.2.1.2 Adopting Extreme Programming

By the end of September 2008, the researcher had introduced the basics of the agile philosophy in general and the major mechanics of XP. While the researcher had no prior experience in teaching XP, the researcher did have adequate undergraduate teaching experience in software development methods. The researcher refrained, as much as possible, from an overly didactic and pedantic approach as he taught the practitioners XP using presentations, dialogs, observations and retrospective/feedback sessions. Remarkably, as the practitioners' immediately perceived benefit in the "analysis" phases of XP - *User Stories* and *Architectural Spikes* - the practitioners did not at this stage, or any subsequent stage, actually undertake any formal training in the use of XP. This raises questions and has several potential implications for the

practitioners' attitudes towards learning which will be discussed in the analysis and discussion portion of this report.

By October 2008, the researcher was intensively recording and transcribing multiple dialogs in addition to copious field notes as the activities of the Dialogical AR Partnership were in full swing. Many of the major themes for subsequent interpretation, analysis and learning specification emerged during this period. During the period of October 2008 to November 2008 the practitioners progressed to the *Release Planning* and *Iteration* activities of the XP method. It is also apparent to the researcher that the practitioners are unable to fully adopt XP in an orthodox manner due to external issues with strategic partnerships and internal issues related to the social and historical context of the team. During this cycle, the researcher sees early evidence that Argyris and Schön's (1974, 1978, 1996) and Schön's (1983, 1987), theories on learning and reflection have bearing in this case. Also at this time, the practitioners report early and palpable successes where productivity, expressed as billable hours, had markedly increased due to their use of *User Stories*. By the end of November 2008, the pattern by which the practitioners had adopted and adapted XP are readily apparent; and it was conclusively clear that they would adapt XP rather than adopt it outright.

5.2.1.3 Adapting Extreme Programming

By late November 2008 and into December 2008, most of the XP method had been introduced to varying degrees of success. While the method's authors clearly state that piecemeal adoption can be okay (Beck 1999), some of the essential philosophical, and thus methodological, tenets of XP are not smoothly adopted in this case. Issues related to individual and team learning

had also surfaced and would serve as the basis for further diagnosis. Many practices endemic to Daphne's original and informal method thwarted total adoption of XP. Thus, the SSC practitioners adapted the XP method to create a blend of old and new. None-the-less, by the middle of December, SSC had tried the majority of the XP and their feedback was very positive overall; the practitioners have realized tangible benefits in their adapted form of XP.

Along with the matters of trying and using XP, a sizable portion of the dialogs were devoted to internal and external issues not seemingly related directly to the use of XP. By this point, the majority of the dialogs have been one-on-one with Daphne and the remaining dialogs consisted of various combinations of Daphne, Fred, Johnny and Velma. There is increasing evidence, somewhat grounded in the team's internal and external problems, that the epistemology of *Reflective Practice* and the concepts of Theory of Action and the Learning Organization are applicable. By this period, the researcher has described the basic outline of ideas within these theories and had explained the applicability of these theoretical perspectives to their problems. It was also apparent in mid-December that a further, and last, iteration involving interventions based on these theories, was warranted. These interventions, and the adaptations of XP, became the basis for the design science artifact discussed in the penultimate chapter of this report.

5.2.1.4 Achieving Reflective Practice

From mid-December 2008 until February 2009, the practitioners considered a number of theory-influenced interventions related to learning and reflective practice. It was also during this period that internal and external issues, mostly related to power, communication and team

dynamics had come to a head. Ongoing and simmering, many of these issues are related to one of SSC's strategic partnerships. By January 2009, the practitioners were in crisis mode due to problems which slowly mounted over the duration of the study. These issues were apparent to the practitioners at the onset of our work together, but the root causes of these problems are not directly addressed in any apparent manner. At this time, the researcher noted that some of the most important and theory-based interventions, were also the least considered and tried by the practitioners. The practitioners espoused a serious consideration of the theory-based interventions, but seemed to lack the time required to strongly consider any potential benefits.

Also, during this time, the researcher received advice that it may have been wiser to introduce the theory-based interventions in a new Dialogical AR Partnership; this counsel is being considered for future research. By the end of February 2009, the practitioners provided feedback which indicated that they had tried the theory-based interventions sufficiently for the purpose of completing the last iteration and exiting the Dialogical AR Partnership. In research, failures can also produce learning and, while it is inaccurate to characterize the outcomes of the last iteration as a failure, the researcher looks forward to trying interventions based on the designed artifact in a new Dialogical AR Partnership.

5.2.2 Reducing the Data: Synthesizing the Dialogical Action Research Iterations

The preceding sections offered a brief synopsis and timeline of the Dialogical AR iterations. Subsequent sections provide richer and detailed descriptions of the experiences Dialogical AR Partnership. The evidence from the dialogs is used to amplify and illuminate the outcomes and issues from the diagnoses, interventions and evaluations. The evidence provides

support for the effectiveness of the theory-based interventions and also supports the designed artifact discussed in the penultimate chapter of this report.

There are several motivations for providing a deeper description of the evidence synthesized into themes rather than a linear and chronological account. These longer descriptions focus on the details of the practitioners' natural setting and a basis for an interpretation of the practitioners' meanings (and the first- and second-level constructs derived from these meanings). A longer description provides for a holistic approach to convey rich interpretations, and underscores the emergent and iterative nature of the theoretical lens applied (Creswell 2009; Wolcott 2009). The sections that follow in this chapter divide the descriptions and analyses along the principle phases of Canonical AR.

5.3 Diagnosis

Diagnosis is a critical phase of Dialogical AR which steers and shapes all subsequent phases. Diagnosis is also the entry-point into the Researcher-Practitioner partnership and an opportunity for arising and reflection. The goals set forth for the Dialogical AR process are also subject to change and reconsideration during the diagnosis phase (Avison et al. 1999:96). Thus, the working hypotheses and nascent propositions gleaned from reflection and self-interpretation in the diagnosis phase stand to set the tone for the remainder of the Dialogical AR study (Baskerville 1999:15).

Here, at the entrance to what Davison et al. (2004) call the Cyclical Process Model (CPM) of action research, the diagnosis phase is an early test of the Researcher-Client Agreement (RCA). In Dialogical AR, the practitioner must be informed of the respective roles

in the researcher-practitioner partnership and be reassured of the researcher's commitment to privacy, trust and ethical behavior in the research setting. The diagnosis phase provides the researcher with the problems and opportunities for empirical observation and interpretation which frames the relevance of the study. Conversely, early patterns of inquiry, observation and data collection establish rigor in process of analysis, adding validity to the research.

The remaining subsections proceed as follows: first, there is a discussion on focus and rigor in the research process and the use of the coding techniques of Grounded Theory as a mode of analysis; next, accounts for the use of Computer-Assisted Qualitative Data Analysis Software (CAQDAS) during the collection and analysis of the evidence; the last subsection focuses on the themes and problems which arose during the diagnosis phases of the iterations.

5.3.1 Directions and Rigor from Coding Dialogical and Observational Data

A systematic process of coding qualitative data, both dialogical and observational, provided for a rigorous collection, classification and analysis of the evidence. According to Strauss and Corbin (1996), coding qualitative data provides "standardization and rigor to the process… (p.13)" of analyzing qualitative data such that the interplay between the researcher and the data is facilitated. A systematic coding scheme allows the researcher to "see" patterns and directions in the data which provide guidance for subsequent data collection and analysis. While rigor and relevance debates are timeless (Agarwal et al. 2005; Baskerville et al. 2004; Benbasat et al. 1999; Lee 1999; Lee et al. 2008), the presence of methodological and analytical of rigor separates controlled and purposeful scientific inquiry from conjecture. To wit:

> *Without rigor, research is worthless, becomes fiction, and loses its utility. Hence, a great deal of attention is applied to reliability and validity in all research methods. Challenges*

> *to rigor in qualitative inquiry interestingly paralleled the blossoming of statistical packages and the development of computing systems in quantitative research. Simultaneously, lacking the certainty of hard numbers and p values, qualitative inquiry expressed a crisis of confidence from both inside and outside the field. Rather than explicating how rigor was attained in qualitative inquiry, a number of leading qualitative researchers argued that reliability and validity were terms pertaining to the quantitative paradigm and were not pertinent to qualitative inquiry. (Morse et al. 2002:2)*

Thus, to ignore or argue away the importance of rigor in any scientific inquiry is to have selected a false choice. This study relies on rigor inherent in Dialogical AR and supplements this rigor with the coding techniques of Grounded Theory as the literature on action research is not always explicit with respect to analytical approaches (Avison et al. 1999; Baskerville 1997; Baskerville 1999; Checkland et al. 1998; Lau 1999; Mårtensson et al. 2004). This research accepts a responsibility to demonstrate the use of rigorous analysis procedures, beyond the mechanisms inherent within action research, so as to minimize ambiguity in this regard for the reader. The next section describes how Grounded Theory, as described by Strauss and Corbin (1996), provides a rigorous mode of analysis for qualitative evidence form this Dialogical AR study.

5.3.1.1 Grounded Theory as a Mode of Analysis

Dialogical AR is grounded in the epistemology of phenomenology (Mårtensson et al. 2004). As such, possible approaches for the analysis of qualitative evidence are (but not limited to) hermeneutics, semiotics, narrative, metaphor and other similar approaches. As qualitative evidence consists of text and text-objects, this evidence requires analytical techniques suited to text rather than those suited to quantification. This difference is not an occasion to forego a systematic analytic approach, but does call for an analytic approach appropriate for textual data.

Grounded Theory is an approach to qualitative research which holds that theories grounded in data are inherently relevant and useful. Thus, Grounded Theory "grounds" theories

which arise from the application of its techniques "in" the data. In many cases, Grounded Theory inductive assumptions not suited to this research; thus, any allusion that this research is producing a "grounded theory" are considered incorrect as this research is not Grounded Theory nor does it claim to product a theory, grounded or otherwise. Furthermore, Grounded Theory itself is somewhat mired in a dogmatic morass due to a schism in epistemology among its original authors. Apart from this controversy, there is utility in Grounded Theory as a mode of analysis for qualitative data.

Analytical techniques of Grounded Theory, such as open, selective and axial coding, constant comparison, and memoing, are all valid and useful techniques for rigorous analysis of qualitative data. As such, this research appropriates some, but not all, of the qualitative data analysis techniques described by Strauss and Corbin (1996). Some advice embedded in the doctrine of Grounded Theory, suggests that interviews should not be recorded and that no literature review should be undertaken prior to research. This research effort has not heeded that advice. Some variants of action research suggest an approach to diagnosis which is consistent with Grounded Theory's position on literature consultation prior to entering the practitioner setting. In the case of this research, the literature, and thus theoretical guidance, was consulted prior to entering the researcher-practitioner partnership. For this study, the nature of the research and research questions suggested that it was appropriate to consider theory *a priori*. This deviation alone would be cause to question the authenticity of this research as Grounded Theory, which it does not claim to be.

As it is established that Grounded Theory as the mode of analysis for the qualitative evidence in this study, a brief description of the elements Grounded Theory used is now given. Strauss and Corbin (1996) suggest that a researcher start off with a ***microanalysis*** of their data,

which is a detailed line-by-line analysis where the researcher engages in ***open coding*** and ***axial coding***. With ***open coding***, the researcher is conceptually classifying words, phrases and passages of qualitative data looking for concepts and categories. As opportunities for new coding arise, the researcher undertakes a process of ***constant comparison*** to determine how new concepts relate to existing concepts. Along the way, the researcher makes broader hypothesis-building and theorizing notes in a process Strauss and Corbin call ***memoing*** (p. 110). Through constant comparison, memoing and open coding, the researcher constructs a number of categories and sub-categories of codes. ***Axial coding*** is the process by which the researcher relates and "aligns" codes along the axis of a category. As the researcher's hypotheses, theorizing and propositions begin to take shape, the research engages in ***selective coding*** where codes are grouped to support emerging theoretical themes. There are additional techniques, such as theoretical sampling, which are salient to the construction of a Grounded Theory, but these additional techniques offer diminishing returns given the goals of this research. The specification of learning, is a requisite outcome most forms of action research, is sufficiently supported by the analytical use of Grounded Theory techniques as they assist in the interpretations which second level constructs and a descriptive basis for identifying espoused theories and theories-in-use.

This sub-section has discussed the importance of rigor in qualitative research and has described the manner in which several techniques in Grounded Theory are used as a mode of analysis for this study. The next sub-section discusses how a Computer-Assisted Qualitative Data Analysis Software (CAQDAS) package facilitated systematic, explicit and rigorous research process (Kelle 1997:17).

5.3.1.2 Computer-Assisted Qualitative Data Analysis

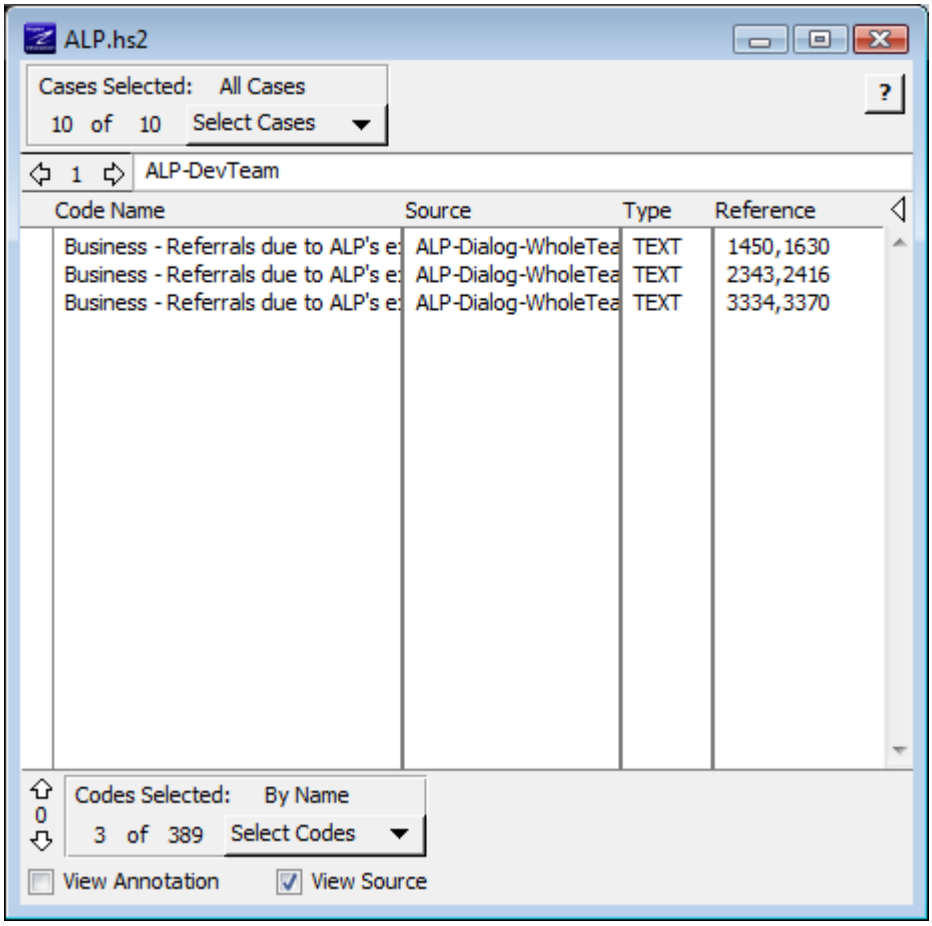
Dialogical AR calls on the researcher to adopt the scientific attitude when studying real-life phenomenon in a natural setting. While the expectations and limitations of the researcher/practitioner partnership are well described in the literature, the boundaries between consultancy and research must be demarcated by constant vigilance on the part of the researcher. Thus, frequent review of and reflection on the evidence - field notes, dialog transcriptions, etc. - is necessary to keep the goals of the research in focus. Toward this end, HyperRESEARCH, a Computer-Assisted Qualitative Data Analysis Software (CAQDAS) package, was used to code and analyze transcripts of the dialogs in order to establish "directionality" and to ground the interpretive analysis (Hesse-Biber et al. 1991; Lee 1995). The *diagnosing*, *action-planning* and *specifying learning* stages of the Dialogical AR cycle were supported by HyperRESEARCH. During the nine (9) months of fieldwork, the researcher was present onsite in the practitioner setting on an average twice per week. Each visit usually lasted for a period of 1-4 hours. The activities of the Dialogical AR Partnership typically consisted of dialogs, instruction/lecture and team-building. In this case, the principle sources of qualitative evidence from the Dialogical AR Partnership are:

- 26 recorded and transcribed dialogs. These recordings were averaged 1 hour in length. Some dialogs are with the company owner and lead developer, Daphne, and others are with various combinations of the team based on progress in a given iteration of the Dialogical AR cycle.
- Internal SSC documents
- Field notes taken while observing the practitioners' work

The researcher observed the developers at SSC work in their natural setting and also observed the practitioners' interactions with their clients in the client's natural setting.

HyperRESEARCH is well-suited to the analysis of qualitative data in the mode of Grounded Theory. HyperRESEARCH facilitated open, axial and selective coding of the transcripts and field data by utilizing the case as a unit of analysis. Figure 33 shows the study window in HyperRESEARCH, which organizes all of the cases and codes associated with each case. Open and axial coding are possible by using the Code List Editor in HyperRESEARCH to assign and relate codes to sources of data, which can be text data or audio-visual media. Over time, it is important to categorize and summarize codes as the body of qualitative evidence expands. Figure 34 shows the Code List Editor and Figure 35 shows the Source Window.

Figure 33 HyperRESEARCH - The Study Window



Thus, the case is the means for collecting, comparing and selecting codes to assist in developing interpretations and second-level constructs from the dialog evidence. In focusing on the case, HyperRESEARCH allows for the aggregation of codes which are ascribed to people, concepts or any singular entity or theme.

For this study, the cases were delineated by combinations of the participants in the dialogs. Thus, the cases in this study were:

- Fred
- Fred and Johnny
- Fred, Johnny and Velma
- Fred, Johnny and Daphne
- Fred, Daphne and IMS (a client)

- Johnny and Velma
- Johnny and Daphne
- Daphne
- Whole Team

As interactions between practitioners during course of a dialog were important in developing historical and social context, the particular combinations of practitioners in any given dialog served as the basis for case selection. Across these cases there were 262 open codes which, through axial and selective coding, and memoing were reduced to 28 categories. There were 3441 occurrences of all codes across all cases. Table 28 shows the number of codes which contribute to each category and how many code occurrences were associated with each category.

Figure 34 HyperRESEARCH - Code List Editor

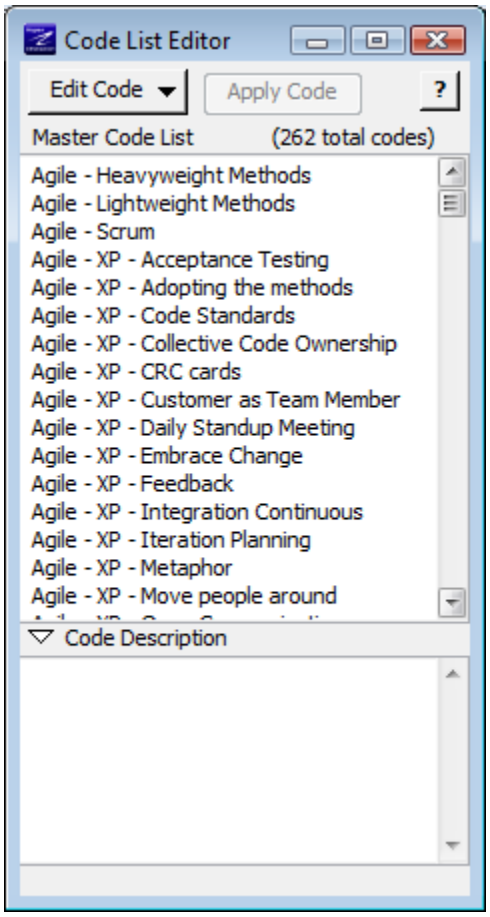


Figure 35 HyperRESEARCH - Source Window

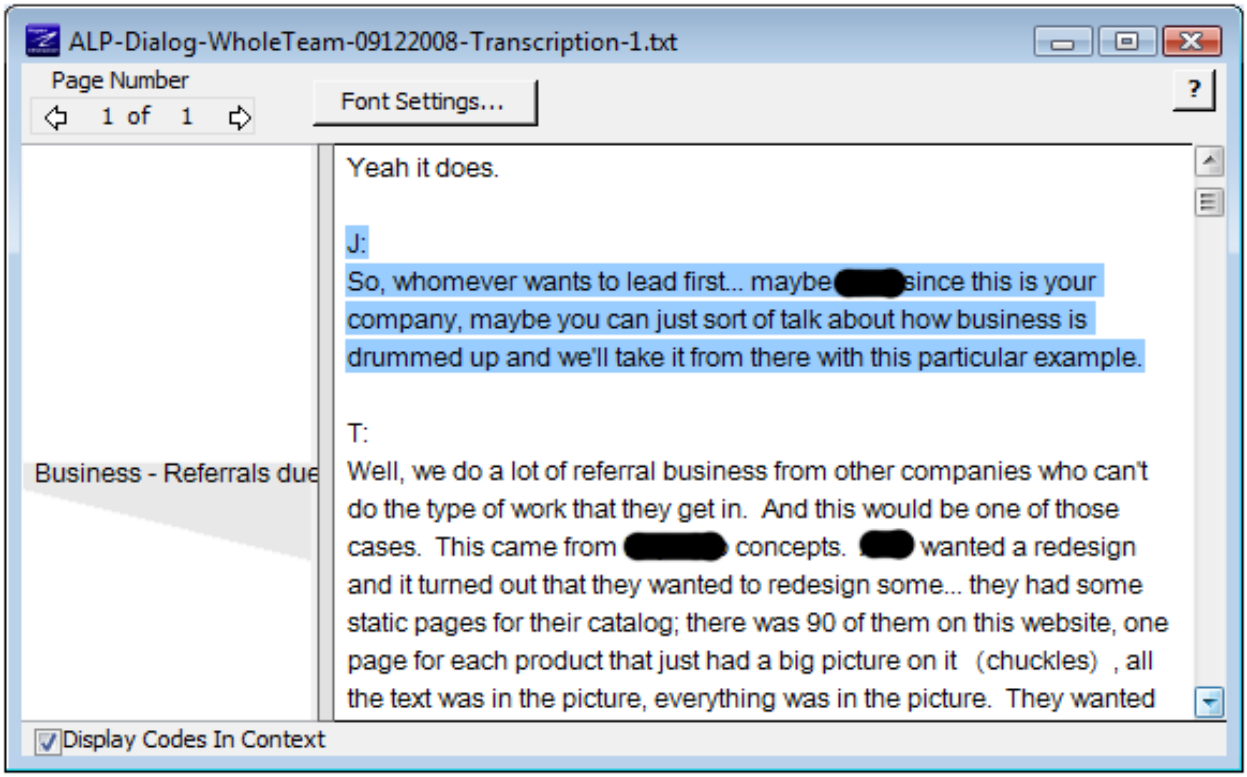


Table 28 Code, Category and Occurrences from the Dialogical Evidence by Category

Codes	Category	Occurrences
39	Agile	1198
33	Clients	491
27	Team	264
21	Leadership	219
18	Project	147
17	Web Technologies	74
12	Individual	76
10	Learning	120
9	Business	148
8	Design	44
7	Company	111
7	Dialogical Action Research	42
6	Development	81
5	Methods	32
5	Misc	11
4	Education	30
4	Professional	34

4	Reflective-Agile	68
4	Skills	72
4	Web Development	23
3	Craft and Creativity	14
3	Knowledge	8
3	Reflective Practice	51
3	Web Projects	4
2	Gender	1
2	Phenomenology	2
1	Pride	16
1	Quality	37

A more instructive and useful guide for problem identification and interpretation in the diagnosis phase is to examine the frequency of occurrence in the codes constituting each category as shown in Table 29.

Table 29 Code, Category and Occurrences from the Dialogical Evidence by Occurrences

Codes	Category	Occurrences
39	Agile	1198
33	Clients	491
27	Team	264
21	Leadership	219
9	Business	148
18	Project	147
10	Learning	120
7	Company	111
6	Development	81
12	Individual	76
17	Web Technologies	74
4	Skills	72
4	Reflective-Agile	68
3	Reflective Practice	51
8	Design	44
7	Dialogical Action Research	42
1	Quality	37
4	Professional	34
5	Methods	32

4	Education	30
4	Web Development	23
1	Pride	16
3	Craft and Creativity	14
5	Misc	11
3	Knowledge	8
3	Web Projects	4
2	Phenomenology	2
2	Gender	1

Table 29 is far more revealing concerning the nature of the dialogs and which issues emerged during diagnosis. For instance, the *Clients*, *Team* and *Leadership* categories associate with a number of the issues which arose during diagnosis and constitute a much greater share of the issues which arose in the middle and latter phases of the Dialogical AR cycle. The *Business*, *Learning* and *Company* codes also "cluster" and constitute a set of problems which also steadily increased throughout the study. Grounded Research techniques for analyzing interpretive data from the dialogs allowed for these codes and categories to emerge throughout the study using a consistent process of open, axial and selective coding, memoing and constant comparison.

HyperRESEARCH facilitated the diagnosis and evaluation phases of the Dialogical AR process. In order to convey the principle issues are and to produce a rich account of the researcher-practitioner partnership, these codes and cross-comparisons of these codes are used to select the relevant passages of the dialog used in the descriptions in this chapter. Care was taken to consider the espoused theories of actions embedded in the dialog as second-level constructs were developed regarding the *Theory-in-use* of action (Argyris et al. 1996). Thus, the use of HyperRESEARCH, enabled a rigorous analytical process from which second-level constructs were developed which provide subsequent support to the specification of learning for practice and the body of knowledge on *Reflective Practice* and agile methods.

While HyperRESEARCH is quite compatible and somewhat predisposed towards Grounded Theory, the use of this software is also conducive to adopting Hermeneutics as a mode of analysis. The Grounded Theory techniques of constant comparison and open/axial coding facilitate whole-part understanding of texts in a manner which is consistent with the Hermeneutic circle. Thus, it could be argued that the iterative processes of Hermeneutic analysis would have also produced a rich basis for interpretive analysis. Even as this research is concerned with matters of design, the very act of design, particularly with respect to reflective practice, is concerned with whole-part reconciliation through the conversation with the situation that the designer engages in (Schön 1983:79). Thus, an iterative process of listening and responding to *back-talk* in the situation is compared to the whole. This relates to the use of CAQDAS for interpretive analysis as either Grounded Theory of Hermeneutics would have shown promise as a theoretical lens under which a systematic analysis of the data could be pursued.

5.3.2 Grounding the Practitioners' Historical and Social Context

The evidence from diagnosis in the first iteration developed a profile of SSC as "found" by the researcher and developed a picture of the practitioners' initial concerns, needs and desires (their "ailments"). This effort also entailed developing a biographical sketch of the company, of the team and of Daphne. During this period Daphne related her history and the history of the company and how she came to start her own business making custom web applications and web sites.

Daphne started the company in 1998 as an outlet for her undergraduate training in Graphic Design. Given her earliest career aspirations in illustration, SSC's initial business

model was the creation and distribution of hand-illustrated greeting cards with a historic focus and theme. At this stage, SSC was a part-time endeavor Daphne ran out of her home. As Daphne possessed a gift and talent for illustration and the creative spirit that comes with such a gift, she soon realized the possibilities of doing business on the web and designing for the web. Customers for her greeting cards began to inquire about her ability to create websites:

> *That's when I started getting the question all the time: "Do you build a web page? Do you know how to build a web page?" So, my first web page, like I said, I built for myself. In 1998, we formed the company; I already had the web page up for the Christmas cards. We went ahead and formed the company and advertised and sold the Christmas cards that year.*
>
> *…I had been getting the question about web pages and very quickly realized that there was a lot more to it than just putting up a pretty face, which was the graphic design aspect. I actually started at [a local community college] (sic)… they had a 6-month continuation education program for web programming. They had two tracks: one for web design and one for web programming. I wasn't sure which one was going to best suit me at the time and I ended up following the web programming track…*

For Daphne, the shift to custom web applications and web sites was somewhat accidental; however she was clearly taken with this career shift.

Daphne steadily grew her business using means which are typical in small business growth: word-of-mouth, referrals and what Daphne terms as "website rescue." As is usually the case, Daphne's road to small-business success was hard-fought:

> *I had formed a company and I realized that my eventual goal was to go into business for myself at that point. But, I didn't have... I was pretty young and didn't have the knowledge and experience.*

Recognizing her inexperience and significant technical skill required to develop custom websites, Daphne sought to improve her skills by returning to school:

> *And I had to pay for it out-of-pocket so there was that involved too. I had actually decided "well, this is good stuff but I feel like I've only scratched the surface, I feel like*

there's so much more that could be done." So I had enrolled in [A local community college] and started taking programming courses.

In full disclosure, it was not long after this period, in 2004 and 2005, that Daphne was a student in a few of the researcher's application development courses. The researcher's involvement in Daphne's education continued as he coached her team in a national software development competition sponsored by a major IT Services and Software vendor in 2005 and 2006. In the interim, as Daphne sought to improve her skills to the end of running her own web software development company, Daphne worked fulltime creating an eCommerce and ERP infrastructure for, TTS, a retailer of specialty products in Central Virginia. In 2005, Daphne resigned from her fulltime job to focus on growing her business as a fulltime endeavor. She left her job with TTS despite having been offered partnership and a stake in the company. Daphne soon realized, more than ever, that she needed to let her propensity to give more than a 100% of her effort work for her rather than for somebody else:

And the other thing that, you know, is true about that... I mean, you put in all that effort for somebody and work really for that and, like I said, they just keep coming back with, you know, what more can you do? I got the feeling that they didn't feel like I was always giving it 100% and that I had more to give and I'm like, when I'm here, I do.

In fact, in the end, Daphne felt that her talent was not appreciated as she had important decisions overridden by management who had hired her initially to manage technical aspects of their operations. Furthermore, hiring decisions she made were questioned for reasons Daphne felt to be unethical:

I had the opportunity to interview some people and they told me "well, this is your thing, you interview people, you pick somebody" and so I was given power to do the interviews and when it came to hiring people they were prepared to override my choice.

I understand they have the right to override my choice - they are the owners and they have to pay the bills, but their reasons, in my estimation, were unethical...

One applicant was [singled out and discriminated against] and I thought:' I like her and I think she can do the job.' That really bothered me...

I thought: "but I like her, I want to hire her" and they would not let me do it. So I said: "okay, alright, I see just about how much power I have around here..." I got to thinking about what the arrangement would be like because... what the contract would look like if they ever agreed to sell the company to me... I'm like: "You know that's going to have so many restrictions and provisos and..." If they're willing to do this kind of thing unethically, I'm a little nervous about pursuing that so at that point I gave my notice and everybody was floored.

Daphne brought with her a small-but-growing client list from past part-time endeavors, and the promising interviewee her former employer refused to hire, Velma, and struck out as an IT Services and custom web application development shop in 2005. Six months into pursuing her business fulltime, Daphne came into a series of what she calls "strategic partnerships" which persist until this day. Her first partnership was with an advertising and marketing agency, MNM, which contacted her by way of her reputation for doing "website rescue." Daphne characterizes website rescue thusly:

I don't know how many times I hear when I first sit down with the client - "you know we had another web person and it just didn't work out" or "they promised us this and they couldn't get it to work." I have taken over so many projects from that level. They just seem amazed when we actually deliver something (she laughs). We also do, we believe in the "full-range approach" when I sit with a client, because we can do everything, I am able to sit and listen to their wildest dreams and even though we may not be able to do it now, we can start with... we know that where we start we can always take them where they want to go, if they want to get there. And I think that's missing from my field because everyone thinks that what I do is easy. There is a perception out there that building web pages is easy to do because there's a lot of things on line where you can build your own web page and have it up in 10 minutes.

So from blank page all the way to custom application development, we can do that. Because of that, I think we fill a real need in the market because I'd say that 80% of the work that we do is work that other people couldn't finish or had to pass off because it was beyond their ability.

It should be apparent that her reputation for website rescue was also backed up by a strong "can do" attitude and a commitment to ethical practice and a high degree of quality in her work. Through her growing reputation, Daphne expanded her opportunities by joining Business

Networking International. Her participation in this group allowed Daphne to pick up her second strategic partnership with KWC. Each of her strategic partners would typically hand her jobs that were outside of their own scope and expertise, thus creating a symbiotic relationship for both parties. MNM, realizing her talent and potential, made an early overture to subsume Daphne's nascent business into their own; however, given her experiences with TTS, she declined.

Furthermore, Daphne, as evidenced by her return to school to improve her skills, recognized the rapid change pervasive in custom web site application development and recognized the need to stay on top of and ahead of technological change. This issue would be a recurring theme throughout our dialogs:

> *…everybody realized they needed a graphic designer to build a real professional web presence. When style sheets came out, and things like that, we were able to deliver a much more professional look and those are all design. We had a lot of control, imagery, Flash came on the scene, which was its own problem because everybody thought it was the best thing since sliced bread and over-used it. But from there the next 4 or 5 years was a cycle of "well, now what can we do?" and that's where programming came in and now you require... a lot of the big companies have a staff of graphic of designers and a staff of computer programmers and the two don't ever talk to each other. There's some kind of enigmatic connection where the graphic designer does this design and hands it off to the programmer and they're supposed to make it work exactly like what the client wants. That carries with it another set of problems because your computer programmers are not graphic designers and they don't know how to do that stuff and not only is there a lot of overhead in an organization like that but delivering what the client wants is a very long and suffering and painful process.*

With a background in graphic design already in-hand, Daphne realized that she could deliver a superior product if she coupled her design skills with development skills. In 2005 she completed a certificate in Application Development in the Information Systems program of a major state university in Central Virginia.

At the time of our initial diagnosis dialogs, in the summer of 2008, Daphne had indicated that her business had doubled every year since 2005. Thus, by the end of 2006 and into 2007,

Daphne and Velma had more work than they could handle. In casual conversation in 2005, Daphne mentioned to the researcher her growing need for help and her desire for methodological guidance. By mid-2007, Daphne hired Johnny for part-time work. Johnny is also a former student of the researcher and a recent graduate from the same undergraduate program in Application Development in which Daphne had completed a post-graduate certificate.

> *Yeah, I was actually looking for somebody part-time and Johnny had a part-time internship that was going carry through the end of December, or December 2007, so the end of the year. So it was a very good arrangement.*

With Johnny helping on smaller projects, Daphne was able to stretch out into larger projects such as eCommerce implementations and eCommerce and ERP integration. This growth in project size and complexity precipitated the need for another developer, but at this stage Daphne wanted a developer with experience who was more up to her level of 10+ years of experience. In a stroke of serendipity and coincidence, Daphne placed an advertisement on craigslist (a Web-based advertisement exchange) and found Fred, who was a graduate of the master's program in Information Systems in the same department and university she had attended in Central Virginia. It also happens that Fred had taken a course in .NET application Development with the researcher, thus making 3 out of 4 developers at SSC former students of the researcher. Again, during the course of this study, none of the developers at SSC was a student in any of researcher's courses nor were they enrolled in any class or degree-granting program at the institution where the researcher was an instructor.

From an early stage, Daphne had settled on using Microsoft's "Classic" ASP as her development environment of choice and, in the early 2000s, comfortably made the transition to Microsoft's ASP.NET. Daphne's coursework in her certificate program and her involvement in software competitions also influenced her choice of Microsoft's ASP.NET. Fred had

coursework in .NET and ASP.NET, and a desire to learn more, making Fred a good fit for the company. Furthermore, Fred came from a medium-sized shop, which also made Fred's experience valuable to Daphne, who was eager for any methodological advice which would streamline her processes and increase her productivity.

The software development team, as the researcher found it in the summer of 2008, had been in place since late 2007. Throughout the course of this research team consisted of the following practitioners: Daphne, Velma, Johnny and Fred. This was the practitioner team that the findings and diagnoses of the Dialogical AR process were drawn from. The next subsection moves forward to the list of needs and concerns that arose during the initial diagnosis dialogs in the summer of 2008.

5.3.3 A List of Concerns from the Initial Diagnosis

The seed which spawned this research effort can be traced back to a casual conversation between Daphne and the researcher in 2005. At that time, Daphne was confident in her processes and the quality they produced, but she wanted the external validation and verification that an extant software development method or methods could provide. The researcher had an active interest in agile software development methods in practice and in academic literature had previously suggested these lightweight approaches to Daphne. Some three years later, the researcher-practitioner partnership had formed in pursuit of a method for SSC and for an artifact which might demonstrate a theoretical basis for agile success in the small-team setting for the researcher.

Despite the importance of entering into the client-researcher infrastructure of Dialogical AR with a theoretical and epistemological perspective, it is equally important to address the practitioners' needs as they are revealed and discovered through the Dialogical AR Partnership and through dialogical discourse within this partnership. Much of this initial diagnosis transpired in the period of August 2008 through September 2008. A summary of these initial themes and issues is given in Table 30.

Table 30 List of Concerns from the Initial Diagnosis

Diagnosis	Researcher's Interpretive Observations
1. "Every project is unique, how can that be efficient?"	SSC is looking for repeatable and successful patterns in the face of constant change, uncertainty and novelty inherent in creating the tailored and customer website that their business thrives on. Daphne shies away from all-encompassing and template "cookie-cutter" frameworks as they aren't easily transferred when a client changes hosting. However, customizing is costly if not done with standards in mind and Daphne believes that a methodology will ensure consistency in their process. Thus, Daphne wants to follow a patterned process for repeatable results while retaining her ability to deliver custom websites
2. "How do we know if we are delivering consistent quality when every project is unique?"	Daphne is fastidiously concerned with and pre-occupied with quality, she considers that she goes above-beyond at all times and insists that this quality is reflected in her work.
3. "How do we prove we are doing quality work?"	Among the motivators of this question is: ""I wish I had a method that ensured a good product in a reliable manner." Thus, Daphne desires in a method, reinforcement in the eyes of others that she is following a sound process that ensures high quality.
4. "How do you sell a customized and tailored website and justify the expense?"	Daphne has found it hard at times to convince potential clients and strategic partners that her work is worth the premium she charges for it. The researcher relays several analogous examples where contemporary artisans and craftspersons unabashedly command top price for high quality. Thus, Daphne desires a process which demonstrates and justifies her quality work.
5. "How can I 'productize' some of my work?"	In what Daphne describes as "productization," Daphne is searching for opportunities for modularity and reuse in her code and processes.
6. "I want to adopt team standards and norms in order to develop a team style."	Daphne feels that 110% of the work ethic at SSC comes from her. She often feels the burden of being the boss and she feels that employees will produce sub-quality work if

Diagnosis	Researcher's Interpretive Observations
	they are not vested. Thus, she desires a quality and standard of work that reflects her own. She feels that small size of her team and company places more burdens on each person to perform optimally and gives as much effort as she does. Again, Daphne seeks quality assurance in her processes. Thus, in looking for a sense of ownership on the part of developers, she wants to explore the "team's way" of doing things.
7. "I am concerned with skills transfer and skills cross training."	Daphne often feels like she has to show her employees how to do everything as she expects all work to be done according to her norms and her standards of quality. Despite this, she wants people to know their roles and responsibilities and act on them without prompting. Daphne is concerned that her developers will waste much time "figuring it out" and she doesn't want to "reinvent the wheel" when not needed. Thus, Daphne wants to transfer her knowledge, retain *Organizational Learning* and develop a team style that surpasses her own skills.
8. "How does a small team find the time to stop, reflect and learn?"	Daphne has a desire to further develop expertise and share and grow this expertise among the team. She recognizes the need to 'keep up' in her business and wants the team to grow and learn from their actions, be they successes or failures.
9. "Continuity: how do I retain and transmit institutional knowledge?"	Daphne feels that, with such a small team, the departure of any one member could be disastrous. Daphne asks how she can increase the chances of team knowledge and skills continuing and passing on to replacements. How can the team be self-replicating even when one or more members depart?
10. "Productivity: how do I increase billable hours and productivity?"	Daphne feels that her extant processes are not producing the productivity, measured in billable hours, that she feels her team is capable of achieving.
11. "How do I address the 'intangible of IT' and educate my clients about what I do and why it has worth?"	Daphne needs a way to make her clients understand what she does and how her processes ensure quality, rather than having the process and product shrouded in mystery and often referred to as "magic."

The significance of this list of concerns is that they were vetted by the practitioner-researcher team in early September 2009 as being representative of the practitioners' initial concerns and elicited by way of dialog. The focus of the dialog remained primarily on Daphne throughout the Dialogical AR study. This was so as, in terms of the historical and social context of SSC and its small team of developers, Daphne would be most representative. Dialogs with the remaining team members are certainly very valuable and transcripts from these dialogs play a significant role in the characterizations and descriptions in this chapter.

The next section discusses the process by which SSC's extant methods were documented. When the researcher entered the Dialogical AR Partnership, there was scant documentation which would suggest what methodological steps, if any, SSC followed at the time. There is no dispute that SSC had some implicit and tacit method, but it turns out that this method largely existing "in Daphne's head" and was passed on to employees in what could be described as an oral tradition.

5.4 Documenting the Practitioners' Extant Methods

The researcher spent his first month with SSC attending their weekly staff meeting and observing how the practitioners did their work. Technically, the developers used sound *N-tier* architectures, data access layers and contemporary tools such as Microsoft's Visual Studio and Adobe's Dreamweaver to construct presentation and logic layers for all of their projects. ERD and data modeling was done using Visio and SQL Server Management Studio. There was a

definite and discernable workflow to how projects were completed and a method by which Daphne assigned projects.

The following subsections discuss SSC's overall extant methods and processes as the researcher discovered them in the summer of 2008.

5.4.1 Characterizing the Practitioners' Desire for a Method (and Methodology)

Despite having an extant methodology, although not formally expressed, Daphne had indicated a desire for a method or methods which would corroborate what she knew to be sound processes behind her success. At this time, the researcher pondered whether Daphne's success may have had less to do her methods and more to do with her hard work and dedication to excellence.

Daphne explains her motivations for exploring a well-known, tried and tested methodology proven to work for small teams in a small shop like hers:

> ***Researcher**:*
>
> *…One of the things Daphne told me two years ago, before either of you worked for her is: "I wish I had some method to know what I am doing is right." It turns out probably a lot of what you are doing is right. So we are just looking to ... the output of my work with you would be a method for you, "the SSC method" we would call it.*
>
> ***Daphne**:*
>
> *Yeah, my thought process behind that, just so you know the context of that conversation, is that I felt like I was delivering quality software, I know what I am doing, from beginning to end, when I develop a system. I can solve business problems, but I am competing in a field that has a lot bigger players. So, I wanted to look back and say we followed these steps so I know what we deliver is comparable. It was done a different way, but it comparable quality.*

In this case Daphne desires a method which conveys and routinizes the good practices she believes she already has. Again, her know-how may have more to do with her own internal actions, qualities and judgment and less to do with a well-documented method that her employees can follow. In any case, notable in this dialog is her desire for reliability and replication; Daphne wants to encapsulate good practices to demonstrate that her company's quality is comparable to that which is found with the "bigger players." Despite her confidence in her own abilities, Daphne found, as they tackled larger projects, that whatever her undocumented processes were, they were not always efficient in the face of the larger and more complex projects she wants to continue to attract and contract for:

***Researcher**:*

So, please characterize the first six months of the year... (2008)

***Daphne**:*

…Obviously the first three and a half months, through April 15th, were very much focused on LMV [their largest contract ever], we did produce other work, but LMV was the large focus. It was nice because we were able to organize and produce a very large project, it was a $50,000 build, and for a small company like this, those were big numbers. Now, when you break it up over 4 months, it didn't pay all the bills (laughs). But, it was cool; it was a real venture into a different area.

***Researcher**:*

Yeah, it was bigger stakes.

***Daphne**:*

Uh huh, we learned a lot from that project. Just because of the amount of project management that goes into a project of that size we needed to use a different multiplier for project management...

***Daphne**:*

Because we didn't make the margin we had planned to make on that project, but it was still decent money.

***Researcher**:*

So you didn't make the margin because some of your own processes...

***Daphne**:*

Were not efficient enough to handle it...

As SSC grew and attracted larger customers, the extant and informal methods SSC used, largely culled from Daphne's past experience and handed down through direct demonstration and explanation by Daphne, were buckling. Even as the SSC developers continued to pride themselves on building custom websites, the growing number of larger projects, which arrived now with greater frequency, tested this position:

***Researcher**:*

…you started this off with "every project is unique," but it seems to me that you're talking about the customization that you do is better than the template-based cookie-cutter stuff, everybody else does...

***Daphne**:*

…we can't build everything from the ground up, every time. There has to be a format to follow so that we know that we're keeping the work consistently good, that other people that work here can look at it, and know what was done... I mean, once you introduce other developers into the mix, you have to have some way of working so that, while Fred is out of town this week I've had to fix a couple of things that were wrong with some of the work that he's done.

Accordingly, Daphne's reliance on an informal process for sharing the techniques "in her head" had become intractable. Daphne realized, as her team had grown and her business had grown that a formally expressed and documented method (and methodology) was sorely needed:

So, we need to make sure that there's some kind of standard of development. I think that the things that need to be there are the way things are put together, even if not necessarily, it's the same for every client, but if we follow the same pattern each time, its repeatable and anybody here can do it.

Actually, though, that brings me to, right around to the... something that I've said to you from the very beginning, even when I was just working by myself: "how do we know if we're developing it from the ground up every time, that we're giving the client something consistent?" Of consistently good quality and built in the right way? Because these other shops that have been around for a long time have developed these frameworks that they are using, either the systems development life cycle, or something that has been

around a long time, they have the staff and support and everything to do that process, to have their process over and over again, and they've done all this...

So, the challenge that I saw from the very beginning is: I feel like we're doing quality work, but how do I prove it?

Here Daphne realizes that the small business she once ran out of her house had out-grown a model centered entirely on her own intuition and habits. Daphne also indicates her desire to compete with larger firms of the sort that she hired Fred away from. Daphne also realizes that she needs a formally expressed and documented method just to coordinate with her own developers and to maintain consistency in quality without her direct involvement in every project. Her own direct involvement was previously the only means she used to ensure her own high standards:

***Daphne**:*

You know, …I think I said this to you and this probably is one of the things that sparked your interest in doing this research: "I need a method, by which, to say this is how we do things here and I know that it generates a good product" - so that if anybody every says, so what's your methodology? How do you arrive at your solution? Well I can say: well, THIS is how we do it. Luckily, over the years, the past couple of years, I've been able to compare my work to some other work that I've seen done and either find it of a similar or better quality so I feel like, you know, the way were doing it...

And even though we haven't written a formal methodology, which I guess is just in my head, and I have conformed Fred to what's in my head...

Luckily, he has been trainable and has listened to what I do... Johnny is going to be less conformist and perhaps a methodology becomes more important with an employee that likes to break out.

Fred was content with delivering back to me exactly what I asked for so I've molded Fred into my way of thinking, so I guess the methodology is in my head.

Daphne suggests very directly and clearly that she needs a methodology which will echo her own qualities and characteristics so that the method(s) will impute her tacit methods into her employees. The literature on agile methods, and perhaps even small teams in general, suggests that it is possible to sustain a team on the "cult of personality" surrounding a very strong leader.

However, Daphne's goal is to run and grow her business and not in full-time development. This being the case, there is an inherent flaw in "molding" a developer into her way of thinking as this transfer of knowledge and skills contains no learning mechanism. Daphne needs a team that can learn and "think on their feet." The practitioners need a learning method which will grow and foster a common repository of knowledge and learning. In the face of novelty, uncertainty and change, Daphne increasingly feels the need to codify and routinize the methods "in her head":

> ***Researcher**:*
>
> *…You've told me multiple times during our sessions so far, that it's not just the quality that SSC brings. SSC's success is also in customizing. It is building a tailored suit each time. The question is, the face of people coming in all shapes and sizes, and you being the tailor, how do you efficiently accommodate the tailoring?*
>
> ***Daphne**:*
>
> *Yeah, I would like to know "is there a way that we can approach these novel ideas that is efficient and, you know, we don't feel like we're flying by the seat of our pants all the time." I won't say that it feels like that all the time. For whatever reason, I sit down and look at a problem, I work through it from beginning to end mentally and I put people on the project and it happens...*
>
> *But, if I'm ever going to hand over the reins to other people, there's gotta be something in place that explains to them how that works because they're [her employees] not inside my head.*
>
> ***Daphne**:*
>
> *And, I don't see the jobs changing, I see them getting even more complex and new needs and new wants… and how do we make that happen all the time? So, like you said, keeping it sane would be of big importance.*

Daphne does "get it" that she needs some help. However, one early weakness in her expressed desire for a method is to replicate her own "moves" for success rather than bring about a method which harnesses the synergy of talent across the team. Fred's years of experience in a bigger team and his Master's degree would certainly bring some fresh perspective to the team. Furthermore, many of the new habits and approaches that the team had adopted in the year prior to entering the client-researcher infrastructure were practices Fred brought with him from his

work experience in a larger company. Daphne needs to harness this strength, not brush it aside in favor of modeling her team in her own image.

The researcher, in an effort to stress the importance of a learning system, begins to casually bring about ideas and themes from Schön's *Reflective Practice* in their dialog:

> ***Researcher**:*
>
> *I think that with the techniques that we'll explore [XP and Reflective Practice] you will address that…, but the real punch line, I think for a small team, and this is where I'm headed with our work together, is: are there ways that you can see yourself as a professional and also conduct yourself which gives you assurance that you're developing these things correctly? I suspect that you're already doing it anyway... It's nice to get corroboration and validation that you're doing the right thing, but ultimately you won't survive unless you feel that, embedded in your daily decision-making, and in your team's daily operations, there is the ability, adaptability to know that, for the team, you're doing the right thing all the time...*
>
> ***Daphne**:*
>
> *Uh huh*
>
> ***Researcher**:*
>
> *What if we could skip the step of method and just say, there are practices and principles your team holds and those are going to get you through, no matter what the details are...*
>
> ***Daphne**:*
>
> *Yeah*
>
> ***Researcher**:*
>
> *If I were to fast-forward to February/March, those are conclusions I think you could be drawing...*

Here the researcher is beginning to see "through" Daphne's problem and realize that there is just cause for the introduction of both XP and the tenets of *Reflective Practice*. The researcher realizes that *Reflective Practice* has great potential to assist SSC in reaching some of their goals. In addition to some of the passages above, the researcher's field notes contain many observations where Daphne expressed a sincere desire to discover an SSC "team" way of doing things to

realize the synergy of combining the team's talents. "Synergy" turned out to be a favorite word and concept for Daphne.

The preceding evidence from the dialogs illustrates that SSC could use a method, well-suited for small teams, which ensures quality and learning and which leads to synergies as the team's skills and knowledge are combined. The next section discusses SSC's extant software development processes in order to establish a baseline from which change can be discerned and discussed.

5.4.2 The Practitioners' Extant Software Development Process

During the initial diagnosis in the summer of 2008, the researcher undertook a careful and methodical examination of SSC's extant methods in order to record point of comparison for later analysis and measurement. By mid-September 2008, the researcher had a fairly clear picture regarding SSC's extant methods. The steps of SSC's extant method were discerned by way of a post-mortem retrospective and walk-through focusing on one of SSC's largest projects to date. This was an IT Services project involving "all hands" for, FMA, a machinery manufacturer located in Central Virginia. FMA's customer base is worldwide and the extent of the project was considerable. The researcher also asked general questions and took the development team through a hypothetical project in an attempt to fully characterize SSC's development processes.

New Business: Assessing Need

New business would generally be of three types: simple websites, complex websites and IT services. Daphne handles all initial customer contact and needs analysis. Daphne also

supplies a potential customer with a survey to fill out, called a "Web Needs Analysis," and then conducts an interview with the potential client to establish a clearer picture of their needs. Daphne then took this information and drafted a proposal and technical requirements document containing cost estimates based on developer hours. Once this process was completed, the customer accepted, negotiated or rejected the proposal and the new contract would proceed, or not, based on the outcome.

Selecting a Project Manager

Daphne then selected a project manager by assessing the size, complexity and extent of the job and assessing each developer's capabilities accordingly. Fred, being the most experienced, would be assigned the most complex web projects, which usually involved eCommerce and /or an IT Services. These IT Services jobs had steadily increased since 2005, especially so in 2008 and 2009. Johnny would handle the overflow from jobs Fred would pass on and was also committed to a long-term maintenance contract for a desktop application used by a university client in Central Virginia. Both Johnny and Velma were also assigned most of the simple website projects. In any case, whether expressed overtly or not, Daphne was a co-manager of every project as she would start the projects off and draw up the requirements.

Initially, direct customer contact with the PM was spotty and largely depended on the project. However, direct customer contact varied such that customer interaction on some projects was extensive and very involved. Fred, Johnny and Daphne each agreed that projects where the customer was more involved provided a deeper and richer understanding of the project as a whole. However, there were some drawbacks as SSC had not established any formal protocols concerning customer interaction. Some customers tended to use interaction with the

team for the purpose of micromanaging along the way, which did not help the development process and, rather, hindered it. Fred was particularly vocal on this topic as he made significant changes to the FMA project based on last-minute feedback from a "hidden" stakeholder. These issues would continue to plague SSC throughout the practitioners-researcher partnership, although the nature of these issues changed as new techniques and methods were introduced to the team.

Layout and Design

With most website development at SSC, the "look-and-feel" of the interfaces is drawn, as art or illustrations, in image processing software such as Adobe's Photoshop. The website design was either supplied by the customer or via 3rd party graphic designer. This seemed odd as Daphne had an extensive background in graphic design. However, SSC was so busy with development that there was no time or resources to also do graphic design. Almost exclusively, Velma would be responsible for page layout and content placement which primarily involved a process where the images for each page, or for the overall template for all pages on the website, would be "cut" to provide placeholder sections for content and for program and business-logic. As Velma had worked with Daphne the longest, Velma's layout process was very predictable and Daphne was able to make very accurate estimates on how many hours layout would take on any given job. Generally, Velma required 14 hours to create templates and layouts for a new site.

Design and Code

Design and code activities primarily fell on Fred, Johnny or Daphne roughly in that order. Fred would usually read the technical requirements document and use that, sometimes in

consultation with Daphne, to create an ERD. SSC has a very specific architectural approach based on an n-tier architecture strongly resembling the Model-View-Controller design pattern. SSC also makes extensive use of T-SQL stored procedures in a Microsoft SQL Server database. Despite their stated desire for re-use, modularity and "productization" during the diagnosis phase, SSC's existing architectural approach appeared modular as is. Most code was written to take advantage of other strategies for re-use such as ASP.NET custom controls and also .NET class libraries for data and logic abstraction. Thus, SSC's desire for reuse largely had to do with identifying portions of code from previous projects that could be reused on new projects.

The coding process varied from a week to 90 days depending on the size of the project. Most of the value and quality SSC offered in their product was developed during this phase. Fred, having extensive experience in other development environments, was responsible for maintaining this quality under Daphne's watchful eye. Daphne and Fred shared some of the same traits and approaches to coding which made the two of them the most compatible on the team. Daphne spent considerable time training Fred to do things "her way" and Fred was usually able to replicate Daphne's patterns.

Test, Debug, Launch

SSC usually contracted with a client for a 30, 60 or 90 day turnaround. SSC termed the day a website was due for public availability a "launch date" and internal Quality Assurance (QA) processes usually ramped up a week prior to this date. SSC maintained an IBM multi-core testing and deployment server onsite as well as utilizing managed hosting offsite which SSC then resold to their clients. Thus, SSC used a fairly reliable and predictable testing and deployment environment. There were some exotic and esoteric installations with several past clients, but

SSC, Daphne especially, demonstrated an adept ability to quickly accommodate "out of the ordinary" installations. Frankly, SSC suggested that an "ordinary" install didn't exist as every website and client were different.

After an internal QA process involving the PM and Daphne, Daphne would determine if a website was launch-ready. At this time, the customer would be invited to review and inspect their product. Quite often, something was not right and, despite requirements specification documents and other agreements between SSC and the client, last-minute adjustments were often ordered. The researcher saw that SSC would benefit from an agile process in this case as agile processes call for early and frequent customer involvement where the customer is also considered a member on the team.

Launch, Post-launch and Maintenance

Most projects of any size, including FMA's project, required post-launch adjustments, feature enhancements and maintenance once the "working software" was used in its intended environment. Sometimes images would be the wrong size, or a menu and of some other user interface feature would be deemed "wrong" and require amendment. As for maintenance, in some cases, the client had their own hosting or hosted within their own IT infrastructure. In many other cases, the client was on SSC's managed hosting and could be easily maintained.

Reminiscent of the Waterfall Model

As a sketch of SSC's extant methods increasingly came into focus during the summer of 2008, it was apparent that SSC followed the classic waterfall model of the Software Development Lifecycle (SDLC). The literature on agile methods, small teams, and web development suggests that the waterfall approach is not effective for small teams nor is it appropriate for most web

development (Cockburn 2000; Cockburn 2002; McDonald et al. 2001a; Pressman 1998). Figure 36 shows SSC's extant methodology.

Figure 36 SSC's Extant Methodology

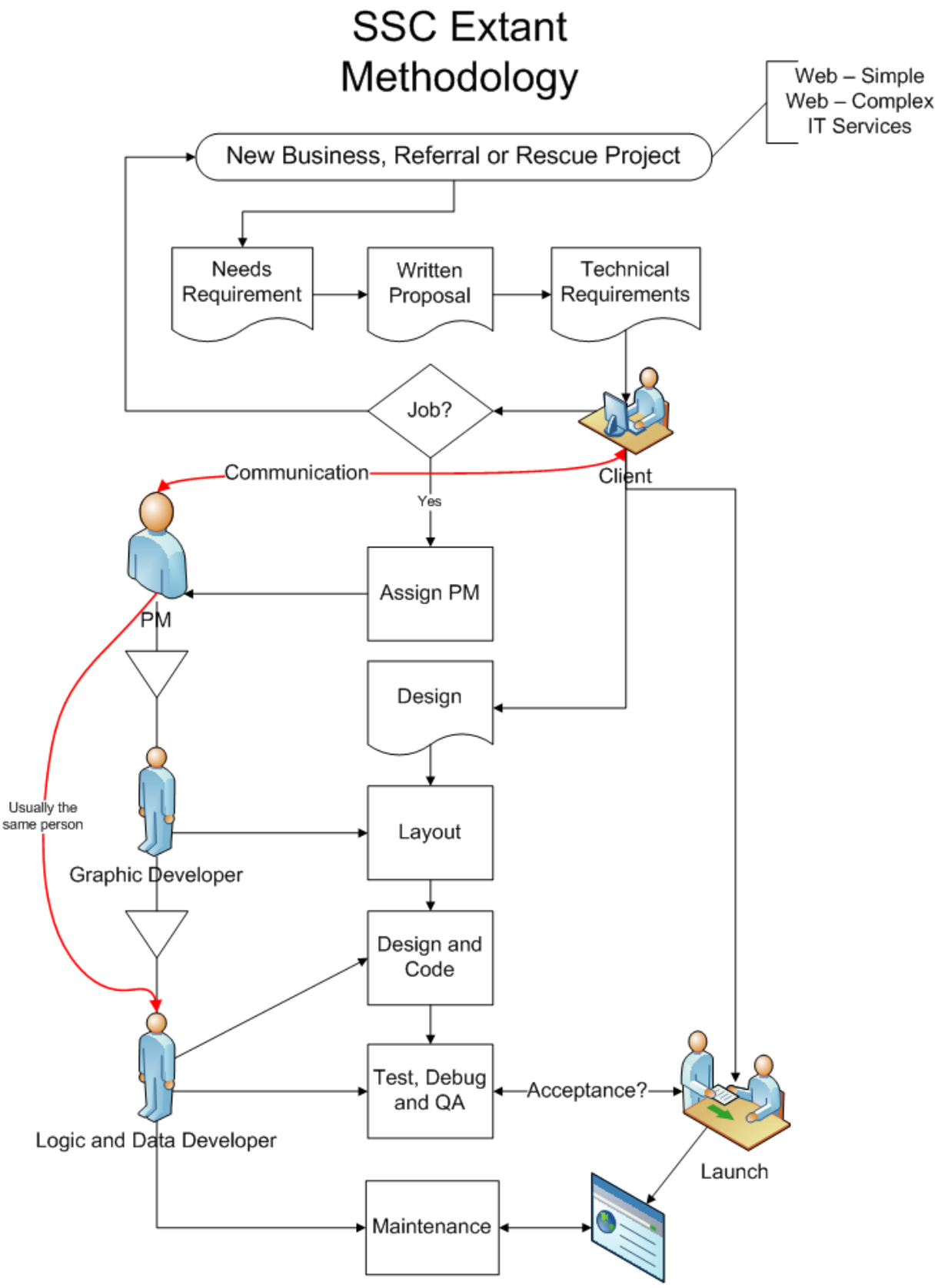


Figure 37 SSC's Extant Methodology and the Waterfall Model

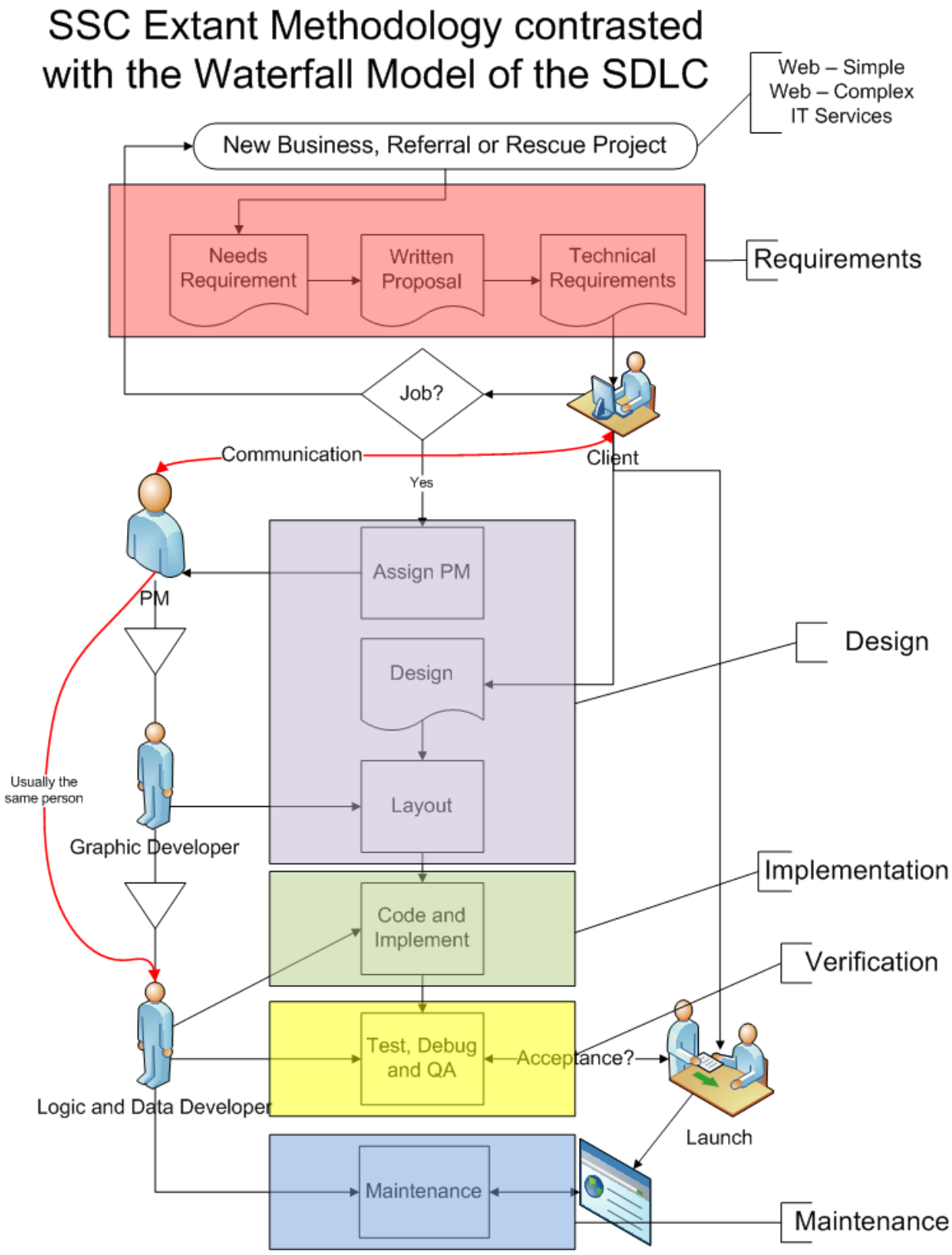


Figure 37 shows the steps of SSC's extant methodology as contrasted to the waterfall model of the SDLC. In February of 2009, as the Dialogical AR Partnership was concluding, the researcher asked SSC's team what they remembered of their "old" method and few of the developers remember much of it as their old method was never formalized and their newer method was formally expressed and familiar. When Figure 36 and Figure 37 were shown to SSC in February 2009, all developers agreed that the diagrams generally captured their methods of operation and also agreed with the characterization of their former method as an example of the waterfall model. The SSC developers each professed that such a sequence was "something they learned in school."

5.5 Addressing the Initial Diagnoses

This section, and its subsections, provides a deeper description, characterization and discussion of the dialogical evidence from the diagnosis phases of the Dialogical AR cycles. Also, each subsection of this section elaborates on the list of concerns (Table 30) which arose during diagnosis throughout the Dialogical AR effort and its iterations. These subsections draw from transcripts from practitioner-researcher dialogs in order to address the issues listed in Table 30. These are: issues related to quality (items 2, 3, 4 and 11 from Table 30); issues related to learning (item 8 from Table 30); issues related to productivity and process (items 1, 5 and 10 from Table 30); and issues pertaining to team dynamics and leadership (issues 6, 7 and 9 from Table 30). Additionally, some issues related to clients and other parties external to SSC are discussed as they are salient to SSC's adoption and use of a new method.

5.5.1 Issues Related to Quality

Quality was an important theme that arose during the initial diagnosis and throughout all of the practitioner-researcher dialogs. Issues related to *Quality* occur four times in the list of concerns listed in Table 30. The desire for methodological assurance of quality is demonstrated in a previous section of this chapter. Daphne's fastidious attention to quality assurance is a key driver in her ability to have grown her business since 2005:

> ***Daphne**:*
>
> *There are a lot of people, from the first time I got in, and I know that I mentioned this before... there's always somebody out there to do it cheaper. So, why is it that we charge what we charge to build a website and that's a question that I get very often and continue to get to this day…*
>
> ***Researcher**:*
>
> *One thing that is clear to me is you really value quality and your reputation and so you really jealously guard these. It comes up over and over in your language and that's good.*

Furthermore, Daphne expects her employees to demonstrate quality and to uphold the virtues of quality assurance. Daphne often feels as though her employees are not capable of producing the same quality that she is capable of. Perhaps this is partially due to the greater risk she assumes and her greater experience or perhaps it is innate, in any case, Daphne firmly believes that only she is capable of producing the highest quality of work in her team. Daphne feels that, among other things, it is her desire for excellence which sets her achievements in quality apart from her employees:

> ***Researcher**:*

Right, so what we want to step back and ask ourselves is: Well, what is development; is it just writing code?

***Daphne**:*

I believe that those fall under systems analysis and design, but if my developers are to be as well-rounded as this technique we are trying suggests and I would like to see them to be then they have to be able to do the system analysis.

Well, they do, in fact, would you not agree that the technique does address mail system design? The whole user story part is all about design and requirements. I would think you should expect this given the training of your employees... but I would think you wanted well-rounded, do-it-all people

Now, here's a question: Does system analysis and design suggest a level of maturity that has to be present before you can put somebody into that position?

***Researcher**:*

Perhaps.

***Daphne**:*

Because if I applied Johnny straight into a project, I guarantee you that he would miss details because he just hasn't learned that the details are important.

***Researcher**:*

I agree that details are important; that's why you need the apprenticeship that pairing [pair programming] can facilitate; or, you you'll have to get only experienced people

***Daphne**:*

Unfortunately, one of the ways that Johnny learns is to make mistakes and I do let him make them.

***Researcher**:*

Well how about you?

***Daphne**:*

Well, that's how I learn! (Laughter)

***Researcher**:*

I mean, you, relative to the work force, and your peers? You are the master, relative to them. There are people out there that you are junior to. I realize that and we all realize that we are on a journey...

***Daphne**:*

That's a very important point because there is something that we touched on when I was trying to talk about Velma's approach… when I suggested something that I knew needed to happen and she questioned me on it.

Um, I think what I was getting at with the whole… is raw intelligence enough? I think there also has to be a desire for excellence.

…And the realization that there is always a new level of excellence to be achieved.

***Researcher**:*

Uh huh, absolutely.

***Daphne**:*

…Because I do not think without that desire for excellence that true learning can take place.

In this exchange Daphne not only demonstrates her desire for quality, but also questions whether others are up to her level in terms of striving for quality. Daphne continues:

***Daphne**:*

But I think that desire for excellence easily trumps raw intelligence…

***Researcher**:*

Maybe they both need to be there?

***Daphne**:*

Yea, I agree with that, but… I don't want to put this in terms of just intelligence measures, but there are measure of intelligence like IQ and things like that. I don't know what Fred's IQ is or Johnny's IQ or Velma's IQ. I don't know what their IQ's are. Do I think they are the same? No. Do I think they could all achieve the same level of ability? Yes. I think they are all smart enough that they could all reach a level of excellence if excellent is their goal.

***Researcher**:*

…What matters is the ability to do this well... I would say, because they [Fred and Johnny] both received Information Systems degrees -- I was trying to get at this earlier, they are both Information Systems trained people -- all three of you are. There should be some advantages to that. In fact, I almost think it is a test of the quality of your [educational] program just to see what is going on here, because they were taught… their course work forced them to see the development process from a lot of levels. I don't think software development is all there is to it. It's also delivering the product and maintaining it - from the twinkle in the eye to the reality, the whole deal. So you need to hire people that can deliver the hardest part and that separates the wheat from the chaff

in a lot of areas, but people coming out of your [education] program… should be able to fit your bill. If they made A's and B's, they should be able to fit your bill. Now, what kind of people are they? I don't know? Do they strive? Are they aspirant?

***Daphne**:*

I think part of the piece of that is that you also have to be humble enough to realize that there is always somebody better; otherwise, you can't be trained.

It is revealing to hear Daphne speculate on the intelligence inherent in her employees, but it is even more revealing when she suggests that a desire for excellence is sufficient to achieve greatness and quality alone. From this dialog alone, the reader may develop the impression that Daphne is the only practitioner who desires quality in her team, however, the researcher's field notes contain assertions from the rest of the team concerning the importance of quality – even if only as a matter of survival. Daphne summarizes this sentiment as the researcher discusses the XP philosophy of *embracing change*:

***Researcher**:*

…the byline to Extreme Programming is "embracing change." This idea is supposed to develop a culture in your team where change is okay. And sometimes people don't like change because it is volatile and they want a safe harbor; I think that is human. The safe harbor is the team…

***Daphne**:*

Well, the safe harbor is knowing that by delivering quality work we ensure ourselves a future work. (Laughter).

In offering this diffusing humor, Daphne is reasserting that a commitment to quality is the driver for growth and success in her company and remains key to future success. Thus, Daphne will not likely embrace any method which does not hold quality as a core principle. In general, Daphne knows that her dedication to ensure a quality in her work that is superior to, or at least on par with the best products available, has delivered her success.

5.5.1.1 Ensuring Consistent Quality

Daphne was repeatedly clear that one of her primary expectations of any formal method was that provisions exist for maintaining quality. In an earlier dialog, Daphne relates her desire for a method which allows her to approach novelty in way where success is repeatable:

> ***Daphne**:*
>
> *But that's one of the things I really like that we do around here is that I've had to come back and say "what's possible, what's possible," because, like I said, MNM [a strategic partner] is quoting these jobs and they're not an IT company, so they're like, "what can be done?" and I'm like, "anything can be done." And that's the answer...*
>
> *It's a matter of time and money... Anything is possible.*
>
> *It's just what time and money are you willing to put into it? So they, they're always coming at me with a new idea: "this client wants to see this happen" and "this client wants to... can you make this happen?" And I say: "Sure, what's the budget?" (laughs)*
>
> *So, I don't think novelty will ever be a problem, but novelty as far seeing new projects all the time, seeing what people want to accomplish. Where the problem comes in is: "how are we sure that we're approaching novelty so that it's repeatable?"*
>
> ***Researcher**:*
>
> *That's actually very related to your method. Because what you want to do is produce high-quality... everybody wants to do this... everybody wants the highest amount of quality in the lowest amount of time. Your billable hours... what that really means is that the more you have to pay them versus what the job was estimated, the poorer SSC is.*
>
> ***Daphne**:*
>
> *That's right…*

In this case, Daphne makes a clear connection between quality assurance and the price that she can charge for her work. This leads to the next major issue related to quality, which is convincing her clients that quality is worth paying for. This topic is discussed in the next subsection.

5.5.1.2 Justifying Quality

Daphne frequently mentioned the need to convince her clients that quality was worth paying for. The developers at SSC understood this point as it relates to quality; Fred often spoke of the need to maintain quality in order to command top price for their work. Nearer to the end of our dialogs, Fred and the researcher had the following dialog on the topic of quality:

> ***Fred**:*
>
> *That's what you are paid to do, but here we are not, essentially, we are not paid to test. We are testing for the sake, the strength of our business and our projects, which obviously is an important thing. The question is: does it bring up an argument that at some point maybe you ... it is so a part of what you are doing, you could argue the quality of your work is that much greater....*
>
> ***Researcher**:*
>
> *Uh huh.*
>
> ***Fred**:*
>
> *.... that, I mean, maybe even, maybe this isn't the point -- you will be able to say "we have to charge a little bit more for this stuff because we know we are actually delivering this much more quality to the product to support all the extra time, because it could easily be a part of what you do as opposed to, "Oh, man, I have to finish this junk of work; I tested it, then I finished it, then I blogged about it, then I talked about it. We put it in the wiki…*
>
> ***Researcher**:*
>
> *Sure, great.*
>
> ***Fred**:*
>
> *But how much of that can you cover?*
>
> ***Researcher**:*
>
> *Well I think it is a classic point. …I tried to illustrate this to Daphne through some allegorical examples. In the guitar world, selling guitars and making guitars, the person who makes and fixes guitars is called a "Luthier," that is the term for it. There are a lot of run-of-the-mill guitars; they are legitimate and functioning guitars, they'll do the job. But there's a higher craft guitars - hand crafted from artisans and they command big bucks for their work. And, generally, these people who craft custom guitars aren't sitting*

around saying, "Oh, God, I wish I had more business, I'm not going to eat next week." They've got a waiting list over a year long. So, lesson is: well-crafted artifacts are recognized as such and generally rewarded. Is quality risky? Sure. But you are making the case that your practices are so tight and sound that the quality --- well, Daphne says she already struggles with promoting and justifying quality as it is; she says that you already do command such a premium price that a lot of potential customers just can't stomach. But other customers, they'll pay for quality. That's the kind of business I hear you [SSC] want to break into and the better your habits and processes are, the more you should expect that. All well-paid craftsmen and professionals, you don't even think twice about commanding top dollar for top-quality work. There's not even a question about it; if you want the best you are going to pay for it. I think you make a great point, when you invest so much in the quality of your process and, therefore, it comes out in the quality of your product, have you stepped up your game and should you bill for that? Well, I think that is a discussion for you guys to have. I think it's a very valid point because you are also talking about professionalism...

In this dialog, the researcher develops an understanding of the Fred's attitudes towards quality and shares his understanding through the metaphorical device of an allegory. Further to this point, Fred offered his own understanding of professionalism and quality:

***Fred**:*

I like to think of a Carpenter's profession, I think they are up against a lot of the same sort of project structure that we are up against. A lot of their work is where things are custom, they have to sometimes learn as they go, face challenges, deal with the architectural issues and structure and all this stuff.

***Researcher**:*

Does a really great carpenter command a really good price?

***Fred**:*

Yeah, absolutely.

***Johnny**:*

They say a good handyman, if after you tell them the price of what they do and their mouth doesn't drop, then you've priced it too low. That's the handyman joke.

In this dialog, Fred understands that other professionals share some of his burdens related to price and quality. Johnny also demonstrates his interpretive understanding of the issue. In an earlier dialog, Daphne and Fred were talking about quality and Daphne had this to say:

> *Well, one thing that I'll -- an observation that I'll give about quality is, Fred and I have had the discussion before. He's like, "Well, we really should deliver it this way," and I'm like "Yeah, but we really can't afford to in this scope of a project."*
>
> *I think that this, having the client involved, and accepting what we deliver every week helps us deliver quality to their standards within the scope of the project...*
>
> *...so I think that quality becomes controlled by the interaction with the client in the scope of the project. So, I think we will be able to tell whether we deliver something of quality going forward.*

Here, well into adopting and adapting XP, Daphne reflects on quality enhancement via client involvement. Yet, in the earliest phases of diagnosis, Daphne clearly characterized her struggle to justify quality:

> *But still found people struggling with the idea that it might cost you $2000 to have a website built. I got to thinking about... or I had some people talk to me about other types of advertising, the yellow pages guy comes in and you might spend $10 or $20 thousand dollars on your annual yellow pages ad. I'm like "why would you do that?" People nowadays do not pick up the yellow pages, they go to the Internet. So, there was that constant struggle of trying to explain to them the use. I guess that's where the intangible of IT meets the real world because, like it or not, a website is an IT project and even though it has a pretty front-end, the pretty front-end was built by Velma in 14 hours, why does the project take 30 to 40 to complete? It's the not-so-pretty code behind that makes it actually work… because I'm always getting: "can you build an eCommerce site for $2000?" No, not here! (Laughs)*

Daphne is already demanding quite a bit out of her methodology, but her attitude is a byproduct of her approach to her business and growing her business. Throughout our dialog, it was abundantly clear that she jealously guarded her reputation and directly associated the quality of her work with her reputation:

> ***Daphne**:*

We have a deadline today; I have one and Johnny has one. And, meeting that deadline is far more important to me than him.

And it is just… he has a very laid-back approach to everything, so we have a problem that will be brought up in the meeting next week specifically on how important deadlines are to the company, because we have to be perceived as someone who delivers.

***Researcher**:*

Right. Yes, your reputation...

***Daphne**:*

Yeah. Now, obviously that's more important to me than any of my employees.

***Researcher**:*

Yes.

***Daphne**:*

How do I make it mean something to them?

For Daphne, SSC's reputation also comes down to factors such as delivering on time, providing added value through custom-built websites and strict observance of ethical behavior. Even as Daphne grapples with reuse and "productization," she realizes that she needs to tread carefully so that she isn't "double-dipping" when she charges clients for re-usable components.

5.5.2 Issues Related to Learning

Our dialogs also revealed how important individual and team learning were for Daphne and also highlight the scarcity of time available for learning. Learning has played a key role in Daphne's ability to grow her business. She has repeatedly recognized the need to learn new skills to move her business forward and she has exerted whatever effort required to posses that skill. Daphne learned how to create a good N-tier architecture from self-guided learning and a die-hard inquisitive spirit:

> *The first time I actually saw an n-tier application was right before I had left the Toy Shoppe, we had, I think I had mentioned this before, I had done a .NET upgrade with a piece of software called .NET Storefront, we purchased the source code for that and I got to take it apart a piece at a time. I saw how a production eCommerce package was built and maintained. And it was a simple n-tier, it wasn't a framework, but it taught it... it finally clicked for me exactly what they had been talking about at [school]. I began to work toward fully achieving separation of code and presentation. I won't say that I did it immediately but I've gotten a lot better at it over time.*

At times during the dialogs, it seems as though Daphne is too hard on her employees or has unreasonably high expectations, but these expectations are understandable not only from the perspective of her commitment to quality, but also her commitment to excellence through learning. Daphne identifies the learning that is possible when every project is unique in some way.

> *Well, part of the challenge, but what also makes it interesting is that every project is unique... I don't care what kind of project I work on and even though they're starting to fit certain patterns, there's something about every project that's different... I wouldn't say I've ever encountered two projects that are exactly the same, somebody always comes up with something new that want to see done. So, because of that, a lot of people see that as reinventing the wheel over and over again...*

Daphne also recognizes the value in learning from mistakes, but she also wants her team's use of a new method to facilitate a learning process where lessons from mistakes don't slip away but are retained. On the topic of professionalism and mistakes, Daphne said:

> *Yeah, well, okay, "professionalism" I would say that in my handlings of company, the ethical way I lead my business, what I expect from my employees, the fact that I am not, I don't beat my employees up for mistakes, but I expect for them to learn from them and I also don't purport myself to be perfect and I try to learn from my own mistakes.*
>
> *But I do think that if you refuse to learn from your mistakes and you keep beating, bringing up the same thing over and over again; I do not consider that professional behavior…*

Daphne also feels that a level of humility is essential in order to learn from mistakes. Despite an espoused esteem prioritization for learning, our dialogs also revealed how the entire team was always pressed for time. In the name of professionalism, there is some expectation to learn "on

your own time" and during the "after hours," but there was little to no time built-in to the software development practices at SSC for ongoing and systematic learning. Fred realizes the implications of the paucity of time spent on learning and, during a dialog well into the entire Dialogical AR process; Fred relates the utility in apportioning time for learning and for reuse:

> *I believe there is a long-term benefit to all this because the last group I worked with, you know, 3 1/2 years, and it always came up over and over again in meetings how we needed to have some sort of central stash of reusable stuff and we always talked about it but never quite put it together. For the exact reasons here: a small group sort of...so to be able to make time for it -- it would be... it's like the prudent thing to do, like you would have to really force yourself to do.*

During our dialogs, I noticed repeated mention of new and emerging technologies the team wanted to learn, and insisted they would make time to learn. However, for the duration of the Dialogical AR study, they did not make time to learn these new skills and technologies. Some of these technologies, AJAX, XML, WPF/XAML and LINQ, would likely improve the quality and competitiveness of their products and ensure greater customer satisfaction. Daphne and SSC were certainly not ignorant of these facts, but business was thriving to the point that very little time was available.

The researcher's field notes are rife with annotations and memos pointing out the team's lack of time for research and development (R & D). At best, Daphne suggested that:

> *I think everybody, as an individual, ends up with a couple of hours here and there that they could spend on learning. Unfortunately, what happens is that those are never at the same time.*

I can vividly remember the rich irony of a dialog cut short when Daphne replied with "we've got to move on, time is pressing" in response to a oration where the researcher was stressing the importance of making time for R&D. This was but one characterizing illustration of how SSC did not afford significant time for learning. Often, Daphne would simply wait until the "hype

cycle" of a new technology reached a high acceptance rate before she would devote time to approach it: "*until it becomes about 60% or 70% accepted, we can't use it.*" As a hedge, Daphne does find small provisions and excuses to at least dabble in new technologies:

> ***Daphne****:*
>
> *Now I have, like with work on our own website and stuff like that, I require them to use the newest technology, because that's where it doesn't hurt us... That's what actually makes us look better.*
>
> ***Researcher****:*
>
> *But you're not getting any billable hours on this...*
>
> ***Daphne****:*
>
> *That's right*
>
> ***Researcher****:*
>
> *…and that's my question: How do you innovate and still balance billable time?*
>
> ***Daphne****:*
>
> *Yeah, like Fred built Gallery in AJAX and it's the latest versions of AJAX to the extent that it doesn't run in IE6, it only runs in IE7 and Firefox2, so it is the newest AJAX out there, which is cool...*

Thus Daphne, despite her desire to spend inordinate time on learning everything new, has to rationalize when, what and where to incorporate new technologies.

Again, I don't believe that there is a lack of desire or concern for learning new technologies. Daphne certainly identified the need to keep up when she submitted to additional and continuing education beyond her bachelor's degree in graphic design:

> *I was one class short of achieving the associate's degree at [a local community college] still feeling like I knew very little about what it would take to program a really fantastic, awesome website. You know, to do everything that I could see coming on the horizon for the web... I was like: "we've just scratched the surface." I felt like, I called it the "stepping stones." There's like stepping stones on a lake. The lake is what I need to know. I was just getting the very highest points of what was out there.*

Even as Daphne's own role and responsibilities turned more towards running her business, her need for strategies to keep SSC's use of technology was at the forefront of her thinking:

> ***Researcher**:*
>
> *Okay, so: being so small, what do you do to ensure that you, and everybody else on the team, is at or ahead of the curve on emerging technologies? Because you are a web development and technology company, you've proven that you've done lots of developing web sites, interactive, high-quality websites...*
>
> *So, what do you do, being such a busy person, and your staff being busy?*
>
> ***Daphne**:*
>
> *Well, see for me, I'm faced with the Bill Gates dilemma: Do I grow the business and become wildly successful as a business owner or do I try and keep up with the technology? I go back to the... I look at his example and my business idol, which is Walt Disney, and he said: "You hire the best, you work only with the best" and that's how you succeed.*
>
> ***Researcher**:*
>
> *And that's how you will keep up with it...?*
>
> ***Daphne**:*
>
> *Uh huh, so if I hire in new people who are the best, or who have learned the new technologies, then I should be able to stay up with the curve that way...*
>
> ***Researcher**:*
>
> *This is an interesting thing for us to explore. So you feel the transformation of being the technology and business leader in your company to maybe <u>not</u> being the technology leader in your company; and, you foresee that and you feel that that shift is imminent?*
>
> ***Daphne**:*
>
> *Yeah, it does, it does feel that way... there's no way that I can keep up with the technology and grow the business at the same time.*

Unfortunately, this assertion conflicts with other beliefs Daphne holds about her own skills and those of her employees, but her forward thinking in this dialog reveals healthy expectations for the future.

Of course not all learning is concerned with the latest technology; some learning is about personal and team improvement. Also, SSC does not just desire an improvement in individual learning, but also desires a process to ensure that the team learns. Daphne, while recognizing the importance of this type of learning, flatly and honestly answered a direct question on the matter:

***Researcher**:*

When do you guys have the time to stop, reflect and learn?

***Daphne**:*

We don't really...

The researcher begins to wonder if this lack of time is symptomatic of the pressures and demands placed on all small teams in a small-shop environment. It seems as though some larger IT companies are able to encourage and facilitate continuous learning for their employees and yet, in this case, the spare time is just not there. SSC's new method would also need to address this issue of learning.

Certainly the time spent in dialog with the researcher was time away from productivity, so it must be acknowledged that the very act of entering into the Dialogical AR Partnership was an investment in learning. I do not fault the team or Daphne entirely, but the evidence from the dialogs emphasizes the need for a new methodology to focus acutely on mechanisms and processes for continuous learning.

5.5.3 Issues Related to Productivity and Process Optimization

At various times during our dialogs, Daphne felt that her initial patterns of operating, which served her well on her path to success, were in doubt now that she had several employees and jobs of increasing size and complexity. Often, Daphne spoke of making profit margins and

whether her PMs would take the initiative to bring a project under budget. Daphne recognizes the need to incentivize her employees and also acknowledges the effects that disparities in motivation have on the team as a whole:

> *And Fred has finished his part of the job waaaay under schedule and Velma just blew it out of the water and he was not pleased... and he did not attack her or anything in a meeting, but he brought it up, which was part of what I wanted to accomplish by making them individual project managers, I wanted to make them personally responsible for some of the jobs so that they would feel a sense of ownership over the job... The reward program for bringing a job under budget and giving them some kind of bonus was part of that...*
>
> *But the other thing is, I'm not the only one constantly coming down on people. The project manager gets to say: "well, why is this taking so long?" or gets to point out "well, you know so and so took 10 hours longer than they were supposed to..." and we can talk about that in the meeting, so that was my goal... you know, to have them work together as a team...*
>
> *…With a different project manager on each one, the jobs also tend to be specific toward a certain skill set, so that usually determines who gets put on as project manager. But that's been part of my trying to get them to see that they're responsible for that. Because if it comes up in the meeting and it's bad work, they get all upset and everything, but I'm like: "You know what, you're the one who turned in the work..."*

Thus, Daphne expects that any method she adopts would distribute a sense of ownership across the team and allow for efficiencies across the team. This also involves each team member "owning" the problems and pitfalls of a project in addition to the benefits of success:

> *…And Fred and I have had that discussion because he will ask me: "How long should I spend trying to figure it out?" He wants me to say "spend two hours on it and if you can't figure it out, then come to me..." There's really not an answer for that because I do want him to gain experience in solving the problem... there's a certain level of experience that can't be attained until you beat your head against the wall solving the problem, but in a small company, you can't absorb a whole lot of that cost...*

It seems clear that the practitioners at SSC most value open and direct communication in order to maintain focus and inertia. On the subject of meetings, Daphne intimated how meetings are an opportunity for communication and awareness:

They have all told me that they like the meetings because they know how much work they have to do, they like knowing what they have to do during a day. Before the standup meeting they'd be like they weren't always sure what they were going to be working on. It wasn't that the work wasn't here; it was that I didn't have the time to go over it with them.

So we maintained a sense of what is going on - what it is they are expected to do and knowing they need to deliver that work.

As her team adopted new methods during the course of the Dialogical AR Partnership, the team's perceived value of meetings only increased:

I think that the daily standup meeting is really helpful from the aspect that it helps me see what I need to have available for the employees so that they can keep moving; it prevents them from having to wait on me. Especially when there's software purchases and stuff involved that they can't do on their own.

Apparently, the benefits of increased communication are of great importance to SSC and are an expected byproduct of any methods they adopt.

For Daphne, the greatest and most important measure of process efficiency relates to individual productivity, which she checks on a daily basis. Daphne's primary metric for individual productivity is "billable hours." Daphne focused on any aspects of method which afforded increased billable hours. As SSC were learning and adopting XP, the *Daily Standup Meeting* was instantly attractive as it offered a means to know, on a daily basis, how all projects were progressing and about the productivity of all employees:

***Daphne**:*

The way that you describe the standup daily meeting... I can immediately see an increased productivity coming from that, which is weird because I know that you thought that it might seem something like a drain on processes.

***Researcher**:*

Yeah, I was concerned...

***Daphne**:*

But what happens a lot of time, you know, when we have the daily meeting and I always finish the meeting, you know, the Tuesday meeting...

... I was like "okay, everybody know what they are working on today?" They are like "Yeah" and they take off and start working. And I think that is a good feeling for everybody. The feedback I've gotten back is "I love Tuesday, I know exactly what I am doing."

***Researcher**:*

Now you can't take an hour every day, I think you work your way into that. Maybe the first several will take a half hour, but I feel that, if you did this for several months and I revisited this in December, and you stuck with it and found benefit in it, that the daily meeting wouldn't take longer than 10 minutes.

***Daphne**:*

I do see in their timesheets, which is not something that each of them get to see (each other's), that there is down time and lost productivity. So I am hoping... I would think that something like that would move the process along more fully and even if there is downtime, maybe doing some of that shared sitting around and learning from one another.

Daphne's focus on productivity is understandable as she personally feels the risks associated with running her own business and her own take-home pay is affected by the actions of her employees; when her employees are less productive, profits are down and she is paid less. Productivity and profit are obvious goals for any business, but a small business seems to feel variations more directly:

***Daphne**:*

Every time you hire somebody you face the same thought.

***Researcher**:*

So, one problem with growing... or with growth is contraction?

***Daphne**:*

Yeah, will the work volume slow down?

I mean there have been times when Fred and Johnny were slow. Luckily, we continue to... we pretty much break even around here, we don't make a whole lot of extra money... we really don't make extra money. But we're in our first... what is this? Our second full year of being in business and look at the overhead we took on starting in October of last

year. Fred was really the first fulltime employee on the books because Velma is technically part-time working 30 hours a week. So, all of a sudden I had to have Worker's compensation insurance, I had to... we had to go monthly on our taxes instead of quarterly... you know we took on rent, we took on...

Daphne's motivations and attention to profit and productivity are easier to understand in light of the preceding dialog. In the month-to-month operation of her business, Daphne is acutely aware that productivity, expressed as billable hours, makes the difference between, profit, breaking even and losing money. These issues often come down to the actions of the individual developers on her team.

Daphne:

Right now, in order for the company to be viable... if Fred and Velma were 100% billable, the company would pays its bills, just the two of them. But they are not 100% billable, so Johnny and I have to make up the difference. And with Fred out of town this week, we're going to take a big hit on billable hours.

Despite year-on-year growth, Daphne must run close-to-the-bone margins at all times. Thus, Daphne has high expectations that a new software development method would provide quick and immediate effects with respect to billable hours. However, beyond profit alone, Daphne also desires synergy in the combination of the talents on her team:

I was talking to somebody about that and they used the word "synergy." The parts together are more than the individual... Working together, they are much more effective than if they were working apart.

The next subsection goes on to discuss further the ways in which the burden of leadership affects the dynamics of Daphne's team.

5.5.4 Issues Related to Power, Risk, Skill and Leadership

A small team is also an intimate team where the habits, quirks and foibles of each individual are omnipresent. Compounding the effects of constant close association is the deep interplay in team dynamics when the owner of the company also assumes the role of team leader. Thus, as would be expected, Daphne must wear many hats and assumes many responsibilities and duties. Understandably, Daphne, wary of losing control of what she has worked so hard for, holds high standards for herself and her team and remains cautious about delegating authority. The subsequent subsections use evidence from the practitioner-researcher dialogs to illustrate and characterize these issues.

5.5.4.1 Power, Leadership and Running a Business

Stan Lee, creator of Marvel Comics, once quipped: "With great power comes great responsibility.[15]" Issues of power, leadership and responsibility, as they relate to running a small web software development shop and team, were a constant and recurring theme throughout the practitioner-researcher dialogs. Thus, the burdens of leadership and decision-making were often the subject of Daphne's reflections. On the subject of the value of her time, relative to the sum total of her responsibilities, she once retorted:

> *I said to her: "you're not getting even one of my employee's time, which is also very valuable, you're getting MY time..." and I don't think you understand that...*

Despite her time being precious and jealous of the use of her time, Daphne also conveyed the sense of isolation that sometimes accompanies the burden of leadership:

[15] This is paraphrased and taken from Stan Lee's "Spider Man" comic series. The phrase originally appeared in August 1962.

I was really looking for the team experience because, one of the things that's true of being the business owner is that it is lonely at the top if you have to make all of the decisions.

Despite the loneliness of leadership, the responsibilities of leadership often demand assertiveness with respect to risk-taking and decision-making:

That was one of the things that people told me, if you are in charge, don't apologize for the decisions you have to make in order for things to work. I remember having to stand up to [an adversary] in a way that he probably didn't much care for.

Oftentimes, the daily life of SSC's operations appeared to be chaotic and fraught with mini-crises and "flash fires" which required constant "fire-fighting." Thus, a take-charge attitude was apparent in the disciplined manner in which Daphne ran her business and approached new projects. Daphne characterized this impression as a natural response in a dialogical process with the decision-maker and leader:

***Researcher**:*

It seems like are a lot of crises lately.

***Daphne**:*

I don't know; it could just be me reacting to things because you are talking to the owner of the company, so I have to handle everything that comes up. I don't have the luxury of ignoring anything.

These statements leave little doubt that the burden of leadership is a prime factor in Daphne's view on quality and how a team process would ensure a dedication to quality among team members. Thus, the work ethic Daphne looks for in her employees is reflective of her own propensity to give 100% (or more) in her efforts; as she once said to me: "…110% of the work ethic around here comes from me."

Despite Daphne's acute awareness of being the boss, it was an issue she struggled with in terms of what an appropriate show of authority would be in a given situation. Like any leader,

Daphne was often concerned with how her leadership was perceived. Later, during the action-taking and during the adoption of the daily stand-up meeting in the XP process, Daphne wondered how the ritual of standing up in a meeting would affect her power relationship with her employees:

> ***Daphne**:*
>
> *I think the only thing that changes, then, is the way that we roll through things is the fact that we'd be standing up. That's an interesting dynamic. Has there been any research into this... because I'm the shortest person here…*
>
> ***Researcher**:*
>
> *I don't know...*
>
> ***Daphne**:*
>
> *I wonder if there is a psychological component to that?*
>
> ***Researcher**:*
>
> *I could check for you.*
>
> ***Daphne**:*
>
> *It is just a curious thing that ran through my head.*
>
> ***Researcher**:*
>
> *That's fair.*
>
> ***Daphne**:*
>
> *Because when I stand up and they sit down, I look down. I'm the boss. Now, I don't want to look down on people - that's not the point, but I'm going to be looking up to both of my developers during this meeting. (Laughter).*

A casual observer might dismiss Daphne's concerns as trite, but the degree of respect afforded to her authority is very important to Daphne. I do not think Daphne wanted to dominate over her employees as much as she wanted respect for her authority commensurate with the deep commitment she had for SSC's success.

At times, Daphne expressed concern regarding her employees' motivations, skills, attention to detail and work ethic. Daphne espoused the inherent value of the diversity of skills and working styles in the team, but Daphne's *Theory-in-use* belied an expectation that her employees should directly emulate her own qualities. In response to a list of "squawks" [16] regarding the short-comings of a delivered product, Daphne said the following:

***Daphne**:*

I would say on the list, about 70% of the things that [the client] identified are things we have just not been careful about.

***Researcher**:*

Well then that causes you to also consider an internal process for improving your team…

***Daphne**:*

Well, I elected at the beginning of that to forward the email that [the client] had written, even though [the client] did not send it to Johnny, [the client] sent it to me. What I learned with Velma was that if I shared the bad feedback with her, it did nothing for her.

***Researcher**:*

Okay.

***Daphne**:*

It made her an unhappy employee, basically, but I needed her to know that I was taking the heat for her.

***Researcher**:*

Partially appropriate, but then inappropriate if it doesn't encourage learning…

***Daphne**:*

Right, so I felt like I'm going to involve Johnny in this process so that he knows and then I called him into my office and we went through them one at a time and I said, "Okay, this isn't done, why wasn't it?" So we went through that. So I imagine it was a little

[16] This is an aviation term often used in reference to defective items which are entered into an aircraft's maintenance log

uncomfortable for him. So he took them home and said he would do them this weekend so that is why it is on my list today to go back and make sure he did them all.

So, partially due to the experiences of daily work, and partly due to the burden of leadership, Daphne frequently wondered whether her employees would react to issues in a manner consistent with her own values.

5.5.4.2 Skills, Work Ethic and the Individual

Throughout the practitioner-researcher dialogs, Daphne emphatically stated a belief that the skills and experience of her employees were at the same level as her own. Daphne's own work ethic was often projected onto her employees as is evidenced in this dialog:

Ahh... I will say... well, everybody has a different work ethic... I have always been the kind of person who gave a 110%, no matter what, which is why I ended up in business for myself, because when I gave a 110% percent and my employer didn't think it was enough... I was like: "I don't think you understand that I'm already giving you everything I have..." They would just assume that I wasn't...

I guess that's because, probably, 90% of the employees out there in the world are not that way... And, I was like, well, if that's the way it is, then I need to just be in business for myself.

Now, because of that, I have... when I hire employees I have to realize that they're not going to produce the same quality that I do.

I received pieces of advice like, "Hey, you're just going to let them go out and really mess it up." And I've had to call an employee out in a meeting before, for sub-standard work, and they got really upset, and that was Velma... she got really, really upset. But she more than doubled the hours on the quote and it turned out she was going through a personal problem...

And I'm like: "You know what, it would be better for you to tell me upfront that there's a personal problem that's going to impact you, than to deliver me sub-standard work and let me think that you're just delivering sub-standard work..." Everybody says "well, I usually don't let my personal life affect my work..." well, that is not true, your personal life DOES affect your work...

Given Daphne's belief concerning the level of effort to expect from employees, it is no surprise that Daphne would prefer if employees were to emulate her own actions and seek to acquire her skills as it is through her actions and skills that she was able to build up her business[17].

5.5.4.3 Transferring Skills and Skills Cross-Training

Daphne often stated her belief that she had more experience and expertise than her employees; as such, she desired a method which would promote skills cross-training and skills-transfer among the team to bring her employees up to her skill level. Often times, Daphne wanted work to be accomplished the way she would do things. In one of the first utterances captured during our first dialog, Daphne illuminates her desire for a method to transfer her skills to her employees:

> *If we had something in place... this is what I expect you to do. You can do it in your own way, but here are steps... you know, here are the benchmarks along the way.*
>
> *I don't care if you veer a little bit as you go along as long as you come back to the center by the time we get to the end. We have a deadline today; I have one and Johnny has one. And, meeting that deadline is far more important to me than him.*

Furthermore, since, as previously stated, Daphne's believes that her extant methodology is "in her head," whereby she wants to transfer her methods directly to her employees:

> *And even though we haven't written a formal methodology, which I guess is just in my head, and I have conformed Fred to what's in my head...*
>
> *Luckily, he has been trainable and has listened to what I do... Johnny is going to be less conformist and perhaps a methodology becomes more important with an employee that likes to break out.*

[17] Schön (1987) would relate to differences in capabilities between Johnny, Fred, Daphne and Velma

Fred was content with delivering back to me exactly what I asked for so I've molded Fred into my way of thinking, so I guess the methodology is in my head.

But, if I'm ever going to hand over the reins to other people, there has to be something in place that explains to them how that works because they're not inside my head.

We see that Daphne would like for her employees to assume her own good work habits regarding the transfer of her skills. However, Daphne also concedes that any new method SSC adopts must establish a team approach. Whereas Daphne hopes that the methods SSC adopts will bolster a team approach in the long term, Daphne has the following opinion of the near-term:

Right, and I feel like, right now, and I've said this before because I've use the term like "babysitting" and stuff like that... I feel like I've grabbed a hold of everybody and they're kind of dragging along behind me...

There will come a point when they will know what I know and where do we go past that?

Daphne gives a clear indication that she desires employees, or teammates, which are as strong as she is. This has interesting ramifications addressed in the literature on agile methods; many books and articles on the use of agile methods mentions the skill level of the individual developer as among the key factors for agile success.

As Daphne looks forward to the day when her employees surpass her skills, she, in the present, must remain cognizant of the differences in skills and, in most cases, look to mold her team after her own capabilities. The next section discusses this.

5.5.4.4 Replication: Refactoring Team Habits to Align with Leader Habits

Another frequently recurring theme, and a motivation to seek the assistance of a formalized method, is Daphne's desire to impute her work style, habits and ethic to her employees. Again, her *Espoused Theory* considers the value of a diverse team, while her *Theory-in-use* suggests that she won't have peace of mind until she knows that everyone else

does things in a manner which is familiar to her. This interpretation is not made in jest of her disposition and does not pejoratively dismiss her motivations: it is no small risk to enter into small business as it is inherently risky in terms of internal and external factors (DeLone 1988; Everett et al. 1998; Watson et al. 1996). As with any business owner, the risks are ultimately all on Daphne's shoulders as her employees could always pick up and move on to new opportunities. Thus, initially, Daphne's apparent *Theory-in-use* was to seek mechanisms to monitor and control the habits of her employees to ensure maximal quality and to mold her employees into her own work ethic and habits.

In much the same way that a researcher taking the scientific attitude will generate second-order constructs to explain the meanings a practitioner places on phenomenon in her daily world, Daphne may be creating an outsized and idealized model her own qualities in what Schutz (1962, 1967) would call an "ideal type." In this case, to impute phenomenology as it pertains to a researcher's actions, Daphne appears to reflect on her ideals for quality and ethical work, reflect on her own first-order constructs of quality and ethics in a universal sense, and thus develops second-order constructs with respect to her employees which can be considered puppet-like "ideal types" which embody Daphne's wider virtues (Gorman 1977: 61).

If work ethic, style and behavior are idealized in Daphne's own understanding, then her constructed ideal type of these qualities become the expectations Daphne has for those around her. Thus, a common theme which emerged during our dialogs was a strong idealization of Daphne's own quality. Interpretively, this likely stems from the confidence required to successfully run a business, the inherent risk in owning a business, the risk in assuming more liabilities with new employees and a historical and contextual sense that all of Daphne's

accomplishments have been hard-fought. Some examples, from the dialogs, of Daphne's assessment of her own skills and abilities are now examined. On the matter of her work at TTS:

> *I know they hired somebody in after I left, they were not able to hire somebody in for some reason while I was still there. That person ended up not being able to deliver the kind of work that I had delivered and they've gone through two or three people now since I've left and I know that they've never upgraded their website. (Laughs)*

Also, on the matter of her initial relationship with her first strategic partnership with MNM, Daphne had this to say regarding the work she rescued in assistance to MNM:

> *Rex had left them high and dry with two websites in process; one of them being eCommerce on the MIVA platform and the other one being an organization called [a non-profit child-sponsorship organization] which is based right here in Richmond…. I think, looking back on it... Well, Rex had told them it was impossible to implement Joe's design on the MIVA platform and I was like: "No it's not." Style sheets were coming in around then and I knew that you could pretty much do anything with style sheets.*

On the matter of her participation in a software competition project and the interest a state agency subsequently took in the finished product:

> *What I saw was the respect that we got. When we worked for the [a state agency within the Commonwealth of Virginia], I got to meet really important people over there, I got to talk to the guy in charge of the [a state agency within the state of Georgia]. I put together quite a book of research and had the ear of people and even though the [a state agency within the Commonwealth of Virginia] couldn't go with us because we... you know part of their requirements are that it has to be a company, a long-standing company, and know that it's going to be supported over time, and that ended up being a roadblock that we could not overcome... They now have an in-house application that does exactly what we wrote.*
>
> *I guess the ultimate... "The sincerest form of flattery is to have someone copy you?" So, when I called and talked to the lady at the [a state agency within the Commonwealth of Virginia], once this [other] (sic) company was established and could support the product, I spoke with her again and she said: "Yeah, we have it already. (laughs)*

Thus, Daphne continued to extol the high quality of her own skills, knowledge and ability as the cornerstone upon which she has built a successful and growing business. Daphne relies on the

certitude that her traits have allowed her to double her business every year since 2005. Thus, Daphne carries an expectation that these same standards should be met by people in her employ.

Daphne is motivated by more than personal satisfaction in her own high quality; she is also interested that others, particularly current and potential clients, recognize her quality. Daphne makes this clear as she told the story of how she attained her second strategic partnership when she gave a presentation at a Business Partners International seminar:

> ***Daphne**:*
>
> *…it looked like it would be a really good opportunity to get business and... but for me, too, as a new business owner out on my own, it was other business owners... there were a lot of other small business owners who were experiencing growing pains, had to do certain things... hiring their first employees. So, the stories I go to hear from the other business owners were very, very important for me. Very supportive… I had some other people who understood what it was like to be me.*
>
> ***Researcher**:*
>
> *And would you say it did generate business?*
>
> ***Daphne**:*
>
> *Actually, it didn't generate too much business straight out of that group... really, I look back on it and I only had 2 or 3 referrals the whole year. Part of the reason became: it was hard to refer people to me because they didn't understand what I did.*
>
> ***Researcher**:*
>
> *How so?*
>
> ***Daphne**:*
>
> *I got my first business, I... they do something called a 10-minute presentation and after I was able to do a 10-minute presentation I got one of the first referrals. Because I was able to demonstrate poorly-built websites versus effective websites, that suddenly clicked with people that what we build here is different than paying your cousin's nephew's son who took a web design course in high school to build your website. I was able to demonstrate that with a power point presentation and it was very effective... So effective, actually, that one of the guys asked me for a copy of it.*
>
> ***Researcher**:*
>
> *Yeah*

***Daphne**:*

"That's the best 10 minutes I've ever seen!" (Laughs)

Daphne's assertive belief in her own capabilities and a desire to make potential clients understand and value the quality product she offers was a consistent theme which arose in many of our dialogs, regardless of whether the dialog was focused on diagnosis, action planning or evaluation. As Daphne continued her story regarding her presentation to the business group and how she impressed her second strategic partner, she reveals that her belief in high standards of quality is the reason she gets "rescue" projects and why she can charge a premium rate:

***Researcher**:*

Now it is interesting that you say this because I think this tying into previous themes we've worked on with "rescue."

***Daphne**:*

Yeah, that was when... when I started to prepare that first 10-minute; I started to realize that was what I was seeing. I was like "you know, I do a lot of that..." I do a lot of looking at where people had gone wrong and helping them to put it right.

***Researcher**:*

And so this is the increasing nature of your business through the beginning portion of 2007? It's through MNM, and just picking up here and there, where largely it's about taking something that's not so great and making a high-quality product?

***Daphne**:*

Yeah. Because there are a lot of people, from the first time I got in, and I know that I mentioned this before... there's always somebody out there to do it cheaper. So, why is it that we charge what we charge to build a website? And that's a question that I get very often and continue to get to this day, even from my employees. I've had to have that conversation with each of my employees at one time or another. They come to me and say, "you know, my friend needs a website, but I don't think he can afford to pay what we charge." So then I have to go through the conversation and explain to them why we charge what we charge and they realize where I'm coming from. None of them has failed to understand why it is that kind of website is not the kind of work that we want here.

***Researcher**:*

Would you say that one function of charging more is that you might take extra time to make a quality website?

***Daphne**:*

Uhm. Actually, some people might see it that way, but the way I see it is that I refuse to generate anything less than a quality website, so it's got to be a certain amount of work that goes into it.

The previous passage very clearly illustrates Daphne's perceived disconnect, in terms of the expectations of quality, between Daphne and her employees. As Daphne began her second strategic partnership with KWC, the dynamics of the relationship are immediately characterized by the quality and care that Daphne feels SSC brings to a project:

***Daphne**:*

Yeah, all these third-party contractors: they couldn't coordinate everybody and they were getting really frustrated with the one that they had. They had some... actually it was earlier than that... because I met with them in November... because they were supposed... they were working with another company, they called them Data Control, it's not a company I was familiar with... They were supposed to have released phase one of a three-phased project at the end of October and it didn't happen. And then when it did go in, the woman who had built the project was in Australia for a month...

***Researcher**:*

Okay.

***Daphne**:*

So, they put somebody else in on it and he said it was just a disaster...

***Researcher**:*

Oh?

***Daphne**:*

So they brought me in November and there were three companies that bid the project, I do not know who the other two companies were but, we received the job!

As Daphne brings new employees onboard, she carefully guides and instructs them on the particulars and parameters of acceptable quality. In leading by example, Daphne relates the first project she worked on with Johnny and how she demonstrated to Johnny and her client that she was capable of bringing high standards to the work she does. In relating the particulars for a

project completed for a research organization located in the state university she graduated from in Central Virginia, Daphne said:

> ***Daphne**:*
>
> *Yeah. So anyway, they had a... just a God-awful process in place at the [research organization]... it was taking 20-some hours and a 17-page document to describe what it was that they were doing. I took that 17-page document and by mid-January, we were able to build a system that would replace the entire 20-hours of work and my estimate for replacing work is that it would now take less than an hour to do the 20-hours' worth of work. And I remember that when we first demonstrated it that they were thrilled with it but they didn't think there was any way it would take less than an hour and I'm told now that it takes as little as 15 or 20 minutes.*
>
> ***Researcher**:*
>
> *Wow.*
>
> ***Daphne**:*
>
> *It is the most complex business problem I have ever solved for anybody. Unfortunately, I can't tell you anything about it. (Laughs)*

We also begin to see that Daphne is just as motivated by the satisfaction she brings to her customers when her high standards and dedication to quality are brought to bear on the projects she undertakes. By her example, Daphne was able to demonstrate to Johnny the kind of quality she expects. However, throughout our dialogs, this quality was idealized beyond what could be taught by example. When Daphne desires a means by which she could "open up her head" and "dump" what she has into her employees, she desires idealized replicas of herself.

In reference to Schutz' concept of the "ideal type," it appears that Daphne engages in the development of an "ideal type" of her own work ethic:

> *In the process of understanding a given performance via an ideal type, the interpreter must start with his own perceptions of someone else's manifest act. His goal is to discover the in-order-to and because-motives (whichever is convenient) behind that act. He does this by interpreting the act within an objective context of meaning in the sense that the same motive is assigned to any act that achieves the same end through the same means. This motive is postulated as constant for the act regardless of who performs the*

> *act or what his subjective experiences are at the time. For a personal ideal type, therefore, there is one and only one typical motive for a typical act. Excluded from consideration when we think of the personal idea type are such things as the individual's subjective experience of his act within his stream of consciousness, together with all the modifications of attention and all the influences from the background of his consciousness which such experiences may undergo. (Schutz 1967:188)*

Daphne holds an *Espoused Theory* that a "team" process and method for software development is desirable and something any method that the team adopts should possess. However, her apparent *Theory-in-use* holds that "employee quality and standards should be my own." The *Theory-in-use* suggests that Daphne has created an ideal type to which she holds and compares her employees to.

Schutz calls these ideal-types homunculi (little men) which are "…an idealization of those typifications and self-typifications practiced in everyday life, and the homunculi themselves will interact with each other in the same way actors interact in the common-sense world…" (Gorman 1977:62). Thus, much as a researcher interpreting phenomenon observed in the "natural attitude" will create ideal types to illustrate relationships among second-order constructs, Daphne's theories-in-use typically involve the homunculi of ideal types. Daphne develops relationships between these ideal types in the case of her employees, her strategic partnerships and her clients. There is an "ought-to" pervasive in her theories-in-use which are not surprising when the historical and social context of her actions is considered: Daphne is used to giving so much and working so hard, that she constructs homunculi which also rise to her level. In adopting the Reflective-Agile Learning Model and Method, Daphne would need a means of accepting the deviations of reality with respect to the scenes and situations in which the homunculi of her theories-in-use act.

Throughout the cycles of the Dialogical AR Partnership, Daphne most valued the aspects of methods or techniques which confirmed her theories-in-use regarding employee work ethic and quality versus her own. Similarly, the aspects of the XP as adapted by SSC which confirmed Daphne's theories-in-use about clients and strategic partnerships were those favored and those to which energy and effort were afforded for adoption.

Fortunately, Daphne's motives are benign as she consistently defends and demonstrates her high standards as being in the best interest of her clients and the quality of the final product. Daphne expressed a higher goal to serve the customer's needs first and foremost:

> ***Daphne**:*
>
> *I know a lot of people... well, I just have a problem with that, and I don't think that it serves the best interest of the client. It happens in the design field as well, people design these websites, they refuse to give over the original design documents and things like that and basically hold... they register the domain names and refuse to give over the logins...*
>
> ***Researcher**:*
>
> *They have it held hostage?*
>
> ***Daphne**:*
>
> *Yeah, they hold the clients hostage and that is just so wrong. And I'm starting to find that this is part of the reason why people have a bad taste in their mouth working on website projects. I've run into a lot of people who say: oh, the last people I worked with were just a nightmare... so we're trying to remain up on current technology, we're trying to remain flexible enough to suit the client's needs. I have no great desire to hold a client hostage and, hopefully, I will earn their repeat business by providing them good service.*

We see that Daphne holds her reputation in high regard in addition to supporting the values of ethical behavior and customer satisfaction.

5.5.5 Client Confusion: Constituencies, Stakeholders and Team Membership

The preceding characterizations and interpretations are of the practitioners' espoused problems and "ills." However, the researcher noticed other issues emerge over the course of the Dialogical AR effort. The diagnosis phase of the summer of 2008 also uncovered concerns with: the experience of web development in the context of a small team and small shop; the dynamics of organizational culture in a small team and small shop; differences in web development versus other development; issues related to professional identity in the context of a small team in a small shop. All of these issues are very relevant to the problem of method selection; however, many of these issues are by-products of those listed in Table 30. Thus, they are addressed indirectly in the preceding discussion. Another range of issues related to customer, constituent and stakeholder involvement also arose during our initial dialog, and these warrant further discussion.

Strategic Partners

One of the means by which Daphne has grown her business was through strategic partnerships with other companies: both in web development and design and marketing. As SSC developed a reputation for delivering on time, having a "can do" attitude and "rescuing" websites from previous failure, SSC's quality was recognized by others. During the uncertain and lean early days of 2005 and 2006, when the company was just Daphne and, later, Velma, SSC's strategic partnerships were a life-saving leg up into a steady and expanding client list. Thus, it can be argued that SSC may not have prospered were it not for these strategic partnerships. However, these relationships are not without flaws.

Business People Just Don't Understand Technology

A common and recurring concern and issue related to SSC's strategic partnerships was rooted in the fact that, especially with MNM, the strategic partner often estimated costs and documented requirements without a complete understanding of the tools and techniques required to do the job. To the extent that MNM remained the "middle-man" in many projects caused countless problems related to customer communication and satisfaction. This problem was pervasive throughout my time in the client-researcher infrastructure and persisted right up until the end. A strategic partner like MNM was welcome, to the extent that they brought in a steady flow of new business. However, as SSC began to adopt their Reflective-Agile methodology, a primary concern and lament became: "Business people just don't understand technology."

In an early dialog with the entire team of developers, Fred began to characterize what he called third-party development:

> ***Fred**:*
>
> *..at that point, the project manager sometimes gets some red flags... you say "okay, well, is the project on course with what we're given? Can we continue to meet our deadline and our budgetary constraints?" If all of a sudden this content shows up and you have a few extra pages, is that in the budget? Is that going to push the deadline out? What's happening? When you find the little box on the wireframe that says: "wait a minute, this looks like a big extra programming module. Is that all of a sudden with the scope of what we thought we were doing?" So, it was never really… sometimes those things don't get discussed in greater detail coming in... The client has a vision, we think we know what they're asking for and then you sort of refine it as you go, sometimes...*
>
> ***Daphne**:*
>
> *And what he's talking about there on the wireframe is the kind of wireframe that we get in from a 3rd party when we are referred a third-party project.*
>
> ***Fred**:*
>
> *Yeah, it's good to point that out because I think it's such a significant portion of our work that comes from external parties that... there can sometimes be a technical disconnect between what they understand versus what we understand and how it's really going to work, and there's... I mean, just that's a whole other beast to itself, really. You try to help*

them understand what those limitations are so that they can communicate with the client and try to work out those details before it becomes a problem.

Thus, the potential for confusion in the case of clients by way of a strategic partner is perceived by the team as being higher than average. Even as direct and/or third-party clients request initial features or, more significantly, features changes, SSC must manage client expectations due to the client's lack of expertise in technology:

***Velma**:*

"...oh, we'll just slap that in there."

…and it actually takes a lot more hours and they don't know what goes into it and sometimes, based on what they're giving us, we don't even know how time-consuming it's going to be.

***Researcher**:*

So, the client thinks that a rendered web page is magic and something is there just because "it's there," but with your technical knowledge, you realize... shoehorning something in... you've laid out everything to be just so and any kind of anomalous size differential is going to cause...

Further to managing client expectations, Fred relates how sharing technical details with a client is challenging:

I know we are trying to just sort of separate out the techno stuff from the, you now, just what's on the surface, what are they asking for, can we show them that it is doing that, but to some degree I've been ... I've tried that, I guess, on my own. I've just tried to adopt it and at some point it just comes up. You are doing this work a lot and you go, "maybe it is just common sense to show them what is going on." Like if you hire a contractor to do work on your house you see, in replacing a window, that they are removing the framing and are putting up glass… like you see it happen and you know they are coming back the next day to put up sheet rock and you know that has to dry and they come back the next day and paint it -- you see it happening and you know what you are spending your money on by the hour. But in this case it is like thin air - the client has no idea -- we may not even have started development and maybe they have given us all the money.

My point is that sometimes with these less technically savvy clients you can show them the simplest thing and say "okay we are entering..." Like one time I worked on a huge course scheduling system...

More to the point, Daphne laments that her "strategic partners," themselves, do not understand the nature of her business and often times "get in the way of" rather than "get behind" the solutions SSC develops:

> ***Researcher**:*
>
> *Well, I have 10 things here, is there anything off the top of your head?*
>
> ***Daphne**:*
>
> *Hmmm. probably interacting with... and I mentioned this briefly, but interacting with companies that don't know what it is we do. The intangible of IT and I mentioned that before.*
>
> *Earlier in our conversation, I'm finding that [a strategic partner] has advertised things that we did not provide; they have assumed that we have done things that they have not asked for. Things like that.*
>
> *And it's because they don't understand at all what it is we do and have no way of understanding…*
>
> *They do not possess any exposure to the world of software development and the fact that we're not just building websites, we're actually building software, it's not something that they understand at all.*
>
> *I had a meeting with them Thursday before we left so that we could just sit around and talk. I told them, "we've got to get together and just talk about projects that are coming up, projects that have gone by, so that I can listen for buzzwords that you've missed... so that you can say things that I can hear and know about…. That you may have either said the wrong thing, or the right thing."*
>
> *I've got to know that they are selling what it is that we're providing. When I do the consultation, it's different. When I'm working as a subcontractor for another company and they're selling my services, it's a problem.*
>
> ***Researcher**:*
>
> *Yeah, it does sound like a problem…*
>
> ***Daphne**:*
>
> *And the communication of somebody who is not an IT professional serving as project manager is very difficult.*

Regrettably, regardless of all the progress SSC would go on to enjoy during the Dialogical AR Partnership, this problem chronically persisted. Despite these problems, Daphne and the

developers at SSC remained confident that more communication, if not just to “manage client expectations” would be crucial in mitigating for the problems inherent in SSC’s strategic partnerships. The practitioners at SSC believe that open and frequent communication with the client helps to establish rapport and a trust factor that usually makes the difference between a successful and unsuccessful project:

> ***Daphne**:*
>
> *One of the reasons that client rapport became important in this project is because they didn't seem to care when we came back and said... "look, it's going to be additional hours because we're done with the original contract..." and they're like "that's okay... this is how we want it..."*
>
> *So, it seems like because everybody did a good job working on it, Fred had a rapport with the client, the client has accepted the fact that changes after launch cost additional money and they're fine with that. They have not complained once...*
>
> ***Velma**:*
>
> *Well they had a trust factor... you have to have a trust factor to know that we really did follow through with what we said we'd deliver...*
>
> ***Fred**:*
>
> *Yeah, and if that communication is solid the whole time, then you've earned that trust and the work outside of the scope of the contact becomes a lot smoother because it's not the discussion every time. You don't always have the privilege.*
>
> *I'm not saying we need to be their best friends, you know, keep up with them on the weekends or whatever, but it is just to have some sort of connection outside of an email or even sitting across the board table for an hour where you don't really necessarily get to talk. I mean sometimes you sit in a room full of 10 people, you might have 3 of them do most of the talking and then you shake hands and you leave. I mean, what, I don't understand...*

Implicit in the practitioners’ values is a need to establish rapport and trust with their clients via direct and face-to-face communication. So far, SSC has enjoyed their rapid growth while maintaining clientele that are local (in a geographic sense) to their base of operations. In this early stage of diagnosis, it was clear that SSC tried for as much face-to-face interaction as was appropriate and possible.

Attendant to the importance of rapport and trust is constituents and stakeholders involvement in the design and development processes. Of the failures I witnessed during the Dialogical AR Partnership (product was late, product was not what the client wanted, product needs redesigning), in several cases (FMA, CHH, and SLT are cases in point) these projects were jeopardized by a failure to capture and account for the needs of an important stakeholder or constituent.

Problems with stakeholders and strategic clients will be further addressed in the evaluation section of this chapter.

5.6 Action Planning

The previous section provided a deep description, grounded in dialogical evidence, of general categories of issues to emerge during diagnosis. Action planning is a step taken within the Dialogical AR Partnership to address the issues identified in the diagnosis phase with the intent to alleviate the "ailments" inherent within the issues. Both the practitioner and the researcher should agree to plans for action where the researcher is cognizant of the introduction of theory and the specification of learning. When the researcher entered the client-researcher infrastructure, he was very transparent about his theoretical perspective and his contention that, based on the literature on web engineering, small-team and small-shop software development, and agile software development methods, that an agile method would be useful. The selection of eXtreme Programming (XP) is covered in the Chapter Three of this report.

However, subsequent to diagnosis, the researcher now has a basis, grounded in the evidence, from which to gauge the appropriateness of the XP method (and intervention) for the

practitioners to try. In light of the evidence the researcher sustained his confidence that an agile methodology, specifically XP, would be useful in a small-team/small-shop setting. Fortunately, the issues which emerged during the initial diagnosis phase are all addressed by XP. The specific steps within the XP method which address SSC's concerns and issues from diagnosis are shown in Table 31.

Table 31 Addressing Issues from Diagnosis with XP

Issue from Diagnosis	How XP Addresses the Issue
Quality	Acceptance Testing Daily Standup Meeting Pair Programming Planning and Feedback Loops Unit Testing User Stories
Learning	Collective Code Ownership Customer as Team Member Daily Standup Meeting Learn and Communicate Pair Programming Planning and Feedback Loops
Productivity	Collective Code Ownership Customer as Team Member Daily Standup Meeting Frequent Releases of Working Software Pair Programming Planning and Feedback Loops Spike Solutions User Stories
Skills Development and Cross-Training	Collective Code Ownership Learn and Communicate Pair Programming Spike Solutions
Client Relationships	Collective Code Ownership Learn and Communicate Customer as Team Member Planning Game User Stories

Table 31 also provides a refined summary of the issues expressed in Table 30. This refinement was a function of researcher reflection and interaction in the Dialogical AR Partnership.

In mid-September 2008, the researcher proposed a plan of action whereby SSC would undertake a phased adoption of XP practices for the next 8-week period. In order to facilitate this adoption plan, the researcher would conduct a series of presentations and seminars to "teach" the principles and techniques of the method. Additionally, it was agreed that the researcher would visit, at minimum, twice a week on Tuesdays and Fridays. Tuesday was the day that SSC conducted a weekly meeting and Friday was a day Daphne indicated would be suitable for our reflective dialog. In Dialogical AR, the dialog is the principle means by which the social and historical context associated with the actions planned and executed during the Dialogical AR Partnership are understood.

Upon agreement to this course of action, the researcher developed a broad plan, shown in Figure 32, which would divide the possible interventions into two major phases:

Table 32 Major Interventions of the Study

Major Intervention	Phases of the Intervention
Learning and using XP (8 weeks)	• Adopting XP • Adapting XP
Learning and using Reflective Practice (8 weeks)	• Learning the Principles • Ladder of Reflection • Blogs and Wikis • Using the designed artifact: the Reflective-Agile method

As a research method, Dialogical AR is iterative and cyclical in nature. Similarly, as a software development method, XP is iterative and cyclical in nature. These structural similarities were mutually sympathetic where the iterations of Dialogical AR roughly conformed to the cycles of

learning and using XP and *Reflective Practice*. In this sense, it could be said that this research contained either two or four iterations of the Dialogical AR Cycle. It is possible to use the principle phases of the Dialogical AR Partnership, shown in Figure 32 as a means to frame four iterations of the Dialogical AR cycle. Alternatively, Table 32 would suggest that there were two principle iterations of the Dialogical AR cycle. This issue can be addressed by considering Mårtensson and Lee's (2004) careful description of the distinguishing elements of Dialogical AR. We can also look to Mårtensson and Lee's comparison between Canonical AR (Baskerville 1999; Davison et al. 2004) and Schön's (1987) description the process of *Reflection-in-action*.

Using Schön's outline of *Reflection-in-action* is particularly attractive, if not for any other reason than the irony of the recursive reference, in explaining the iterative processes of action planning, action taking, evaluation and learning during this research.

Table 33 Comparing Canonical AR and Reflection-in-action (Mårtensson et al. 2004; Schön 1987)

Canonical AR	Process of Reflection-in-Action
Diagnosis	Unexpected outcome, novelty, is encountered during the course of routine "knowing-in-action"
Action Planning	Reflection is initiated
Action Taking	Meta-Reflection leads to on-the-spot experiment
Evaluation	Outcome of the experiment
Specifying Learning	Further reflection and/or recalibration and update of "knowing-in-action"

As the process of *Reflection-in-action* was certainly equally instructive during the course of the Dialogical AR Partnership, the comparison in Table 33 suggests that *Reflection-in-action* can produce results similar to the cycle of Canonical AR. The philosophical underpinnings of Mårtensson and Lee's (2004) initial outline of Dialogical AR credits the social constructionist

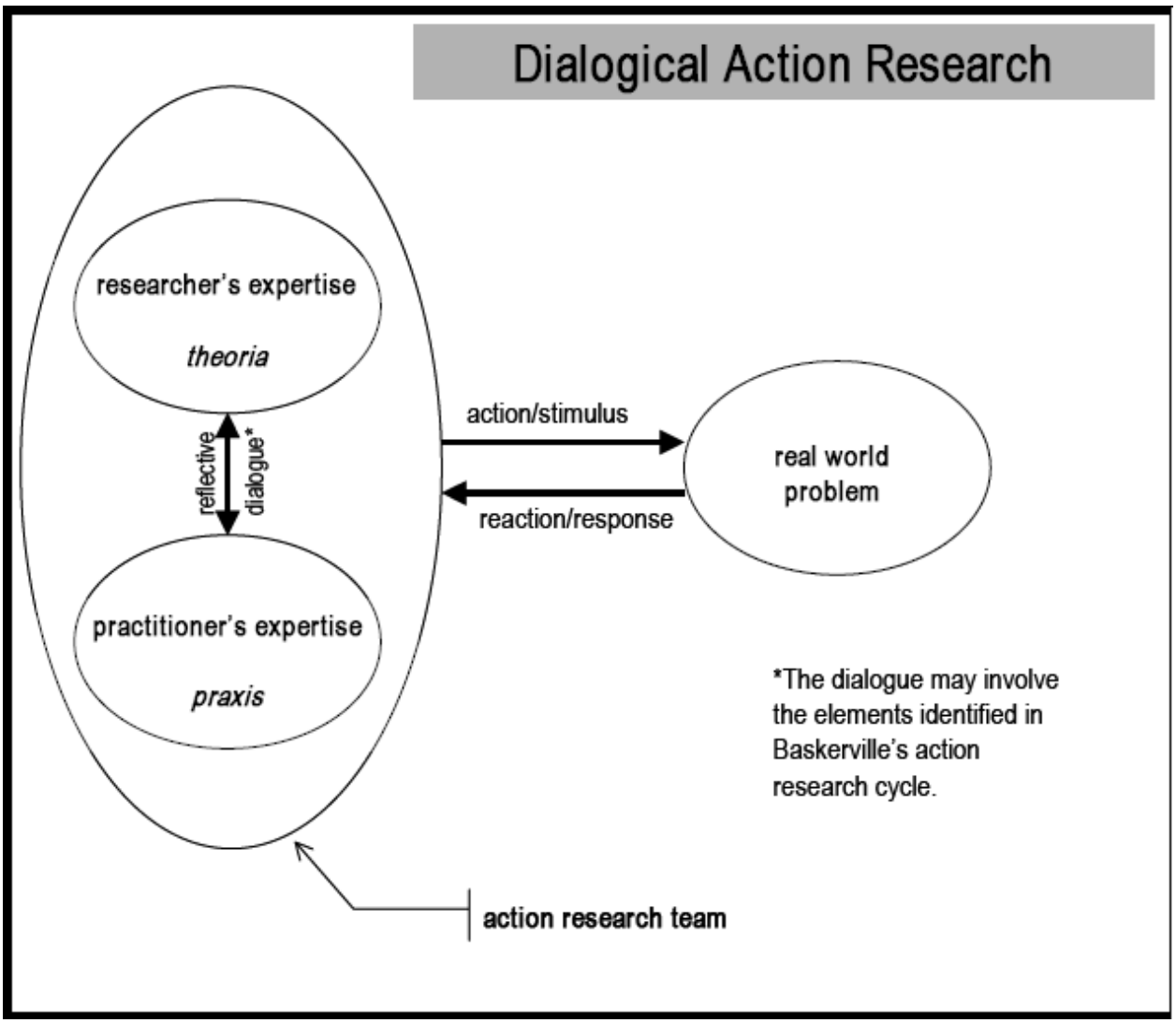
and phenomenological influence of Schutz and the concept of the natural and scientific attitude when creating interpretations of first-level constructs. This guidance, along with Schön's epistemology of *Reflective Practice*, lends support to the concepts of knowledge heterogeneity and knowledge contextuality stressed by Mårtensson and Lee as critical outcomes for the rigor and relevance of any instance of Dialogical AR. Thus, the incremental and iterative steps undertaken in action planning and action taking in this research were not as discrete as is implied in Canonical AR cycle shown in Figure 24. The Dialogical AR Partnership used dialog in order to iteratively and incrementally diagnose and evaluate throughout the period of September 2008 to February 2009.

The dialog also facilitated a continuous learning cycle for both practice and theory. The researcher's reflection, note-taking, constant and comparative coding, participatory action and dialog in the Dialogical AR Partnership incrementally constructed not only the researcher's next moves in the partnership, but his understanding of how the evidence could inform the theories brought into the Dialogical AR Partnership. In the period from September 2008 to February 2009, each new dialog and interaction offered some degree of learning the practitioners faced. Periodically, within this timeframe, the researcher would pause and give a presentation and/or seminar to the practitioners to consolidate their learning and to confirm or disconfirm the researcher's interpretations.

While it may have been easier to present this work in the discrete steps of the Canonical AR cycle, the Dialogical AR process was fluidly iterative in that it contained cycles-within-cycles in much the same manner that the *Release Planning* phase of XP contains iterations-within-iterations (Figure 20). Therefore, Mårtensson and Lee's (2004) diagram of the processes

of Dialogical AR (Figure 38) is more representative of iterations and cycles of this study than those of Canonical AR (Figure 24).

Figure 38 Cycles of Dialogical Action Research (Mårtensson and Lee 2004)



While this was not an expected outcome of this research, the researcher entered into the Dialogical AR Partnership expecting to use the clear-cut phases of Canonical AR. As this report will show, this usage pattern did not affect the outcomes.

5.7 Action Taking

The preceding section discussed the processes of action planning during Dialogical AR Partnership. Since most action-taking centered on the activities discussed in Table 32, this section will describe the phased adoption and subsequent adaptation of XP as well as the introduction of *Reflective Practice* into XP's processes and techniques.

Table 34 Timeline of Action-Taking

Timeline	Action-Taking
September 2008	User Stories Daily Standup Meeting
October 2008	User Stories Planning Game Release Planning Acceptance Testing
November 2008	Iteration Planning Development Activities
December 2008	Pair Programming Unit Testing Ladder of Reflection
January 2009	Blogs and Wikis
February 2009	Reflective-Agile Learning Method

Table 34 above shows the general timeline in which the action-taking activities transpired. SSC would typically build on the previous action-taking steps, and through action-taking and subsequent reflection in dialog, discuss which elements of XP were more or less appropriate for their particular setting and why. The subsequent subsections discuss efforts made towards adopting and adapting XP and *Reflective Practice*.

5.7.1 Adopting Agile Practices and Extreme Programming

SSC's action-taking steps are documented in Table 34. This subsection will demonstrate and discuss SSC's adoption of XP processes. SSC very quickly adopted and found utility in most of the elements of XP they tried. It would be inaccurate to suggest that SSC attempted all of the techniques of XP. It would also be inaccurate to suggest that SSC was aware of all of the techniques of XP or even desired to learn the method in an orthodox fashion. The researcher may have been responsible in encouraging this incomplete attempt at adopting XP as the researcher often quoted the literature on XP which suggests that a partial and tailored adoption is acceptable and still of possible benefit to a team adopting XP (Beck 1999; Jeffries et al. 2001).

The researcher and practitioner agreed that it would be constructive to monitor the progress of two projects while SSC adopted XP practices to determine what effects XP might have on the success of these projects. Daphne suggested that GAB, an eCommerce project coordinated with one of SSC's strategic partners (MNM) and, IMS, an eCommerce/ERP-integration project, would serve as good projects to watch. Fred was selected as the PM for IMS and Johnny was selected as the PM for GAB. As SSC adopted the XP processes, the researcher would occasionally accompany the SSC development team as they met face-to-face with their clients. The researcher took careful field notes as the practitioners tried out the new techniques and interventions in a "production" environment.

In September 2008 and into October 2008, SSC found immediate utility in the *Daily Standup Meeting* and in *User Stories*. According to Daphne, these activities provided the team with increased focus and increased productivity. The dialogs during this period mainly focused on learning, using and adjusting to the processes of XP. *Spike Solutions* were also useful to SSC as they afforded justification for experimentation and testing. From October 2008 into

November 2008, SSC tried pair programming mainly as a means to create *Spike Solutions* whenever uncertainty was encountered.

Figure 39 SSC's Adoption of XP Processes

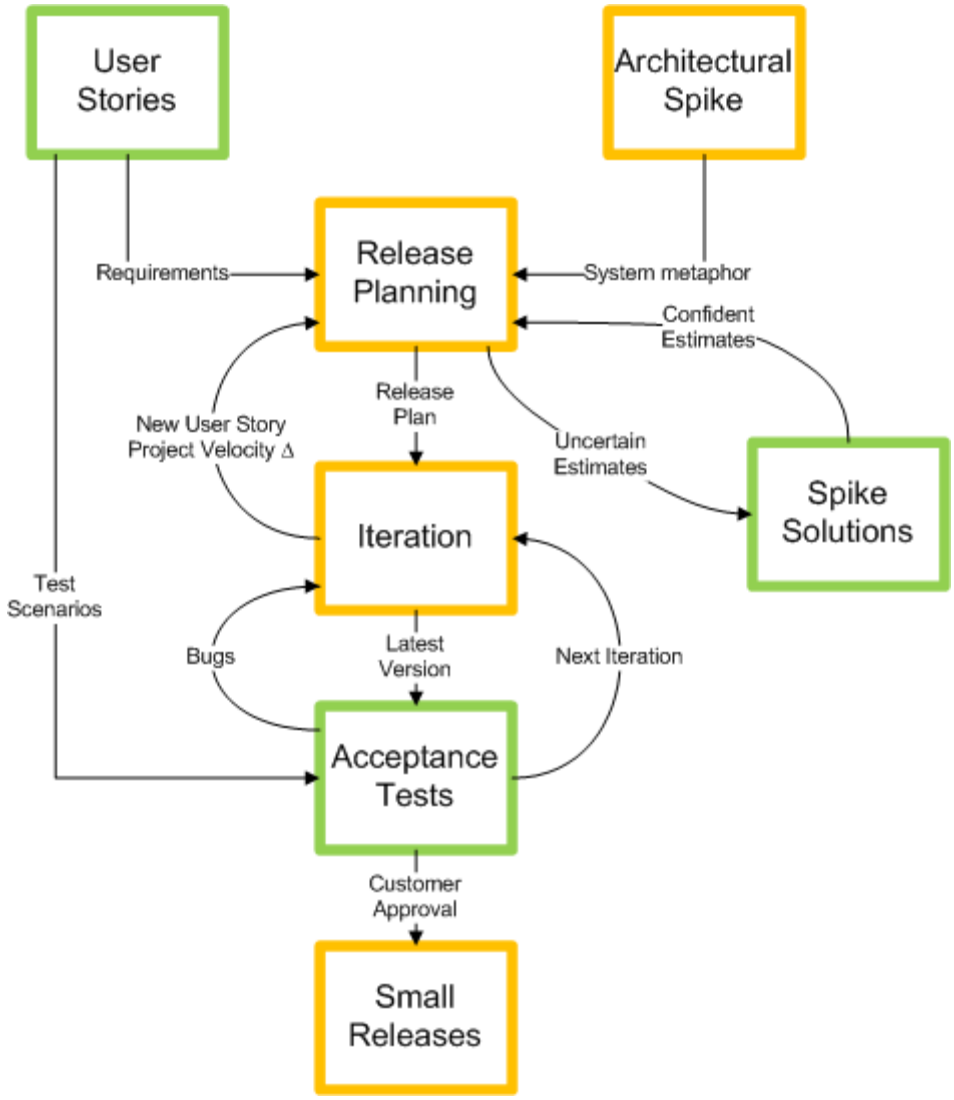


Figure 39 shows the major processes that SSC made an attempt to adopt: the elements outlined in green are those which SSC were able to fully and successfully adopt and the elements outlined in orange are those which were partially adopted or adapted. By November 2008, SSC focused on adopting the XP processes related to *Iteration Planning*.

Figure 40 SSC's Adoption of the Iteration Activities of the XP Method

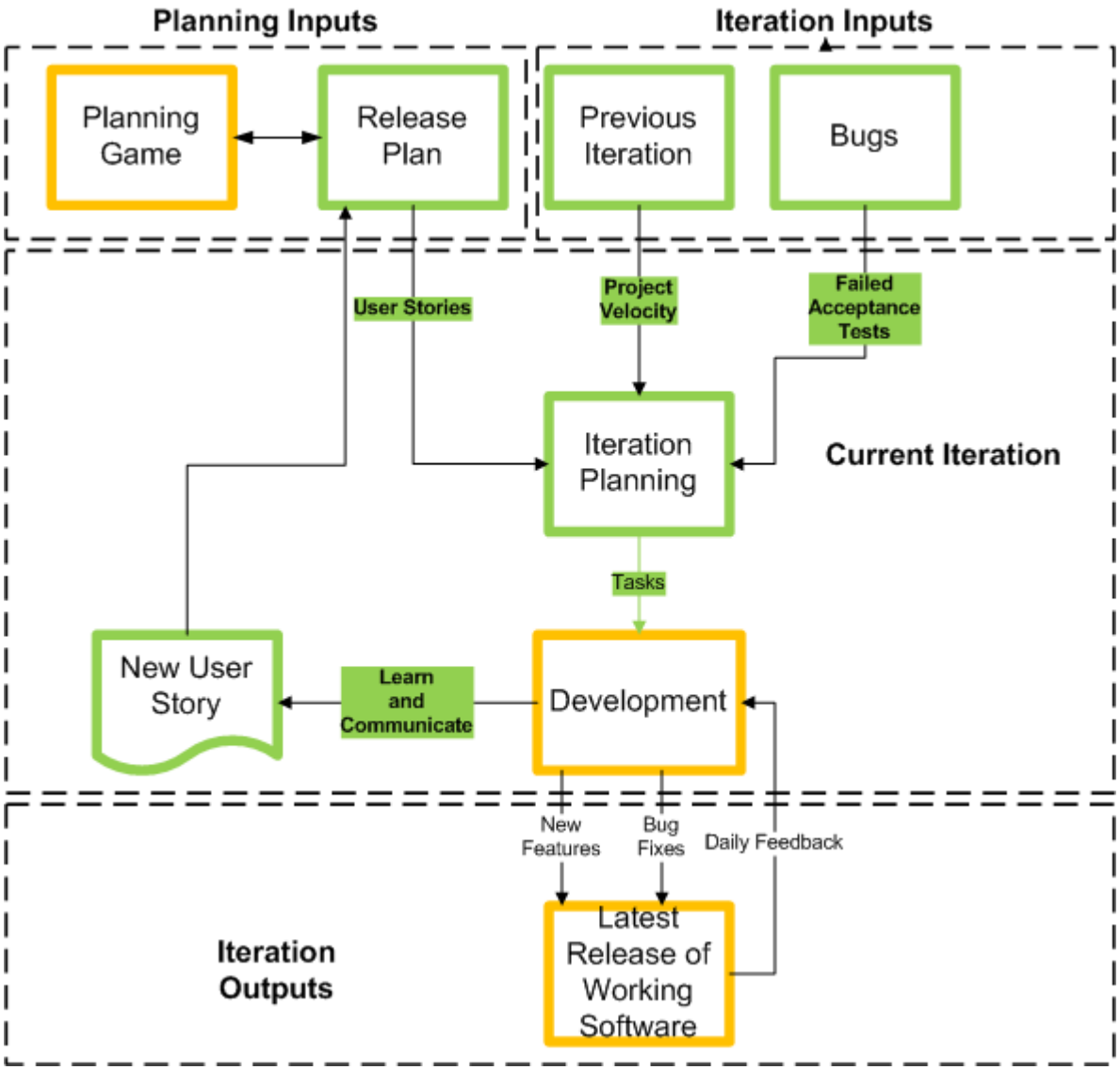
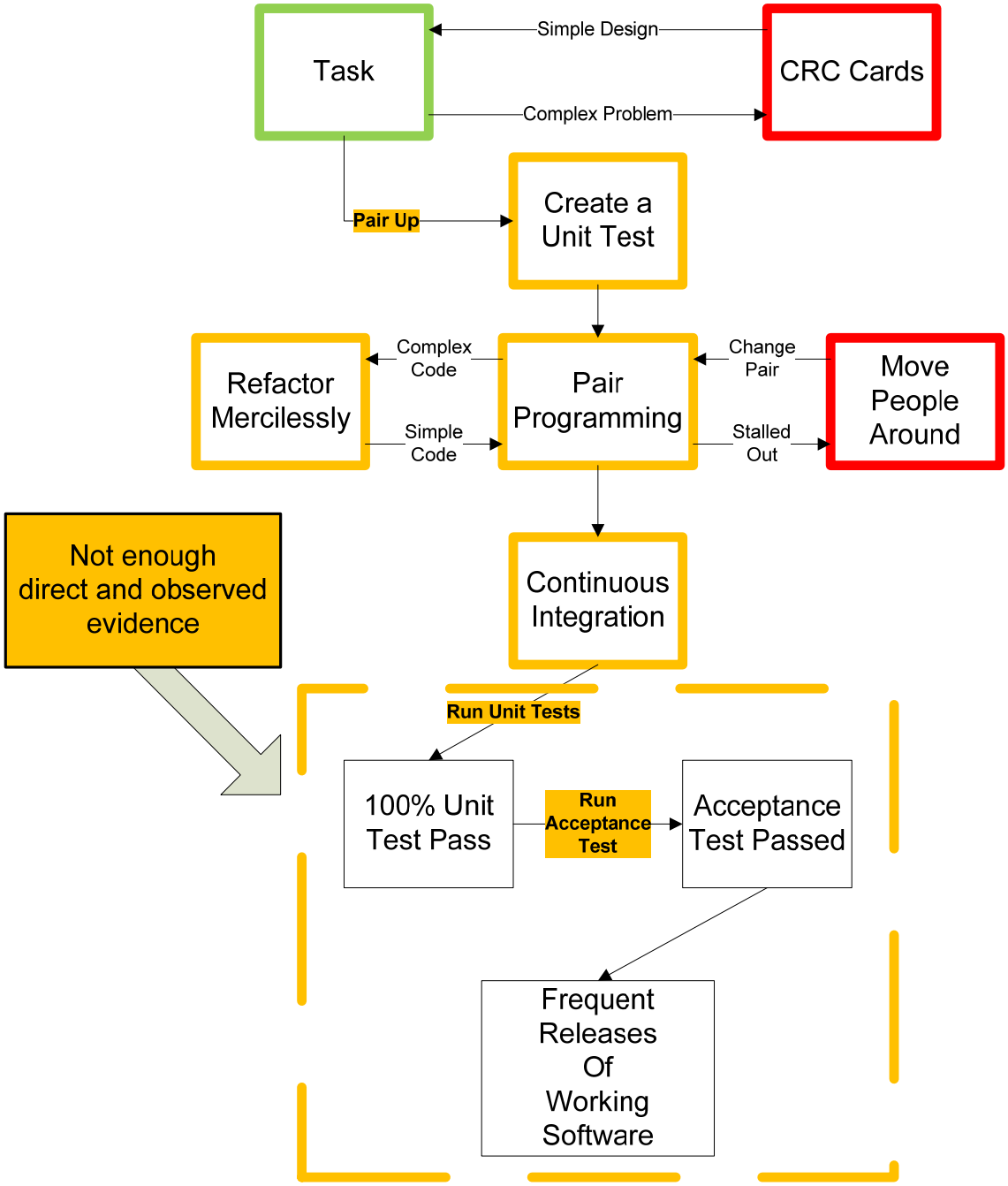


Figure 40 shows the *Iteration* activities SSC were able to fully or partially adapt (the colors have the same meaning as previously stated). By late November 2008, SSC was well into learning and adopting XP and responded with positive feedback concerning the benefits of adopting XP. By the November-December 2008 timeframe, SSC was adopting the *Collective Code Ownership* portions of the XP method. At this stage differences in the nature of SSC's web development projects and other impediments arose which prevented SSC from fully

adopting the XP activities within *Collective Code Ownership*. When the researcher suggested that XP encourages all production code to be written using Pair Programming, SSC responded that it would not be possible to do so.

Figure 41 SSC's Adoption of the Collective Code Ownership Activities of the XP Method



Similarly, during this period, the researcher saw no direct evidence that *Unit Testing* was implemented in the manner suggested by the XP technique. This was despite the fact that the researcher "taught" the SSC developers how to do *Unit Testing* with the ***NUnit*** software package. As shown in Figure 41 (Red means no adoption), the principle of *Move People Around* was not adequately considered or attempted save for the times when two developers, usually Johnny and Fred, would "pair" to create a *Spike Solution*. Lastly, none of the principle *Release Planning* activities, such as *Acceptance Testing* and automated *Unit Testing* procedures, were witnessed or documented by the researcher.

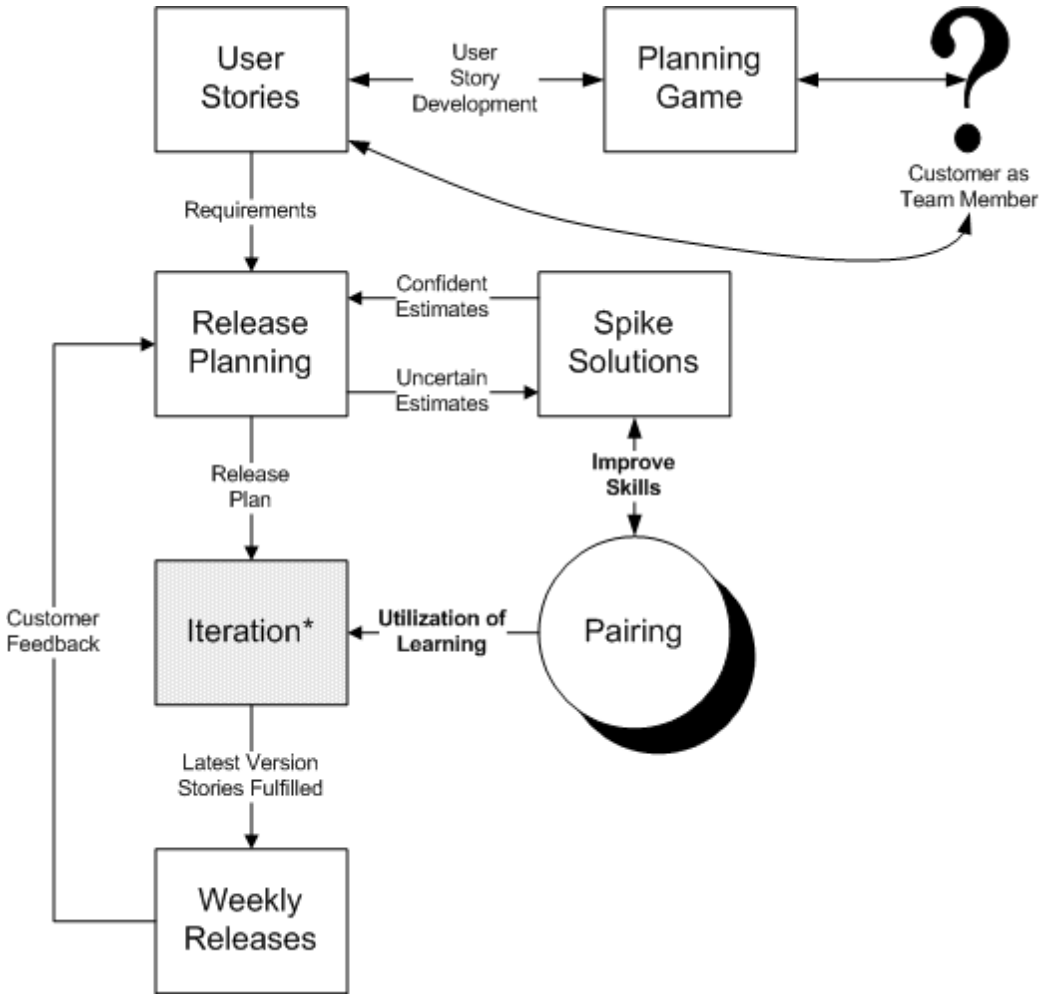
By the end of 2008, SSC had learned and/or tried the majority of XP practices and offered the researcher substantial positive feedback about their successes with using the method and also reported areas where the method was not going to "fit" them. The next subsection discusses areas where SSC had adapted XP to fit their specific circumstances.

5.7.2 Adapting Agile Practices and Extreme Programming

There were several reasons and circumstances which caused SSC to modify or omit elements of the XP method to fit their own circumstances. Among the more challenging aspects of XP to adopt was the concept of *Customer as Team Member*. SSC has, at any given time, anywhere from 20 to 40 projects underway which are at varying stages of completion or maintenance. As new business comes in every week, augmenting the "team" by 20 to 40 members was not realistic. Furthermore, SSC's projects spanned from as little as a week or two to three months (and beyond). Daphne very quickly realized what a challenge it would be to

include a customer on the team in every circumstance. However, SSC did see the utility in this approach in the case of larger clients and with their strategic partnerships.

Figure 42 XP Processes as Adapted by SSC



The impracticality of including a customer as team member on every project had follow-on effects and implications for *User Stories*, *Acceptance Testing* and the *Planning Game*. I often observed Daphne writing *User Stories*, and later *Acceptance Tests*, based on her notes or her memory of a client's intentions rather than capturing these intentions directly in the user's words (and in their own hand-writing if possible). In short, it was readily apparent that SSC would not

be able to follow the orthodoxy of XP. The general model of XP, as adapted by SSC, is shown in Figure 42.

The larger number of simultaneous projects also had effects on *Iteration Planning* and *Release Planning* as SSC was forced to provide frequent releases of working software on schedule which is more rigid than XP intends. Furthermore, SSC, at least from the researcher's observations and in Dialogical AR Partnership dialogs, did not observe any "test first" or "test driven" *Unit Testing*. This was despite the fact that Fred and Johnny did recognize and profess value in the approach – Johnny aptly called it "testing memory." However, it is very probable that SSC will move on towards a "test first" approach to testing over time.

5.7.5 Adopting and Adapting Reflective Practice

By design, agile processes, and XP in particular advocate the use of frequent reflection for learning. However, "reflection" in a general sense is not necessarily equated to the reflection which Schön is referring to. This is not to say that there is not a strong relationship between the common and natural understanding of "reflection" and Schön's *Reflective Practice*; however, Schön's work is meant to operate at the fundamental level of theory.

Table 35 The Ladder of Reflection as presented to SSC

Rungs on the Ladder of Reflection
Designing **(Reflection-in-action)**
Description of the Designing **(appreciation, critique, advice)**
Reflection on the Description of Designing **(What does the description mean?) [Meta-level to description]**
Reflection on the Reflection on the Description of Designing **(reflecting on the dialog itself) [Meta-level to reflection]**

Mathiassen (1998 and 2002), Hazzan (2002, 2003, 2004a, 2004b, 2005) and Tomayko and Hazzan (2004) have each advocated, in both a general and specific sense, for the consideration of Schön's Epistemology of *Reflective Practice* in as a theoretical lens used to understand software development processes and related phenomena. Thus, the application of Schön's work to improve the understanding and effectiveness of the use of agile methods has precedence in the literature. In the particular case of SSCs adoption and adaptation of XP, dialog concerning the outcomes of XP adoption naturally led to focusing on learning and communication. Argyris and Schön's (1974, 1978, 1996), and Schön's (1983, 1987), collective theoretical propositions on organizational and individual learning were increasingly relevant and warranted by the learning and discovery processes during the Dialogical AR cycle.

While the researcher shared the virtues and lessons of this body of theory on learning and reflection, the circumstances and consequences of adopting and using XP also prompted the need for a theoretical perspective on learning. Learning seemed to be among the more useful advantages which XP offered to SSC. Thus, an opportunity to test the researcher's suspicion that a system for learning would be just as effective and important as the actual design and

development steps of the XP method, or any method, had presented itself as the Dialogical AR Partnership unfolded.

To the researcher, the most opportune phases or steps to directly consider practices and techniques representative of the researcher's theoretical perspective on learning and reflection appeared to be in the daily *Collective Code Ownership* activities of XP (see Figure 43). So, by mid-December 2008 and into February 2009, the researcher introduced the concepts of *Reflective Practice* and *Organizational Learning* (including *Single-Loop*/*Double-Loop Learning* and *Model I*/*Model II* behavior) into the practitioner-researcher dialog. Schön's model for learning in pairs, the *Ladder of Reflection* (see Table 35), was introduced into SSC's *Pair Programming* efforts. The researcher also introduced the use of Weblogs (Blogs) and Wiki Wikis (Wikis) to facilitate overt patterns of reflection both in team and individual practice and learning. While these contributions may seem hackneyed, there is a close association to reflection in the history of both the Weblog and the Wiki Wiki.

Figure 43 Introducing Theories of Reflection and Learning into SSC's Daily Agile Processes

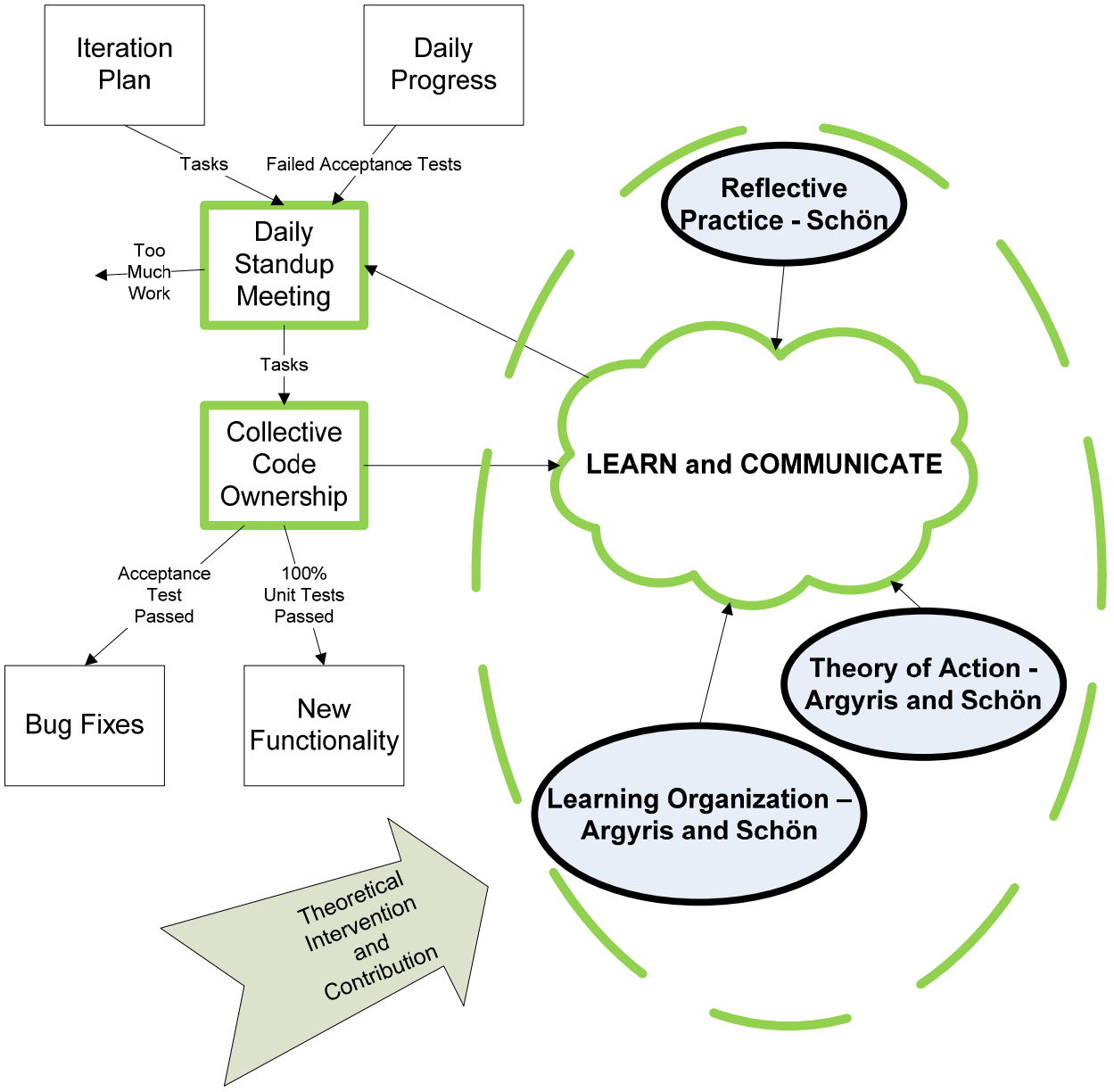


Figure 43 indicates the location in the XP method into which the theoretical interventions were introduced. SSC's daily practices offered the best opportunity for understanding and implementing the ideas and practices of *Reflective Practice* and *Organizational Learning*.

A threat to the success of this last phase of the Dialogical AR Partnership arrived in the form of an unexpected (yet, in reality, quite predictable) crisis which arose in several projects involving SSC's strategic partner, MNM. One of the clients included GAB, which was a focus project for the Dialogical AR Partnership. For a significant portion of January 2009, SSC was not able to devote their full attention to adopting the practices and concepts of *Reflective Practice*. However, during the month of February 2009, evidence from the practitioner-researcher dialogs suggested that SSC did try each of the interventions related to *Reflective Practice* and, as was the case with XP, found more utility in some things over others. It should be noted that SSC's adoption of *Reflective Practice* was not nearly as rigorous as their adoption of XP. Despite this, the evidence from the dialogs and from the researcher's observations and interpretations is sufficient to support the designed artifact and to demonstrate contributions to practice and theory.

This concludes this chapter, which undertook a deep description of the Diagnosis and Action phases of the Dialogical AR cycle. The next chapter proceeds with a description of the Evaluation and Learning phases of the Dialogical AR cycle.

CHAPTER 6 Evaluating and Interpreting the Evidence

The principle aspect of Dialogical AR which distinguishes it from mere consulting is the application of theory-based interventions which lead to the specification of learning based on evaluation and interpretation of the outcomes and consequences of action. This chapter discusses evaluation and interpretation of the outcomes of action taken in the practitioner setting during the course of the Dialogical AR Partnership at SSC. The material in this chapter leads to and supports a discussion of the designed artifact, the Reflective-Agile Learning Model and Method, in Chapter 7 and provides the setting and basis for the conclusion in Chapter 8.

The sections of this chapter are as follows: First, evidence from the evaluation steps across all iterations of the Dialogical AR cycle is examined. Next, there is discussion regarding the philosophical assumptions from which the empirical evidence is evaluated and analyzed. Furthermore, evaluation and interpretation of the evidence is examined for validity according to established and accepted criteria for qualitative and interpretive work and for Canonical AR. This is then followed by a discussion which summarizes the action research setting with reflection on the evidence. The presentation, analysis, interpretation discussion of the evidence in this chapter then provides the basis for the designed artifact, which is discussed in the penultimate chapter.

6.1 Evaluating the Outcomes and Consequences of Action

The evaluation phase of Canonical AR calls for the action research team to reflect on the outcomes of action (the interventions) and assess their worth relative to the diagnoses. While it would be easy to conclude that XP was successful for SSC, which it was, such a conclusion would not facilitate the most important aspect of action research: the specification of learning. The inputs to learning are the interpretive reflections on SSC's diagnosed problems, both before and after introducing interventions meant to address those problems.

This section continues with a descriptive account, drawn from practitioner-researcher dialogs, which illustrates reflective evaluation on the outcomes of actions guided by theory-driven interventions. The researcher builds a case which demonstrates the efficacy of Schön's (1983, 1987) *Reflective Practice* and also explores which of the XP elements had the greatest effect on the practitioners' problems and issues. To this end, this section proceeds as follows: first the outcomes of the interventions, indicated in Table 31 and including the early and obvious successes in XP adoption, are discussed; this is followed a discussion regarding the philosophical grounding for the researcher's interpretations; next, the emergence of a guiding philosophy for agile methods is related to the evidence from this study; then, it is suggested that all of SSC's issues are reducible to or related to learning and communication; last, the evidence is used to support the need for the designed artifact: the Reflective-Agile Learning Model and Method.

6.1.1 Evaluating Productivity in Early and Sustained Successes

As the practitioners at SSC lacked any documented method at the time the researcher entered the Dialogical AR Partnership in the summer of 2008, it is difficult to determine, at the onset of the partnership, whether XP would be welcomed by the practitioners or whether XP

would have any effect. In evaluating the impacts of XP, it is easy to see how success could be read as a "false positive," whereby the benefits of the selected method were not necessarily due to the particulars of XP, but rather the increased structure that introduction of any process or methods would incur. None-the-less, as the search for and adoption of a method was the genesis of the Dialogical AR Partnership; it was incumbent upon this partnership to find a method. As Chapter 3 discusses the rationale for selecting XP and as Chapter 4 discusses the rationale for adopting the theoretical lens assumed by the researcher, this section describes, in light of the dialogical evidence, what degree of success the interventions achieved.

On the matter of productivity, Table 31 indicates that this issue was addressed by the following XP activities: *Collective Code Ownership, Customer as Team Member, Pair Programming, Frequent Releases of Working Software, Planning (Release and Iteration) and Feedback Loops* and *User Stories*. This section will evaluate which of these had the greatest impact.

In October 2008, SSC immediately capitalized on the success of *User Stories* and the *Daily Standup Meeting*. These two steps of the XP process would continue to benefit SSC for the duration of the Dialogical AR Partnership in many areas, including productivity. Additionally, the *Frequent Releases of Working Software* and *Customer as Team Member* had some degree of impact on productivity.

Over time, as SSC used their adaptation of XP on a daily basis, the team indicated in dialogs that they had forgotten the particulars of their processes before adopting XP. Also, the team began to use the techniques in XP for actual team building and shared context. However, the researcher had developed concerns as to whether XP simply reinforced a hierarchical

structure from Daphne downward. By December 2008, Daphne had the following to say about the effects of *User Stories*:

> ***Researcher**:*
>
> *User Story gathering and requirements gathering… it's clear to me that you've adopted these in your culture.*
>
> ***Daphne**:*
>
> *Well, it's just amazing to see how it increased everyone's productivity and buy-in into the job because they were not sitting around wondering what to do, they had a list of things to accomplish and if, like Fred said, if the client puts him on hold, he just went and got other cards.*

Productivity was an important issue to emerge during diagnosis and Daphne's words convey a sense of satisfaction in how productivity had increased. By mid-November 2008, Daphne reported in the course of dialog, that she noticed a significant increase in billable hours in the previous month with an average increase in 3.5 billable hours per developer. Thus, the overall percentage of developer billable hours, for the first month that SSC had used XP, had risen appreciably. It is apparent from the transcriptions and from the researcher's field notes that this initial measure of success was attributable to SSC's use and adaptation of XP. Additionally, this initial success convinced Daphne of the utility of learning and using XP further and it was evident to the researcher that, from this point onwards, the practitioners had let their guard down and had really begun to embrace Agility. Subsequently, this accepting attitude contagiously evolved into an accepting attitude towards the theoretical interventions introduced from December 2008 to February 2009.

With respect to productivity, there could be no better evidence of success in using XP than its effects on profitability. In February 2009, near the end of the Dialogical AR Partnership, Daphne shared a quick sketch of her cash flow statistics for 2008. As Daphne shared this

information, she used the superlative "that's incredible" in reference to her belief that learning, adopting and adapting XP and the Reflective-Agile Learning Model and Method had significantly improved her profitability through greater developer productivity.

Table 36 SSC Revenue and Expenses for 2008

Description	Amount
Monthly Sales Average for 2008	$21,295.15
Monthly Sales Average First Six Month Period	$18,168.37
Monthly Sales Average Second Six Month Period	$25,731.83
Expenses Average for 2008	$19,980.56

Table 36 is a summarized accounting of SSC's sales and expenses for 2008. It is clear that SSC made a profit at the end of the year, but it is also clear that SSC was more profitable, due to an increase in billable hours, during the Dialogical AR Partnership in latter half of 2008. Thus, a preliminary conclusion can be made that the interventions introduced during the Dialogical AR Partnership brought SSC into profitability for the year. This was corroborated directly by Daphne in the transcripts: during diagnosis she spoke of inefficiency and lost profits, however, in dialog during subsequent iterations of the Dialogical AR cycle, she reported improvements in billable hours.

Another remarkable outcome was observed as the team adopted *User Stories*. While *User Stories* were used to the intended effect, there were some unintentional consequences of use. It seemed as though the practitioners struggled to adopt the technique in an orthodox manner. Daphne's earliest efforts to adopt XP involved retrofitting *User Stories* into existing and underway projects rather than capturing these stories in the client's own language. This was not an issue for Daphne as her apparent motivation for *User Stories*, at the onset, was focused on

improving employee productivity; initially, Daphne saw *User Stories* as a new tool to monitor developer output and activity at a finer granularity of analysis.

> ***Daphne**:*
>
> *I was able to see ahead of time when they needed my work to be done and complete it on time and I think that is where you see my increased billable hours. I was forced to produce the work.*
>
> ***Researcher**:*
>
> *That's great and you are reviewing productivity daily and you are leading daily. So, at a very early stage, you are doing better with this than I probably hoped for…*
>
> ***Daphne**:*
>
> *Basically I break everything down into what you've asked for and what it will cost…*

By this same reasoning, Daphne valued the *Daily Standup Meeting* as it afforded her a daily, rather than weekly, monitor of productivity and project velocity:

> ***Daphne**:*
>
> *The other thing that I have been doing, just from the business aspect, is that I have been monitoring productivity.*
>
> ***Researcher**:*
>
> *With User Stories? Are you attaching times and actions to everything?*
>
> ***Daphne**:*
>
> *Well, yes, in an overt… because we record under our time-tracker on a weekly basis -- I had been looking at those numbers monthly, I got to where I was looking at those numbers weekly and now I am looking at those numbers daily because of the standup meeting.*
>
> *The one thing that I have seen is an increase in productivity already.*
>
> ***Researcher**:*
>
> *Oh?*
>
> ***Daphne**:*

I was averaging 3.5 billable hours a week and last week I had 12. Not only that, our percentage of billable hours went from 63% in September and that's the month's average because we had to complete work there, to 81% over the last two weeks.

think that the daily standup meeting is really helpful from the aspect that it helps me see what I need to have available for the employees so that they can keep moving; it prevents them from having to wait on me.

Additionally, Daphne valued the *Daily Standup Meeting* as a means for learning and communication. The developers also valued the *Daily Standup Meeting* for frequent and reliable access to Daphne for frequent feedback and direction. Thus, in the initial stages of adopting XP, success was related to Daphne's, as the lead developer and company owner, maintaining a top-down relationship with her employees.

As *Pair Programming* was largely rejected for the purposes of developing production code, there was also little impact that *Collective Code Ownership* had on productivity; Figure 41 indicates that the adoption of *Collective Code Ownership* was minimal. *Frequent Releases of Working Software* did improve productivity to some degree as this step increased output such that SSC's clients were receiving working software earlier and more often. SSC adapted XP to *Frequent Releases of Working Software* on a weekly basis, which prompted regular customer interaction and, thus, regular feedback for *Release Planning, Iteration Planning* and a measure of *Project Velocity*. Initially, there was some resistance to *Frequent Releases of Working Software* largely due to technical and logistical difficulties inherent in web development and deployment. However, the researcher suggested that SSC use this obstacle as an occasion to create *Spike Solutions*. Once considered in this light, *Frequent Releases of Working Software* was seen as a means to facilitate learning. The face value of *Frequent Releases of Working Software* was also apparent:

Researcher*:*

If you are making your customers happy frequently, then you won't have any surprises, they won't have any surprises and, furthermore, there's a good chance it doesn't take as much work as you think to make them happy.

***Daphne**:*

We made a decision related to that last week. We were sitting there talking about what the customer meant by something. We were like "let's give them what we've got and see if they like it." (Laughter)

Most of these XP processes and concepts are related in that their functions feed each other. For this reason, as *User Stories* met with greater success so too did the concept of *Frequent Releases of Working Software:*

***Researcher**:*

Now, let me ask you this... are you saying that you are going to be able to develop working software at the end of that stack?

***Daphne**:*

Well, we have already demonstrated to the work in e-Commerce software.

***Researcher**:*

Okay.

***Daphne**:*

What they want to see is that their products are not in there incorrectly because the categories are messed up. They want to see their major categories in there. They want the customer to be able to click between major categories.

…both of those Spike Solutions which are related to that and we need to get these sub-categories in there. So all of this is importing the data we have, combining it with their data and giving it back to them the way they want to see it. I don't see why that can't be done in the course of three days.

***Researcher**:*

I'm not agreeing or disagreeing with you -- at the end of those [User Stories] do you have working software and do they all three have acceptance tests and you can demonstrate that they pass?

***Daphne**:*

We have to develop acceptance tests. I mean, I kind of, it is kind of obvious…

I think that it is great to be able to show the client something that works every week. I think it is fantastic. [A strategic partner] was thrilled to see [a client] up last week, a week before the due date.

6.1.2 Evaluating Issues Related to Quality

In the initial diagnosis, and throughout the Dialogical AR Partnership, Daphne was very focused on quality: ensuring it, proving it and sustaining it. There is no mistaking the premium Daphne placed on quality; it was the hallmark by which her own reputation had developed and served as the primary basis for SSC's business growth. Daphne was not willing, at any time, to sacrifice quality; not for productivity, not for learning, not even for customers.

Table 31 indicates that quality was addressed by the following XP activities: *Daily Standup Meeting, Pair Programming, Frequent Releases of Working Software, Planning* (Release and Iteration) *and Feedback Loops, Acceptance and Unit Tests* and *User Stories*. This section will evaluate which of these had the greatest impact as they relate to quality.

By January 2009, Daphne felt that her team's new processes had provided the quality assurance she desired:

…Otherwise, we just have our word against someone else's word that we do a better job. I think the fact that we have some pieces in place that actually provide structure and organization help people feel like we do a better job, up front, before we even get started.

Adopting and adapting XP improved quality as it imparted greater focus to the developers and improved their attention to detail. As the practitioners tried, adopted and adapted *Unit Tests* and *Acceptance Tests*, Daphne observed improvements in the quality of the developers' code:

***Researcher**:*

And, you stated to me to some degree the benefits you expected to see, which means that if you are not totally seeing them yet, you still believe they are on the way. Can you tell me in your own words, any benefits you see in the iteration part of this technique?

***Daphne**:*

Well, Johnny said some time was saved, taking benefit of my experience and seeing what I did, um, I would think that unit testing makes code more portable over time. One of the things that Johnny and I discussed about him doing unit testing was that we felt that [a client], who comes from the corporate world, would get the "warm fuzzies" if she knew that we had written four unit tests for a piece of code that we put into place. He is launching a change of address unit and they asked for changes, so we didn't launch it last week but they know that is because they requested changes. So, I told him to put a unit test in it and I sent [a client] back an email saying that we had four unit tests planned and stuff like that.

So, her response was just very happy, so I feel like it gives the client "warm fuzzies" knowing that there is some testing going on, some formal aspect of testing is happening.

I think getting [the client] to say what she expects to see the system do and to hear her say "Yes, that is what I wanted." That falls into the category of acceptance testing.

***Researcher**:*

Yes, it does.

***Daphne**:*

Hearing her say that means that we have accomplished the goal that we had set out to deliver.

I would imagine that that means, you know, the client is happy.

***Researcher**:*

She has working software and she is happy with it.

For Daphne, there could be no greater mark and assurance of quality than positive client feedback and, due to their new quality assurance process and systematic testing, Daphne got that feedback.

Over time, as the practitioners continued to adopt and adapt XP, they had developed a coordinated approach where *User Stories* and *Release Planning* were connected to *Unit Tests* and *Acceptance Tests*. As this is the intent of XP, this pattern is not unique; however, these steps

presented the reliable processes for quality assurance which Daphne had originally sought in a new method:

***Researcher**:*

So it's really improving your processes.

***Daphne**:*

Yeah, did you get the feedback from Johnny and Fred on the unit testing?

***Researcher**:*

Yes, what do you have to say about it?

***Daphne**:*

Fred and I discussed this and I told him, "Well, you and I, all along, and I know Johnny does this too, as we are writing code we test it, we are writing a method, we do that."

And what he and I talked about was maybe we save these tests and start storing them... What I started doing with the project I'm working on doing down at [a local state university] is every time I ran a test to the screen I took the method and I moved it down to the bottom and put it in a region in the code called "testing methods," and I prefaced all of them with the word "test."

***Researcher**:*

So it's not a huge stretch. And so, you are picking those tests up and putting them into a separate place?

***Daphne**:*

Well, and that's what Fred and I talked about. I said, "For now, since I missed the training on NUnit… ;" I was saving my testing methods because I know that the tests proved a concept along the way related to what I was trying to do. And Fred was explaining to me how he put them into a separate class and I said, "Well, okay, I like that idea." I probably put them in a public class right beneath the class so that they are in the same code file so that I can look back and forth between them.

***Researcher**:*

That'll work.

***Daphne**:*

So I said, "okay, I can do that." So I'll separate them out into another class and he said to get NUnit to work with them you just have to put in some stuff in front of them. So, I told him, I recommended to him that he start saving some of those testing methods

because I said "how many times you run a test, deleted it out of your code and realized that you needed it again?"

***Researcher**:*

Right, and the conversation we had and the way we discussed testing is that the whole purpose of these tests is that they are part of the code and cumulatively over time – Johnny put it very well, he said "it's a testing memory."

***Daphne**:*

Okay.

***Researcher**:*

And it is. It's actually your organizational learning manifested in tests.

***Daphne**:*

Okay.

***Researcher**:*

Because when you revisit code, you don't have to wonder... the example I gave is this:

If you go back to code you worked on six months ago, you may not recollect as much, but I know that during the time you were actively working on it, you knew every nook and cranny. You know all the reasons why this is that way and that the other.

Well, these tests manifest your memory and when they all pass, you don't have to worry about it. It doesn't mean new problems won't arise… say new problems show up with FMA, okay? Maybe your service contract is over, I don't know, but let's just say they reopened it because there was a new problem and you had five new things and five new tests, on top of ten old tests, now you have 15 tests. So, it's accumulative knowledge of all the ways you thought to break and stress your code, all of the problems you faced. So, when things change you can feel good about that. So, if you can swing the time, I strongly encourage you to pair with one of your developers so that they can teach you NUnit.

***Daphne**:*

Well, one of the things we talked about is to upgrade Visual Studio before the end of the year. So, one of the reasons I was collecting those testing methods was so that we can go through the process of "here is our testing methods, how do we use that in 2008?"

***Researcher**:*

Right. That would be a good internal spike… Johnny and Fred know enough about NUnit to be dangerous and Visual Studio is just about the same as NUnit for unit testing.

By the conclusion of the Dialogical AR Partnership in February 2009, the developers had extensively adopted both *Acceptance Tests* and *Unit Tests* in a more systematic (but not complete) manner than was apparent a few months earlier in December 2008. During a final "debrief and retreat" meeting and dialog in February 2009, Fred shared the following about *Acceptance Tests*, *Unit Tests* and *User Stories*:

> ***Researcher**:*
>
> *I think I underestimated you… basically, in some places there are things you've done that I may not have directly observed enough, so I need to get corrections from you…*
>
> ***Fred**:*
>
> *Alright, let's do that.*
>
> *I have a point. Acceptance tests, you mentioned... we talked about having small releases and how...*
>
> ***Researcher**:*
>
> *Yes.*
>
> ***Fred**:*
>
> *For the most part, it works for us to do it like a weekly release.*
>
> *I've actually found, to the contrary, that showing the client almost per user story... Like, sometimes a card just has: "client sees product detail page and it looks such and such a way," whatever. I'll just give them that and say step-by-step are we giving you what you are asking for?*
>
> *Because sometimes you may discover along the way something that affects the steps that will come after that and it is good to hit those points... and go, "yep," check it off, put the card in the box, get the next card out, show it to the customer, put that card in the box.*
>
> ***Researcher**:*
>
> *So it sounds like you are doing acceptance tests - here is how to know you are; and, if you can say "yes" to this, then, I'll amend my analysis. They (acceptance tests) are tied to the User Story, which you just shared; and they are expressed in the client's language. So, they express how they'll know.*
>
> ***Fred**:*
>
> *Right.*

***Researcher**:*

There's no other magic to it. They express what the client wants; then you refine… in the interim you make your tech documents, your tech requirements, and all the like. At the end of the iteration -- in their words – and from back in the user story, when they were imagining what they wanted...

***Fred**:*

Yep.

***Researcher**:*

... "this is what you said at the time, this is what you see." Now they can change their mind at that point, and that's why you have an agile, change-oriented process.

***Fred**:*

I absolutely have experienced that.

While a focus on quality is evident in XP in processes, the continuum from *User Stories* to *Unit Tests* and *Acceptance Tests* is perhaps the most relevant to quality (Jeffries et al. 2001: 138). Because the practitioners did not initially warm to the testing aspects of XP, most subsequent interventions focused on adopting processes for learning and communication. However, by February 2009, the practitioners' "buy-in" to testing was emerging; this was apparent once Daphne realized that customer satisfaction was directly tied to the testing processes in XP: from the *User Story* to the *Acceptance Test*. As the mutually-designed artifact, the Reflective-Agile Learning Model and Method, began to take shape, it was apparent that the interventions related to learning were also directly linked to quality. Daphne shared this realization as she related an experience with *Acceptance Tests* and the team's use of their Wiki:

***Daphne**:*

I know you wanted me to mention the acceptance testing…

***Researcher**:*

Yes, please.

***Daphne**:*

I did a couple of things with the process. One is I wrote down all the spikes (Spike Solutions) on their own card and created pages for these in the Wiki. So I would have opened in Internet Explorer. Like Fred said, if I couldn't stop and put it in the Wiki right then, I knew I wanted to come back to it later. There have been a lot of times I knew I wanted to come back to something that I looked at six months ago but I couldn't remember where I found it, so I don't want to lose this knowledge. And one of the things that I did was an Excel export and that is something both of them are getting ready to tackle. It might be easier with Infragistics, but we'll wee.

The other thing I did is there's an exact card where I sat down and asked [the client] what the acceptance tests would be and, basically, she wanted to make sure that the tables in the report looked exactly like they do in... well, she just gave me a sample of raw data ...

***Researcher**:*

Okay.

***Daphne**:*

.... this is my example to go by and the acceptance test was the table generated by my automated mail merge which had to look exactly like this. It had to have the same numbers, the same layout, everything. They were having to build these up manually and I'm doing it with a piece of code now and when we ran through the project the day I delivered the project, we sat through and we ran through these things and I wrote down things that she wanted me to check that were going on behind the scene.

***Researcher**:*

Oh, good.

***Daphne**:*

... so I went back and verified. She gave me this as an example that had to be output ... it does exactly this. That was the test for that one.

***Researcher**:*

So it is in her language and it is attached to the user story?

***Daphne**:*

Yeah.

***Researcher**:*

That's it, that's acceptance testing.

Daphne:

... and then, for the validation form, right there, she wanted to make sure that each of the field rules were validated; since it was too many rules to fill out the card, we wrote a validation sheet and when we went through we basically had this sheet sitting there and we went through and tested the form.

What is most apparent in this dialog is the degree to which Daphne recognizes how *Acceptance Tests* lead to quality and how her own personal process of reflection and learning, manifested in the use of the team Wiki, was becoming more habitual.

From the dialogical evidence, it is apparent that the most relevant XP processes related to quality were *User Stories*, *Unit Tests* and *Acceptance Tests*. The *Daily Standup Meeting* also had a significant impact on quality as Daphne indicated that team was much more focused as a result of the *Daily Standup Meeting* as it relates to learning and coordination.

6.1.3 Evaluating Issues Related to Client Relationships

As the practitioners adopted and adapted XP, their focus shifted to issues related to client and strategic partner learning and communication. January 2009 was a mixed month; while the team was progressing in their adopting, adaptation and use of XP, several clients, all third-party clients through MNM, were very unhappy with their websites. It was apparent that the inefficiencies of SSC's strategic partnership with MNM were the root cause of the clients' dissatisfaction. In the case of a newer third-party client arranged through MNM, Daphne was able to provide better estimates with *User Stories.* Her estimates from *User Stories* were more accurate than MNM's initial estimate:

***Researcher**:*

[A well-known hotel chain]? That's a new client?

***Daphne**:*

Yes, I wrote the User Stories and re-wrote the proposal and [MNM] had already submitted the proposal. Taylor emailed me back and she said, "Yeah, your quote is $4,000 higher than we quoted." And I said, "Well, maybe that's why you are losing money on your web projects."

They didn't get the whole story.

***Researcher**:*

Okay.

***Daphne**:*

Well, I did get the whole story and she said "well, it is clear where they deviated from the original proposal so I'll be able to say these are additional things that came up in the meeting and they'll either approve it or they won't."

But I think the difference there is we are not going to lose on that project because [MNM] let me quote it up front.

***Researche**r:*

So elaborate on that please; what you are telling me is that based on some of your new techniques you felt that you were able to capture a more complete set of user requirements?.

***Daphne**:*

Yes.

There were at least three major features that were not captured in the original proposal that [a well-known hotel chain] wants on their web site.

***Researcher**:*

So your new agile method is superior in this case because it captured more of the customer's needs? Because you were more engaged with the customer?

***Daphne**:*

Yes, and yesterday, following the meeting, once [male client] had left the meeting with SLT, we sat down [female client 1] and [female client 2] and gathered user stories on their eCommerce site and several things came out that had not previously been captured.

***Researcher**:*

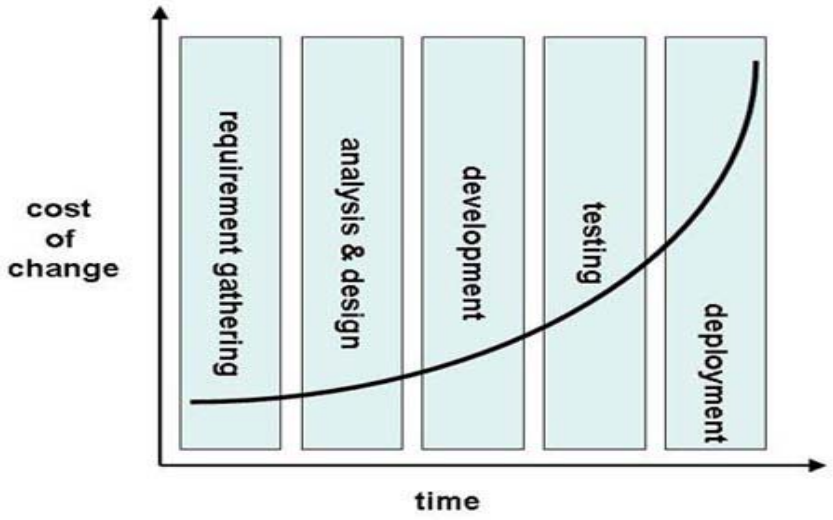
So that part has been a real success for you, I think more so than anything.

***Daphne**:*

I do think that it helps in a lot of ways to make sure we give the client exactly what they are looking for.

Thus, Daphne iteratively used *User Stories* to capture user requirements and was better prepared to respond to change. This experience resonates with and corroborates guidance from the literature on agile methods which suggests that when agile processes accommodate change earlier in the development process, when these changes are less costly: A $4000 increase in price caught early may have thwarted a price tag several times larger if fixes and changes were introduced at a later date. Figure 44 demonstrates the typical relationship between cost, time and change.

Figure 44 Relating Change, Cost and Time in the SDLC



An agile process does not always ensure that the developers will get a *User Story* right the first time; however, agile processes hedge against the costs of undetected change or, at the very least, document a customer's understanding of the problem in their own words. In one case, observed in January 2009, the practitioners appropriately placed the onus of responsibility for stakeholder omission on the client as a result their new agile processes:

> ***Daphne**:*
>
> *It hasn't prevented issues from arising, like HCC, but what it did do in HCC situation is that they accepted that they were responsible for the change.*

***Researcher**:*

So it is a good way to not only elicit what the customer wants, but to make the process open and transparent such that if there is change, then change is okay… the cost of change is a two-way street....

***Daphne**:*

…you are able to keep things above board and honest and transparent, I think.

However, Daphne did not seek to castigate the client; rather, Daphne values the transparency and clarity of her agile process and the surety it brings in client relationships. The benefits of this transparency were also apparent in other projects. IMS was a client whose project was tracked by the practitioner-researcher team throughout its lifecycle in order to examine the effects of adopting XP on the project. In the case of IMS, the practitioners' use of XP also provided clarity and transparency such that change was no surprise to either the client or the practitioners:

***Daphne**:*

Yes, and I think that it is helping with IMS too because I know the project has gone two months longer than they wanted it to, but when we ended up handling the day-to-day problems instead of Jim, it had doubled the size of the project. There was nothing we could do about that. I think that they also fully realize that the working software they accept is going to be 90% ready for them to use in-house as well as for their web site. So they are going to be in really good shape when we are done.

***Researcher**:*

So, they will have a product that really works, that fits them, that is what they really needed.

***Daphne**:*

And we will have delivered it in 4 months what has taken 2 years for somebody else to set up. So I feel like IMS is happy with the communication. The level of communication has kept the client from going completely crazy with the delay.

***Researcher**:*

Right, because they are aware…

***Daphne**:*

They are aware and they have been involved in every step.

In this case, *Frequent Releases of Working Software* improved client communication such that quality and productivity were enhanced. While it was challenging for the practitioners to include their customers as a literal team member as is called for in the XP literature (Beck 1999:68), *User Stories* and *Frequent Releases of Working Software* did keep the team in touch with their clients in a more focused manner.

> ***Daphne**:*
>
> *They argued...no, they argued back and forth over it.*
>
> ***Researcher**:*
>
> *You can't help that.*
>
> ***Daphne**:*
>
> *Somebody didn't think it was a big deal. And I told them, at this point, they saw this, they saw it up before launch...they approved the design...*
>
> ***Researcher**:*
>
> *Okay.*
>
> ***Daphne**:*
>
> *...I said that you tell them that they are going to pay us again.*
>
> ***Researcher**:*
>
> *Showing it to them early gave them the opportunity for benefit; if they didn't realize benefit and it is outside of your control then you have a weak team member in the client.*
>
> ***Daphne**:*
>
> *Yeah. Their Mom decides.*
>
> ***Researcher**:*
>
> *And so going forward, what I would suggest is this: that this methodology, and I haven't ... I don't think I've pressed on the issue hard enough, really insist that the client is on the team - it's non-negotiable.*

6.1.4 Evaluating Issues Related to Skills Development

During diagnosis, Daphne indicated her belief that her team's skills, wisdom and judgment were not equal to that of her own. Daphne's requirement was that a new software development method would address skills development and cross-training. In adopting a new method, Daphne wanted an "SSC way" to emerge in the team's practices which were also modeled after Daphne's own practices. Thus, Daphne desired confidence in her employees' skills such that her employees could assume more responsibility and, accordingly, share in the rewards of increased productivity.

Table 31 indicates that the XP processes which address skills development the most are: *Collective Code Ownership*, *Learn and Communicate*, *Pair Programming* and *Spike Solutions*. Whereas *Learn and Communicate* and *Collective Code Ownership* are outcomes of adopting XP, *Pair Programming* and *Spike Solutions* are activities which assist in skills development, skills cross-training and skills transfer. The SSC developers used each of these activities to good effect for learning and skills development.

The SSC developers made significant strides with *Pair Programming* during the latter period of the Dialogical AR Partnership, once the interventions related to *Reflective Practice* and *Organizational Learning* were considered. The SSC developers started to use *Pair Programming* in December of 2008 and, by February 2009, the developers had become comfortable in using *Pair Programming* for skills development and learning. However, given the very small size of the SSC development team, it was nearly impossible to use *Pair*

Programming to create all production code, as is prescribed by "orthodox" XP practices (Beck 1999; Jeffries et al. 2001; McBreen 2002a)[18].

Pair Programming is not without controversy and some say that *Pair Programming*, and XP in general, might actually hinder productivity for a team of SSC's size; in this sense, McBreen (2002) characterizes a team of SSC's size as a "Tiny Team." Thus, some have suggested that a team which appropriates XP should "disaggregate" the elements of XP and subsequently critically examine the value of each element (Glass 2003; McBreen 2002a). In such an examination, SSC found most value in *Pair Programming* for skills development and knowledge transfer via *Spike Solutions*.

Daphne immediately valued *Pair Programming* for skills development and saw developer pairing as a means for skills transfer from herself to Fred and from Fred to Johnny, and so on. Citing her own higher skill and experience level, Daphne saw *Spike Solutions* and *Pair Programming* as means to elevate her employees' skills until parity wither her own skills was reached. Further, Daphne fully expected the skills of her employees to surpass her own and she saw *Pair Programming* as a means of transferring skills when new problems required her expertise. In Figure 45 approximates Daphne's view on the skills disparity between those of her employees and her own. Over time, the researcher came to regard Daphne's feelings on this matter as an *Espoused Theory*.

[18] It is difficult to speak of XP "orthodoxy" when adaptation and modification are encouraged in XP by some of its original authors. Furthermore, there is considerable controversy surrounding many XP practices where a significant number of scholars and practitioners have come out strongly against XP.

Figure 45 Espoused Skills Disparity at SSC

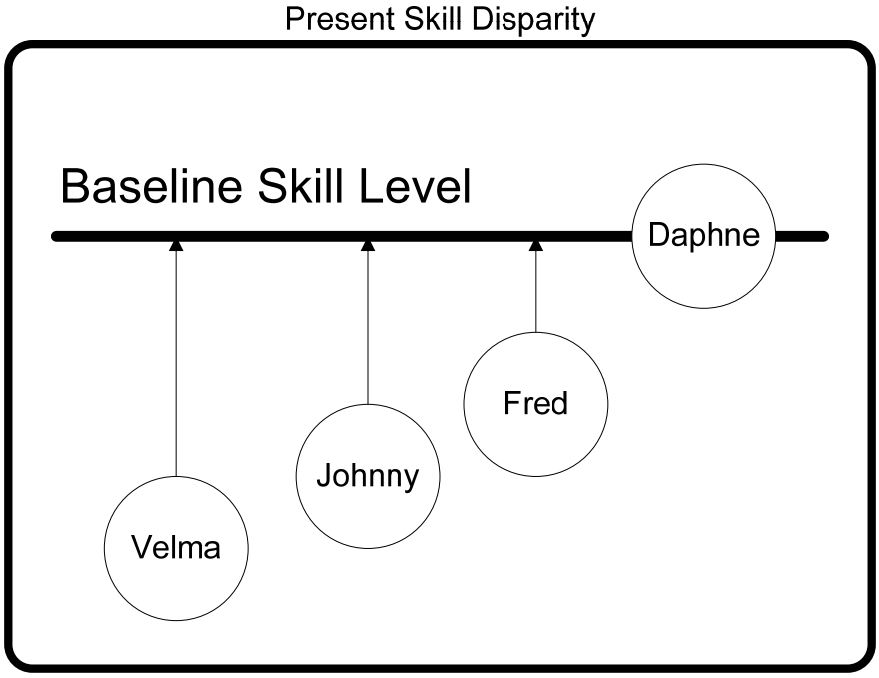


Figure 46 Estimated Future Skills Disparity at SSC

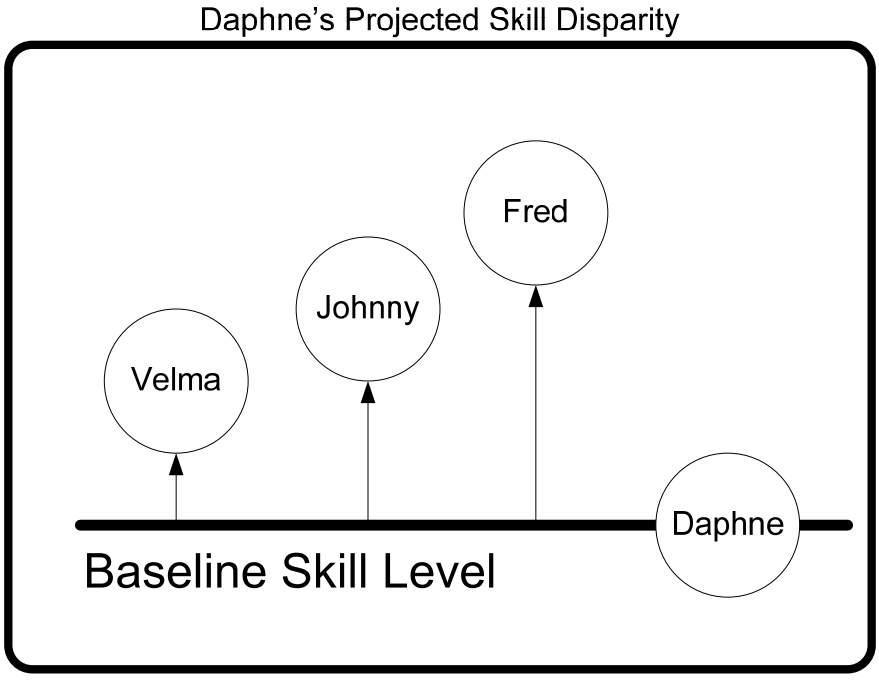


Figure 47 Ideal Skills Distribution for SSC

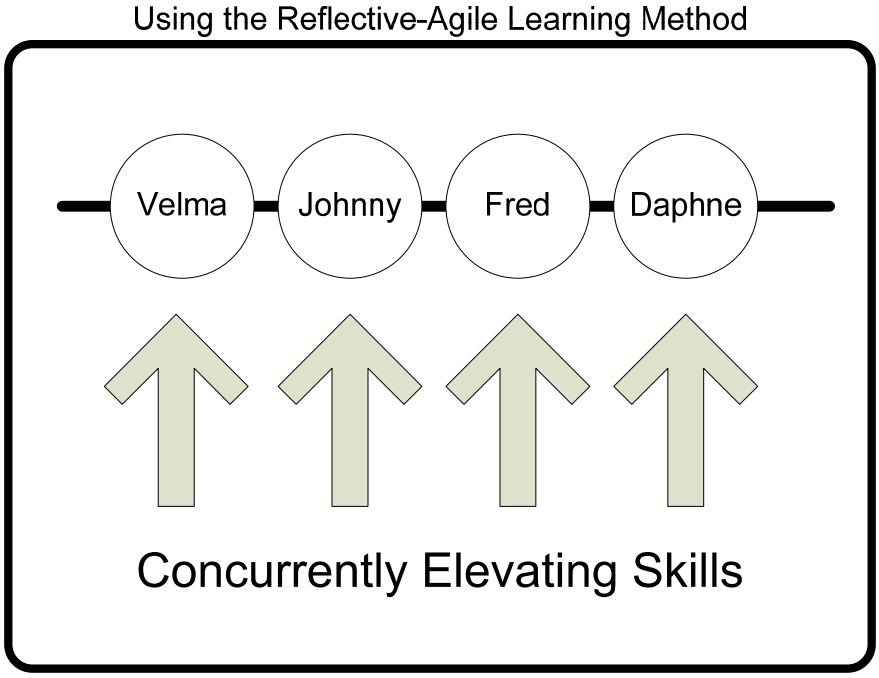


Figure 46 approximates Daphne's estimate of what skills disparity will be subsequent to her developers' use of *Pair Programming* to create *Spike Solutions*. However, the researcher offers the process of learning approximated in Figure 47 as a better approach. Ideally, the methods SSC adopts should enable learning processes which keep individual skills on par with overall team skills. Furthermore, the team's skills should collectively increase as they use XP with a strong learning system also embedded within XP.

During the course of the Dialogical AR Partnership, *Pair Programming* did facilitate learning and skills cross-training; the researcher observed Fred and Johnny pair on several occasions and documented this activity in dialog and field notes. Both Johnny and Fred engaged in a dialogical exchange during pairing which also facilitated other activities such as *Refactoring*:

***Researcher**:*

What "collective co-ownership" represents are some of the activities we have been discussing, such as "Refactor Mercilessly." …don't be afraid to change the structure of things when you get new information.

***Johnny**:*

We did a lot with the control that we are working on.

***Researcher**:*

…when I last watched you pair?

***Johnny**:*

Every other day there is new information which prompted us to change the structure.

***Fred**:*

I think it is working out… we worked on the form capturing component and that was the first time we really sat down together for any extended period of hours

***Johnny**:*

…and we both kind of have ownership over it because we both started it not knowing much about it and are kind of equally invested.

***Fred**:*

Yeah, we were equally confused and then equally rewarded for solving the problem.

Both Johnny and Fred realized, early on, that *Pair Programming* is a critical aspect of *Collective Code Ownership* which describes a process for learning, agility and skills transfer. In their actions and reflections in *Pair Programming*, Fred and Johnny developed shared and mutually-constructed context which provided means for learning, skills transfer and skills cross-training. Later in the same dialog, Johnny and Fred speak in depth on their earliest impressions regarding Pair Programming:

***Fred**: So that's nice… along the way, we taught each other things. I mean, even little things - from keyboard shortcuts to common programming practices.*

***Researcher**:*

Yes, Daphne told me that your feedback regarding Paring was you thought maybe you were goofing around too much.

***Fred**:*

(Laughter) That's not what I believe I said.

***Researcher**:*

Please tell me, what did you mean?

***Fred**:*

No, we were -- this morning we were just kind of talking in passing about just what we were able to get accomplished this week. And I had said -- I was actually saying how I liked our dialogs with you and how much we appreciate what it seems to be doing for us, which was a positive thing....

***Researcher**:*

Oh.

***Fred**:*

I said "a lot of times we sit there and are tinkering with code and we are just sort of talking about one thing or another and then all of a sudden a good chunk of the day has just gone by. Not in a negative way, just that it's like you missed half a day.

***Johnny**:*

It was more fun to pair program, I thought. Like, the day went by faster, it may not be a good thing for business, but it was more fun, I thought, programming with someone else.

***Researcher**:*

Well, that is interesting feedback; let's focus on the context of what you are saying. Correct me if I am wrong, but it seems like "tinkering" was sort of pejoratively used, which means it's kind of a negative connotation. "Tinkering" means you are not cranking out production code but you are fiddling around. Is that what you mean by it?

***Fred**:*

No, just, um, it was more like we were all trying to solve the problem together, which was ... I guess I do the same thing when I am just sitting here by myself, you know, like I'm trying to use different methods, whatever, is that the way to do it? Refresh the screen; is that the answer I am looking for?

Fred and Johnny have described how learning through the shared context of *Pair Programming* didn't feel like work and was a rather rewarding experience. In solving problems together, their skills transfer is facilitated as is their learning. Fred and Johnny continue the dialog and reflect on one pairing episode when the researcher had actively participated:

***Fred**:*

…this was like triple programming... there were three of us there and we were thinking: well, maybe we can try this and Johnny would say "well, why don't you try that?" and you would go, "…you know I've done this before with my other students and…" something like that.

***Researcher**:*

Yes.

***Fred**:*

...and next thing you know… well, it was weird to be at the keyboard and then bombarded with all three ideas at once as opposed to trial and error just me by myself. So I guess the word "tinkering" just came out of the fact that it became more apparent as there was somebody over my shoulder doing it with me as opposed to just sitting there by myself, I'm thinking "I' don't just do my job, I'm going to have to try to figure it out."

Fred realizes that pairing will require some adjustment and that pairing is a departure from past work habits in which he solved problems in isolation. Both Fred and Johnny realized that learning in isolation would not necessarily benefit team learning. The researcher goes on to explain the utility in *Pair Programming* with respect to learning and skills sharing:

***Researcher**:*

So, one of the benefits I see you have realized is that when two people were "figuring it out," you were also making your organizational learning and knowledge that much wider. You know it means when you get together, and you actually have a reflection point for "remember when," "do you remember when we were?" Because you did it together, and especially if it was something you didn't know how to solve and after pairing you now know, then you'll probably remember that time - the time you went from didn't know to know, but you'll both know it and, so, by being able to refer to that you have a basis for new learning.

Since you were both here and then you did it together, you now have a shared context for that learning and then when you go to a new situation, like…

This is Fred, at my right-hand, and this is Johnny, at my left-hand; now both were equally saying "I don't know." Okay, and then both, through the interaction of pair programming, transition to "now I know." So, then some new thing comes along but only Fred addresses it. Fred can say to Johnny, even if you are not working together, "Do you remember that time?" And Johnny will respond, and what he says, because he was in the same context of learning as you were, transforms your understanding of that problem and because he knows how you solved it, you actually moved together, even if you were not

pairing. So that's shared context of learning should benefit you in the future, even in times where you are not necessarily pairing.

***Fred**:*

What is cool is that is exactly what happened, even this morning, because I still was tweaking it, but obviously we run into situations here where there's just a deadline and with Johnny working on one piece and I'm working on another piece - if it is not done separately there's no way we can get it done today.

Even, I guess, the argument is that, in the long run the pair programming gets done quicker but it definitely would not have been the case today. We had ... and, I mean, it is okay to recognize that and act on it...

***Researcher**:*

Okay.

***Fred**:*

... but exactly what happened: I've gotten to a point with the form capture component; Daphne made a suggestion yesterday and then I ran with that angle and it was done - I finished all the cards (containing User Stories) - put all the cards away, when I put the last card in the pack, I had it working, it was .... I showed to Daphne, she said that's what I wanted it to do.

***Researcher**:*

So that was an acceptance test.

***Fred**:*

That was really cool.

Fred realizes that there are limitations to pairing but he also realizes that pairing can be used as a means to approach the unknown and for shared learning. In a later dialog, Fred indicates that the learning process described by the researcher in the previous dialog excerpt was something he had recently experienced:

***Fred**:*

I was thinking about the point you made earlier about, like, we sat down together, like physically, at the same computer for a little while and got to a certain point that we both had collective ownership of what was going on...

***Researcher**:*

Yes.

***Fred**:*

... we both understood it; and, now, we have gone our separate ways, we're working on different things today. This morning I spent about an hour - I don't know if it is called re-factoring? But I reworked the code...

***Researcher**:*

That's refactoring.

***Fred**:*

…essentially it was the same result a different way. Okay, so, I was able to literally turn around in my chair and told Johnny "Daphne mentioned this to me last night as an alternative way of doing it, given how we are actually going to have to implement in our program, it is probably wiser to do it this way, we didn't consider it because we just hadn't pieced it together that way." It's one of those things where, when you are building it, and it is working and it is showing up on the screen, and you realize: "Oh, what I really have to do is plug it in this way for this program, it's not, the fit is not the same."

***Researcher**:*

Yes.

***Fred**:*

But now I was quickly able to say, "Oh, I'm right there with you, I totally get what you need to do..."

***Researcher**:*

Yes, precisely.

***Fred**:*

It was really just, you know, if it was steps like 1-10, we've already done 1-9 and they were cool, the same thing Johnny and I worked on, step 10 just got refactored; the end result was the same, but now we understand, both of us, how it is going to be plugged into these programs we are building over the next few weeks.

It was clear that Fred and Johnny understood the relationship between *Pair Programming* and Learning. Later, when working on a project for a major state university in Central Virginia, Fred and Johnny would continue to pair program to good effect. Daphne was very pleased with how Fred and Johnny progressed while pairing. They moved from a standpoint where pairing was primarily used to create *Spike Solutions* to a point where *Pair Programming* offered a consistent

system for shared learning and knowledge and skills sharing. Initially, the researcher thought that *Pair Programming* would be a "hard sell," however, the SSC developers were able to appropriate *Pair Programming* in a manner that was appropriate for them; thus, they tailored the method to their needs.

Pair Programming was not just a means of shared problem solving, but it was also an occasion for forward thinking for team and individual repertoire development. Johnny and Fred describe, in subsequent dialog, how pairing became a means for thinking-in-action such that their designs evolved and formed "on the fly" as the pair conducted a "*conversation with the materials.* (Schön et al. 1996:176)"

***Johnny**:*

Next to the cards (containing User Stories), the cards were in the design also. So, we would take that as a basis to our design and then expound, so yeah.

***Fred**:*

You mean like when we were sort of coming up with creative ways to solve the… there were points I think we were even going like beyond the requirements of what we were doing. We were just trying to think forward.

***Johnny**:*

Yeah, more so if the cards weren't quite specific enough.

***Fred**:*

As if we were still filling in the blanks... or make believe. (Laughter)

***Researcher**:*

What I observed is that your interactions were clearly mixed and there were things you were thinking of on the fly. Because your interaction causes you to have new thoughts, and I think that you were doing a lot of designing on the fly, you don't know how used to that you are. From my observation, you weren't used to doing it together as much, but how often do you think: "…are you just trying to chase down requirements or are you ..."

***Fred**:*

Oh, I see...

***Researcher**:*

"…seeing things as they unfold and making decisions on the spot that fundamentally changes or progresses the design?" Because that's what I saw you doing...

***Fred**:*

That happens, seems like, a lot.

***Johnny**:*

Especially if there is not a bunch of forethought. Like, with this, we didn't know what we were doing, so periodically we would have to change our plan.

***Fred**:*

But it's not like it's all brand new and has never been done it before - a lot of it is based on what we've practiced.

***Researcher**:*

But your interactions seemed to have been part of the design. Meaning, it wasn't just what Fred thinks or what Johnny thinks, but the interplay between those two things was now an active role in creating the design and, therefore, you owned that design together.

***Fred**:*

Yeah, yeah.

***Johnny**:*

Yeah, there's a few times when we were going back and forth whether to typecast the form as a text box or as a button… every single time he'd (Fred) mentioned that I'm like "Ugh, I don't want to do that" and so would you (to Fred), but if we would have had to do the whole thing, or if Fred would have just done that without thinking about it and thinking about how…

I don't think I would have been totally behind it - even if at the end, if that was the only way it could have been done - if I wasn't in the loop and I would have just saw how it was done and…

***Researcher**:*

You wouldn't have been invested.

***Johnny**:*

…yeah, I wouldn't have liked it as much.

During and as a result of their "conversation with the materials," Fred and Johnny were beginning to understand the meaning of *Collective Code Ownership* in XP. In this sense Fred

and Johnny now mutually "owned" their code by way of their shared effort in designing "on the fly" and through the shared context they creates as they debated and "haggled" over design.

The researcher's field notes at the time are also full of quick notes indicating the ease with which both Fred and Johnny were thinking aloud on a problem as they implemented and adopted a reflective practice for themselves. This was not because Fred and Johnny had to impute their reflective actions onto XP – the method is reflective by design. When two sets of eyes were cast on a problematic area of code, developer rapport between Fed and Johnny increased as did their propensity to pose good and challenging questions to each other. The spontaneity of instantaneous designing while the pair was thinking and knowing-in-action was also a byproduct of XP's tenet that each developer thinks at different levels of action. Thus, in a manner similar to Schön's (1987) *Ladder of Reflection* exercise, Fred and Johnny would, in turns, assume strategic and tactical roles. During the researcher's observations, Johnny did well at "big picture" strategic thinking focusing on the problem, while Fred did well with "tactical" thinking focused on the code. Johnny would think ahead: "Should this form handle email as HTML or text-based?" At the same time, Fred would reach for a reusable library or tweak a configuration file in response. Frankly, it was very satisfying to watch.

Field notes from this period also mention how each programmer started to learn the tacit knowledge and assumptions of the other programmer in the pair. Despite sitting in close proximity to each other in the same office for 40+ hours a week, one gets the sense that Fred and Johnny may have learned several new things about each other during their pairing experience. It should be mentioned that while Daphne espoused an enthusiasm for the opportunity Pair Programming held for skills and knowledge transfer, Daphne was not observed to have engage in Pair Programming during the course of the Dialogical AR Partnership. Daphne had described

teaching sessions she thought were *Pair Programming*, but she did not subject herself to the full benefits of *Pair Programming* and the *Ladder of Reflection* as Fred and Johnny had. The researcher's field notes point out that Daphne's espoused support was at odds with her actual use of Pair Programming: Daphne wanted to impute her own knowledge and skills on her employees but did not avail herself of the new tools presented to the team for this very purpose.

6.1.5 Evaluating Issues Related to Learning

Table 31 indicates that learning was addressed by the following XP activities: *Collective Code Ownership, Daily Standup Meeting, Learn and Communicate, Pair Programming*, and *Planning* (Release and Iteration) *and Feedback Loops*. This section will evaluate which of these had the greatest impact as they relate to learning.

As the researcher reviewed the dialog transcripts, it was apparent that learning was key to many, if not most, of the practitioners' espoused problems as well as problems identified in the researcher's interpretive analysis. Diagnosis revealed that while SSC expressed value in learning, they did not have time to make learning a daily habit. Thus, the researcher determined that SSC needed an embedded learning system which would be a habitual and integral part of XP. As the strictures of any given method will not likely "fit" in all circumstances, the practitioners needed learning habits that transcended the particulars of XP. McBreen (2002) and Glass (2003) have both said that methods are not (and should not be) "one-size-fits-all." As "…software is not a mechanical process" where you can simply drop it in, and as "…you will never be able to adopt a process without doing some adapting to fit…" (Glass 2003:119). It was apparent that the practitioners required a learning process which considers their shared social and

historical context. Inherent in Daphne's espoused desire for a new method is her belief that a method will corroborate and confirm her own instincts about quality. However, if Daphne continues to believe that quality and learning are a function of transferring what is "inside her head" to her employees, then Daphne would fail to realize the utility in a learning system that distributes learning, skills and knowledge.

Again, the XP elements *Collective Code Ownership* and *Learn and Communicate* are outcomes of adopting other "concrete" agile practices. In observation, it was the *Daily Standup Meeting* and *Pair Programming* which most effectively enabled team learning. The feedback loops inherent in *User Stories*, *Planning Game*, *Release Planning*, *Iteration Planning*, *Frequent Releases of Working Software* and *Acceptance Tests* were also important inputs to learning. However whether and how the practitioners choose to respond and react to these feedback loops will determine whether they, as a team, actually learn from feedback. As the researcher transcribed and reviewed transcripts of the dialogs and field notes, it was evident that issues related to learning and communication would be of the greatest importance to SSC as a good learning process would sustain SSC's progress regardless which software development method they used.

In the last months of the Dialogical AR Partnership, the researcher's interventions increasingly focused on learning and reflection. The researcher felt as though the team's software development method or methods should have, at their core, a sustainable learning process. Thus, the researcher envisioned SSC as a learning organization and shared this vision and direction with the practitioners. Nearer to the end of the Dialogical AR Partnership, the evidence from the dialogs indicated that the practitioners were beginning to understand what it meant to become a

learning organization. In a dialog in February 2009, Daphne spoke on individual and team reflection and team learning:

***Daphne**:*

I think that the blogging, the Wiki, and the daily standup are very important ways to keep an active project on tract....

... we have so many projects going on that if we do not follow, either through the daily standup or through… now we are trying to use this Wiki, so that the projects will not get off track. Fred and I were talking about the QSI project on the way over here. I emailed Fred and Velma early in the week and said that their instructions were on the Wiki and they were due today...

***Fred**:*

Yep.

***Daphne**:*

.... but the Wiki hasn't really been followed. (Laughter)

***Fred**:*

I admit I hadn't even gone to the Wiki yet on that project.

***Daphne**:*

So, obviously, that is something we will talk about and try to work on, but I do think that if we are going to commit to reflection on-action, we are going to have to visit the Wiki and the blogs...

.... now, along those lines, one thing that we are actually not doing is keeping personal blogs. We have no private blogs. We have the team blog, but nothing private, which I think as far as looking back and reflecting on our own actions in relationship to one another, where we have a private blog that nobody else can see, over time, we might start to see a trend in the way we react to one another.

We might avoid future problems based on past experiences if we were blogging about that stuff, and that would be the one that we could say "you know, I probably shouldn't have said that."

***Researcher**:*

Well, some call it journaling but in 2009 we can use blogs, but they also call it journaling -- pre-blog, pre-browser...

***Fred**:*

Right.

***Daphne**:*

So I think as far as our personal reflection goes, it is not being done probably as much as we could. I don't know if that would assist in our relationships to one another just because we might reflect on something we did in the past, look back on it and be like "why did I react to that that way?" Because that's what we learned as we were trying out the Ladder of Reflection ...you know?

***Researcher**:*

Yes.

By this point, in February 2009, SSC had tried the last of the researcher's interventions and, through their action planning; the Dialogical AR team had agreed to try blogging and a team wiki as a means for engaging the *Reflection-in-action* and *Reflection-on-action* components of Schön's (1983) *Reflective Practice*. Additionally, the team had been using the *Ladder of Reflection* to this end (Schön 1987). Daphne encouraged her employees to use these new tools for reflection and learning in the next portion of the same dialog:

***Daphne**:*

I wrote down beside that.... "Getting to the reflection on the reflection of the description of designing"- why we did what we did when we did it.

***Researcher**:*

Yes.

***Daphne**:*

... that has global ramifications for the entire company. In fact, if we stand back and say.... not just ... well, you used an example of Fred doing something, explaining it to Johnny and Johnny is trying to understand Fred's explanation and description: "Well, why did Johnny think I meant that?"

***Researcher**:*

Right.

***Daphne**:*

...Well, that internal process of Fred saying: "Why did Johnny think I meant that? It means that the next time he and Johnny approach a project together he'll have a better understanding of where Johnny is coming from.

***Researcher**:*

Yes, that's right.

***Daphne**:*

... and I think, overall, certainly, as a team that is all pulling for the same goal, the ability to understand where one another is coming from is invaluable.

I think it would cut down on a lot of difficulty, so, that's my reflection on reflective practice. I will say that as we have done a lot of extreme problem solving... I don't know if we can call it programming. (Laughter)

***Fred**:*

It is double extreme.

This dialog, and the empirical support it lends to the researcher's own evaluation and interpretation, is very significant to the research questions in this dissertation and constitutes, in the researcher's eyes, a full "green light" emphasizing the importance of a learning system for SSC as they adopt XP or, perhaps, some other method. At this point, the Reflective-Agile Learning Model and Method (RALMM) certainly seemed justified and validated, in terms of perceived usefulness, by the team. Daphne moves on, in the same dialog, with an example of how elements of RALMM were becoming necessary to their operations:

***Daphne**:*

I set up the other day for [a new client]... I kind of jumped on it and Johnny really jumped on the bandwagon -- he saw me doing some stuff in SQL Server and he's like "you need to put that in a Wiki -- we need to all know how to do that."

And he's right, we do all need to do that because it is very simple... what it was it was causing him a problem and he thought he couldn't restore the database.

He thought that he couldn't do it, he thought there was something wrong with the database… he thought that it couldn't be done. He thought that there was something wrong with the backup file, but it was really he just hadn't put in the correct parameters and he didn't know where to look.

***Researcher**:*

Yes. Well, the Wiki is supposed to collect your organizational knowledge and learning, so... it requires forming habits...

That is a word I didn't stress in this presentation but I have in two others in the past: these things are "habits," and so, you know, habits aren't always easy to learn or break: They are patterns and that's why I said you should consider your own social and historical context. You are not going to magically snap your fingers and your new habits overnight. It is a gradual process. And you have to... like any time you want to assume good habits, see the value in it.

***Daphne**:*

I think if we take that extra step of trying to figure out not only why somebody saw it, or the fact that somebody saw it differently, but why they saw it differently... I think will actually... the coherence would be better.

Organizational Learning was not the only type of learning of importance to emerge in the diagnoses, nor was it the only issue addressed. The other learning which interested SSC concerned acquiring new skills and knowledge and transferring skills and knowledge throughout the team. As previously discussed, this type of learning is addressed by the *Pair Programming* and *Daily Standup Meeting* elements of XP. However, the preceding discussion on habits is also appropriate to the goal of learning and acquiring new skills and knowledge. Thus the practitioners at SSC should to engage both XP and RALMM for team learning.

The practitioners also demonstrated early understanding of *Spike Solutions* and immediately saw value in this activity as a means for learning. Remarkably, Daphne very early on asked developers to blog learning from these spikes; it was apparent to Daphne that this organizational/team knowledge must be preserved for the future. The researcher immediately recognized Daphne's directive to her employees to blog *Spike Solutions* as being related to and perhaps a form of *Reflection-in-action* and I thus was encouraged to further explore *Reflective Practice* with the team. This also demonstrated the practitioners' willingness to adopt the daily

habits required to become reflective practitioners. This is what Schön called a "Reflective Practicum." The researcher's field notes at the time reflected, excitedly: "this is real progress!"

6.1.6 Signs of Success

So far, this chapter has discussed evaluations on the outcomes and consequences of action in the Dialogical AR Partnership. Many of the descriptions along the way - those characterizing diagnosis, action planning, action taking and evaluation – have used somewhat tentative language regarding the practitioners' adoption and adaption of XP and their use of the theory-based interventions. This section is concluded by highlighting material from the final and concluding dialog in February 2009. These excerpts from that dialog are notable as they describe SSC's overall satisfaction in having entered into the Dialogical AR Partnership and having undertaken processes of reflection, self-discovery and growth. Before this chapter consolidates its evaluation on issues and actions relating to learning and communication, these final reports from the practitioners give a sense of their thinking at the conclusion of the study. Daphne on the topic of change, fear and growing her team:

> ***Daphne**:*
>
> *In a later slide you talk about not fearing, not being afraid, being confident of change. I will say that I am 100% more confident that we can incorporate a new person into this team and fit them right in and make them productive almost from the start. That is not something... I think that was something I mentioned early on... that I felt would be difficult about hiring on somebody, having to start over from the beginning… I no longer feel that is the case. I believe that the type of interaction that we've set up between our developers and me, the projects, I believe that people will gain a lot of exposure really fast.*
>
> ***Researcher**:*
>
> *Good, good. And you would attribute that to some of the changes you've adopted during the course of our work together?*

***Daphne**:*

It didn't exist until we started this work.

Daphne's desire for a method for her developers was at times correlated to Daphne's desire for quality, a jealous protection of her reputation and to legitimize her team's process. On this matter, Daphne shared the following:

***Daphne**:*

About when I came to you and said: "…I wish I had a method."

Well, I wrote down the word "legitimize," which was the word that came into my mind and that was what prompted me to make that statement.

Basically, I felt like we were producing quality work; I was producing quality work, but when I am the small company going to a big company trying to say I can do just as good a job, how do I legitimize myself? And I remember just having to convince Stephen at LMV that I would deliver what I promised.

Now we've done this thing and I've had a couple of people come to me, "okay, well what is the process that you would use?"

And I could just blurt it out because we actually have a process and it has a very legitimizing effect on our business to say that we have a process. And people are like "wow, that sounds really impressive." So I think that we have established what it was it was I hoped to be able to say:

"Yeah, we've got this way of working and we can deliver what you are looking for."

We were able to tell GAB and TLS that we could deliver their project on time because we knew we had a process in place. We were able to tell them "you'll see it; you'll see it in our process; you'll be able to give us feedback on it as it goes up," and it really turned that project around.

***Fred**:*

It worked!

***Daphne**:*

It worked. It did.

In this section, Daphne expresses satisfaction with the face validity of her new method; she is happy to have a documented and sound process which substantiates the quality or her product in

the eyes of her clients. However, much further than face validity, both Daphne and Fred credit their new methods, and their learning system, as a significant success factor in handling of a crisis with a strategic partner and third-party clients in January of 2009. Whereas the researcher initially believed that the practitioners had lightly considered some of the interventions as a result of the "distraction" of this crisis, it appears that SSC's new practices, and their new learning system attached to their methods, had really paid off for them.

6.2 Focus on Learning and Communication

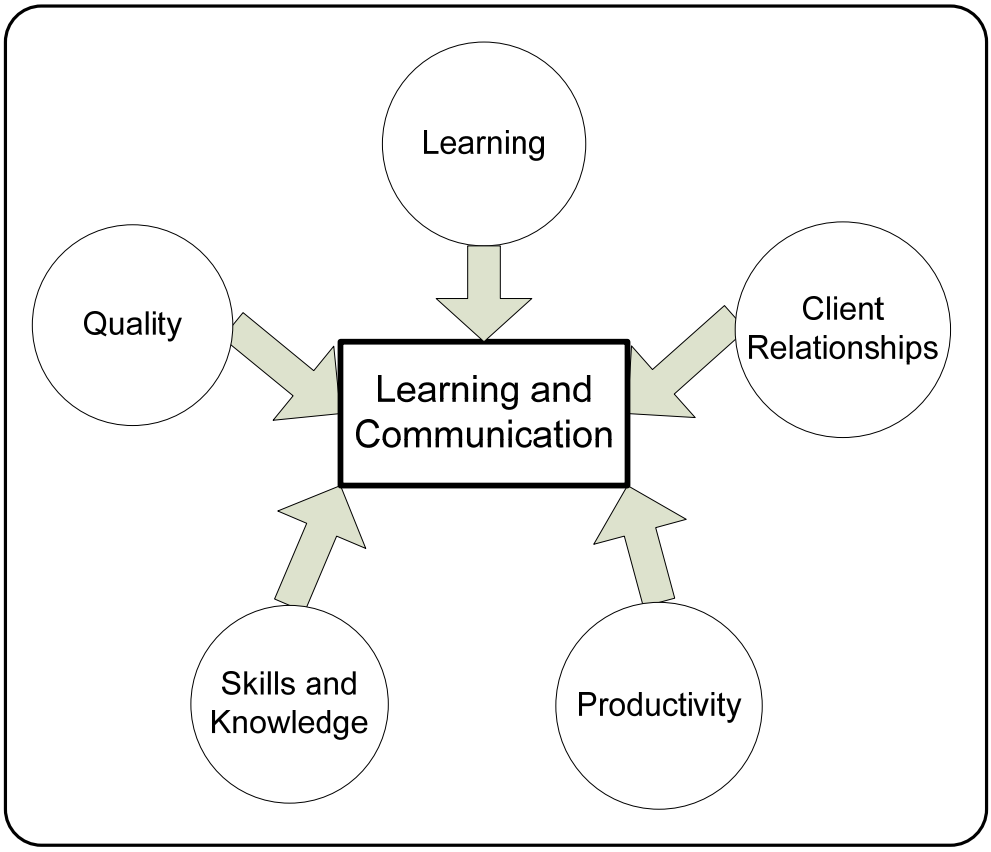
This section proposes that the majority of the SSC practitioners' issues, and actions taken to address these issues, are reducible to concerns related to learning and communication. The impact of the theory-based interventions during the Dialogical AR Partnership constitutes the basis for this proposition. Thus, the outcomes and consequences of action planning and action taking are summarized and evaluated, on the whole, as being functions of learning and communication. The theory-driven interventions related to communication and learning body of theory were drawn from the following theoretical perspectives (Table 37):

Table 37 Bodies of Theory for Learning and Communication

Scholar	Body of Theoretical Knowledge
Don Schön	• *Reflective Practice*: o *Problem setting* o *Reflection-in-action* o *Reflection-on-action* o *Naming and Framing*
Chris Argyris and Don Schön	• *Learning Organization* • *Single-Loop Learning* and *Double-Loop Learning* • *Model I and Model II behavior*

Following from the previous section, this section will contrast and summarize the outcomes of actions and interventions to: illustrate the applicability of the above bodies of theory; outline the principle components of the Reflective-Agile Learning Model and Method (RALMM), and provide a basis for reducing SSC's issues as being functions of learning and communication (Figure 48).

Figure 48 Simplifying Issues and Actions to those of Learning and Communication



6.3 Philosophical Grounding for Interpretation

Dialogical AR, as a research endeavor undertaken in the scientific attitude, must do more than solve the practitioners' problems; Dialogical AR must evaluate theory-driven interventions

and actions in order to specify learning. In order to specify learning, the researcher must review the evidence from the Dialogical AR Partnership and, in adopting the scientific attitude, become the instrument of scientific investigation. Thus, the added value in the researcher's interpretations is found in the researcher's interpretations and second-order constructs and in the relationships and interplay among these second-order constructs. This section discusses how the phenomenology of Schutz guides the interpretations used to specify learning in this study.

6.3.1 A Phenomenological Approach to Interpretation

Baskerville (1999) situates action research as a research approach concerned with "…complex social systems…" which "…cannot be reduced for meaningful study" (p.3). Thus, as this research adopts a Dialogical AR research method, there are perspectives on research in general which the researcher is also adopts (Baskerville 1999:4-5):

- **The Interpretivist Viewpoint**: As the researcher becomes an integral part of actions taken in the Dialogical AR Partnership, the researcher is constantly incorporating interpretations of action taken in the natural attitude. Thus, "meaning" in the actions and consequences of action is "constructed" in the Dialogical AR Partnership by combining theoria and praxis. (Baskerville 1999:4; Mårtensson et al. 2004:508)

- **The Idiographic Viewpoint**: The Dialogical AR Partnership, by design, focuses on the historical and social context unique to the practitioner setting. Thus, given the representative organizational sample of one, there is an abductive quality to generalizations which arise from an action research study. Any statistical notions of generalization would be inappropriately applied to action research

- **A Qualitative Approach to Evidence Analysis**: As the "meaningfulness of actions," expressed as text or text-objects, constitutes the evidence in a Dialogical AR study, a qualitative analytic approach is warranted

This researcher unequivocally accepts each of Baskerville's (1999) imperatives in adopting a qualitative and Interpretivist approach to evaluating and interpreting the dialogical evidence from

the study. As matters pertaining to epistemology are significant in this research, the importance of making clear which philosophical assumptions are assumed in a qualitative and interpretive research effort cannot be understated. Thus, as "…the most pertinent philosophical assumptions are those that relate to the underlying epistemology which guides the research" (Myers et al. 2002:5), this section revisits the influence of the phenomenology of Alfred Schutz in this research. Myers (1997) has said that phenomenology and hermeneutics serve as the philosophical basis of the interpretive approach to qualitative research. Walsham (2002) corroborates Myers' position in identifying phenomenology as among the philosophical bases for the philosophy of *Interpretivism*. Thus, considering Walsham's classifications on the epistemological stances on knowledge and reality available to a qualitative researcher, the use of Dialogical AR in the case this study, assumes an epistemology which Walsham calls *Normativism*[19] and an ontology of *Internal Realism*. (Walsham 2002:104) This is so as Dialogical AR is a form of social inquiry which is guided by the "scientific attitude" whilst engaging in the "natural attitude" of action.

Burrell and Morgan (1979) would classify Dialogical AR as existing within the *Interpretive* paradigm, where a researcher "…sees the social world as an emergent social process which is created by the individuals concerned" (Burrell et al. 1979:28). As meaning in the Dialogical AR Partnership is cooperatively and iteratively constructed, Dialogical AR is consistent with Burrell and Morgan.

With respect to philosophical grounds, Mårtensson and Lee (2004) emphatically state Dialogical AR's *Social Constructionist* position and phenomenological grounding (Mårtensson

[19] Alternatively referenced as *Nominalism* by Burrell and Morgan (1979)

et al. 2004:514). Mårtensson and Lee also cite and credit both Berger and Luckmann (1966) and Schutz (1962, 1967) for providing the theoretical lens of *the social construction of reality* from which the philosophical position of Dialogical AR is drawn. This section continues with an examination of social constructivist and phenomenological position and considers the suitability of this philosophical grounding for this research in light of the evidence from the Dialogical AR Partnership.

6.3.2 Constructing a Subjective Understanding of Meaning

Burrell and Morgan (1979) succinctly summarize many of the broader positions held within Interpretive and Phenomenological views of Schutz (1962, 1967) and Berger and Luckmann (1966). Burrell and Morgan (1979) suggest that the interpretivist philosopher or scientist:

> *…sees the social world as an emergent social process which is created by the individuals concerned. Social reality, insofar as it is recognized to have any existence outside the consciousness of any single individual, is regarded as being little more than a network of assumptions and intersubjectively shared meanings.* (p. 28)

Thus, in taking the Interpretive and Phenomenological perspective on meaning, the Dialogical AR researcher is "…more orientated towards obtaining an understanding of the subjectively created world 'as it is' in terms of an ongoing process" (Burrell et al. 1979:31). In this sense, a researcher utilizing Dialogical AR is "…concerned with understanding the essence of the everyday world" (p.31) and developing actions based on this understanding. While using the Interpretivist approach offers significant guidance for Dialogical AR, Burrell and Morgan's characterization gives the impression that Interpretivist researchers are passive observers who are not concerned with change but rather with understanding the world as given. However, Berger

and Luckmann (1966) would hold that the subjective reality for the researcher to understand is a volatile and changeable phenomenon that is individually and socially constructed. Dialogical AR goes further by instigating change via an iterative development of the subjective understanding in the Dialogical AR Partnership.

While "meaning" is mutually constructed in the Dialogical AR Partnership, the specification of subjective meaning is not the practitioners' job – certainly not in the case of Dialogical AR's imperative to specify learning for a body of scholarly knowledge. To this end, the researcher must possess a methodical and theory-driven basis for action, evaluation and learning where these activities are motivated by socially-constructed and intersubjective meaning in the Dialogical AR Partnership. The body of knowledge from the work Schutz, and Berger and Luckmann, presents a convenient duality of focus on both the individual and the social construction of reality and meaning. Berger and Luckmann (1967) describe the processes of socialization as a two-step construction of subjective meanings: (1) Primary Socialization and (2) Secondary Socialization. Schutz characterizes a process of constructed understanding, from the objective to the subjective, as *Verstehen* where the social scientist develops second-order constructs (subjective understandings) from his or her observations on the first-order constructs held by individuals in their common-sense understanding taken in the *natural attitude of everyday life*. Thus, in order to evaluate the outcomes and consequences of action in the Dialogical AR Partnership, the researcher must document and interpret the mutually constructed meanings developed within the partnership so as to create meanings of the *second degree*.

6.4 The Evidence and Theoretical Reflection on Agile Methods

This dissertation is not the first effort to reflect on a philosophical and theoretical basis for the success of agile methods. An increasing number of scholars in information systems development, software engineering, and related disciplines, recognize the paradigmatic shift which agile methods represent (Abrahamsson et al. 2003; Hazzan et al. 2004a; McBreen 2002a; McBreen 2002b; Nerur et al. 2007; Rajlich 2006). Of these scholars, Nerur and Balijepally (2007) and Hazzan and Tomayko (2004) are particularly aware of the connection between agile methods and the work of Argyris and Schön (1974, 1978, 1996) and Schön (1983, 1987). In the case of this dissertation, that the theory-driven interventions primarily are drawn from the work of Argyris and Schön is an *a priori* assumption grounded in the literature. While this assumption was made clear in Chapter 4 of this dissertation, the matter is now revisited in light of the evidence from the Dialogical AR Partnership.

Theory-driven interventions in the Dialogical AR setting were inspired by the researcher's familiarity with the work of Argyris and Schön and also by Tomayko and Hazzan's (2004) work on the human aspects of software engineering. Tomayko and Hazzan (2004) suggest how Schön's *Ladder of Reflection* can be used as a device to enhance reflection in *Pair Programming*. Similarly, Tomayko and Hazzan (2004) emphasize the importance of the learning organization in reference to the work of Argyris and Schön (1974, 1978) and Senge (1994). Thus, several of the interventions introduced into Dialogical AR Partnership are directly informed by Tomayko and Hazzan (2004).

Similarly, the researcher's perspectives on creativity, art and craft in design are partially inspired by McBreen (2002b), Schön (1983, 1987) and Schön and Rein (1994). Nerur and Balijepally (2007) have also established a significant and direct precedent for the theoretical arguments made in this dissertation which align agile methods with the Schön's epistemology of

Reflective Practice. While Hazzan (2002, 2003 and 2005) and Tomayko and Hazzan (2004) have also linked *Reflective Practice* to agile success, Nerur and Balijepally (2007) very clearly acknowledge the paradigmatic shift "…from a mechanistic perspective to a perspective that acknowledges the existence of environmental uncertainty and complexity…" (p.80) apparent in agile methods and liken this shift to contemporary thinking in strategic management. Nerur and Balijepally also see *Reflective Practice* directly in the design strategies inherent in agile methods:

> *Designers use their own personal expanding knowledge to continually reframe the problem and devise an appropriate solution. In the spirit of Donald Schön's reflection-in-action, the process involves repeated modifications of the practitioner's design through insights and knowledge gained from active interaction with the problem situation. The improvisations and learning resulting from this dynamic interplay ("conversations," as Schön called it) between the designer and the problem lead to a succession of updated problem representations.(Nerur et al. 2007:80)*

Nerur and Balijepally suggest that the discipline, art and science of design are all well-addressed in agile methods in a manner which departs from the dominant paradigm of process, engineering and optimization (2007:80). Similarly, Nerur and Balijepally recognize that design constitutes a "generative" metaphor for agile methods which "…incorporates learning and acknowledges the connectedness of knowing and doing (thought and action), the interwoven nature of means and ends, and the need to reconcile multiple worldviews" (2007:81).

In a comprehensive and substantial manner, and drawing from design and strategic management, Nerur and Balijepally outline the emergent metaphor of design which is manifest within agile methods and somewhat resonant with the emerging emphasis on design in Information Systems research (Blevis et al. 2006; Järvinen 2007; Lee 2007; March et al. 1995; Orlikowski et al. 2002). The researcher aligns with Nerur and Balijepally's (2007) thinking and draws upon their conclusions regarding the theoretical influences on agile methods and the new

and emergent metaphor of design. In this regard, the designed artifact and contributions of this study remain mindful of and indebted to Nerur and Balijepally's (2007) guidance.

Whereas Nerur and Balijepally (2007) locate *Reflective Practice* within a metaphor of design, the evidence from the Dialogical AR Partnership suggest that *Reflective Practice* contributes to *Organizational Learning* in a manner sympathetic to but also distinct from an emphasis on design. A learning system, as manifested within the Reflective-Agile Learning Model and Method, focuses on team interpersonal dynamics beyond what is mentioned in Nerur and Balijepally (2007) and more aligned with Tomayko and Hazzan (2004). Therefore, the evidence from the Dialogical AR Partnership contributes to both antecedents from the literature and also situates the outcomes of this research between them. Table 38 relates the traditional view of design in contrast to Nerur and Balijepally's (2007) emergent metaphor of design.

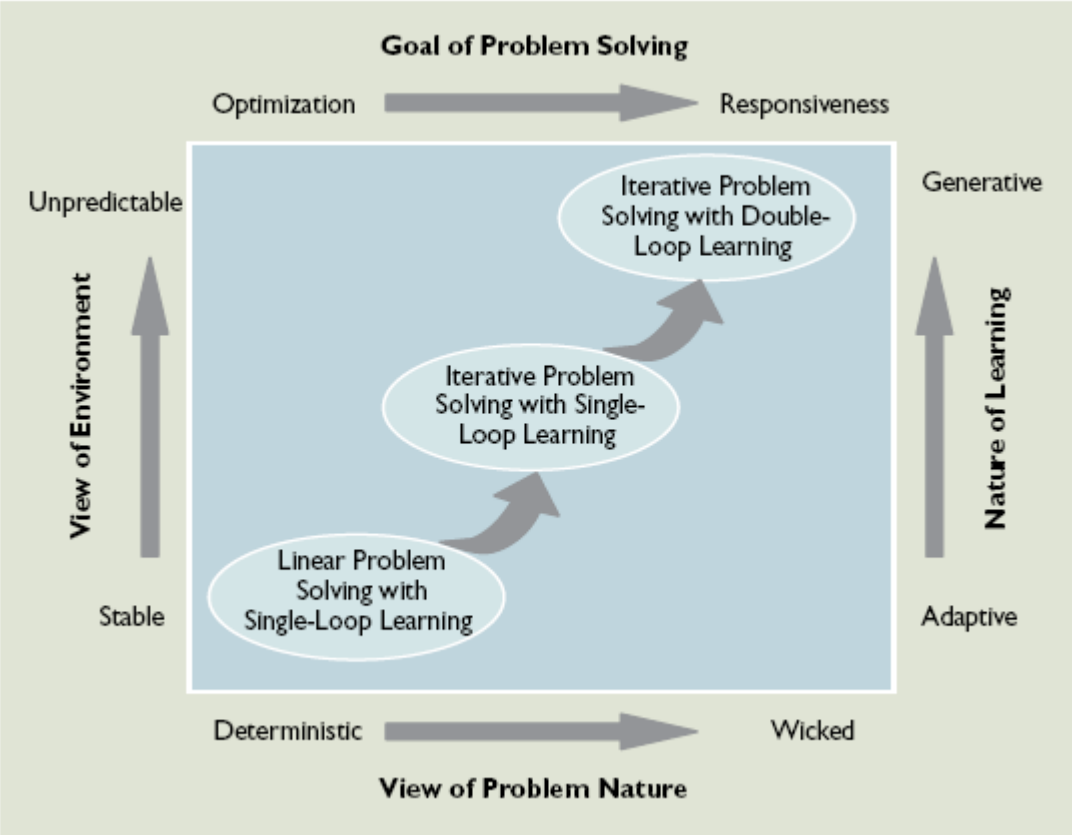
Table 38 Traditional and Emerging Perspectives of Design (Nerur et al. 2007)

Aspect	Traditional View of Design	Emergent Metaphor of Design
Design process	Deliberate and formal, linear sequence of steps, separate formulation and implementation, rule-driven	Emergent, iterative and exploratory, knowing and action inseparable, beyond formal rules
Goal	Optimization	Adaptation, flexibility, responsiveness
Problem-solving approach	Selection of best means to accomplish a given end through well-planned, formalized activities	Learning through experimentation and introspection, constantly reframing the problem and its solution
View of the environment	Stable, predictable	Turbulent, difficult to predict
Type of learning	Single-loop/adaptive	Double-loop/generative
Key characteristics	-Control and direction. -Avoids conflict. -Formalizes innovation. -Manager is controller. -Design precedes implementation.	-Collaboration and communication – integrates weltanschauungs, or worldviews -Embraces conflict and dialectics

		-Encourages exploration and creativity and is opportunistic -Manager is facilitator -Design and implementation are inseparable and evolve iteratively
Rationality	Technical/functional	Substantial
Theoretical of philosophical roots	*Logical Positivism*, scientific method	Action learning theory, Dewey's pragmatism, phenomenology

In Table 38 Nerur and Balijepally (2007) imply that the emergent metaphor of design addresses new situations in which traditional views on design have been deemed inadequate or not sufficiently explanatory in the face of new experiences and evidence from practice. During the Dialogical AR Partnership at SSC, this notion is substantiated in the experiences of the SSC developers who recognized that the problems they faced and the solutions developed were "…not what we were taught in school." Thus, in the case of custom web development at SSC, many of the traditional views on design did not fit with SSC's experience whereas XP has fit. Nerur and Balijepally (2007) outline their characterization of the shifts in attitude towards design wrought by the rise and success of agile methods in Figure 49; the influence of Argyris and Schön's work is very apparent.

Figure 49 Evolutionary Shifts in Design Thinking (Nerur et al. 2007)



An additional and substantial body of antecedent work is that of Mathiassen and Purao (1998) and Mathiassen (2002) in what they describe as "Reflective Systems Development." This work also brings together many of the theoretical perspectives which have motivated the interventions introduced into the Dialogical AR Partnership in this research. In work that somewhat predates the agile methods movement, Mathiassen and Purao (1998) outline a model for Reflective Systems Development which incorporates *Organizational Learning* from Argyris and Schön (1978, 1996), *Reflective Practice* from Schön (1983, 1987) and also suggests action research as a means for researching and developing systems. Mathiassen and Purao (1998) outline an approach to systems development which summarizes 20 years of their own research and practice. While Mathiassen and Purao (1998) outline what constitutes a genuine paradigm

on “Reflective Systems Practice,” they do not mention agile methods in their 1998 work or in their subsequent work in 2002. Thus, while this research also owes Mathiassen and Purao recognition for their very important antecedent and seminal work on the role of *Reflective Practice* in systems development, the omission of reference to agile methods presents an opportunity for this research to also “bridge” to theirs. As a sketch of Mathiassen and Purao’s (1998) position on *Reflective Practice* in systems and software development, they offer the following eight lessons in Table 39.

Table 39 Key lessons in Reflective Systems Development (Mathiassen 1998)

Reflective Systems Development	
Lesson 1	*Reflection-in-action* provides a useful understanding of systems development practice.
Lesson 2	Methods are primarily frameworks for learning.
Lesson 3	Improving practice requires an experimental attitude and an appreciation of the involved communities-of-practice.
Lesson 4	Sustainable project management traditions are fundamental in successful improvement strategies.
Lesson 5	Action Research brings relevance to the research process while supplementary approaches improve the validity and reliability of the research results.
Lesson 6	Dialectics is a useful framework for understanding practice based on different reference disciplines.
Lesson 7	Combining interpretations with interventions and normative propositions increases the relevance of the research.
Lesson 8	Research and teaching are relevant practices for learning about systems development.

Several of Mathiassen and Purao's lessons are directly applicable to this research and have served to reinforce the interventions introduced into the Dialogical AR Partnership at SSC. Lessons 1, 2, 3, 5, 7 and 8 have direct bearing and influence on this dissertation and the ideas introduced at SSC.

This section has discussed some of the critical antecedent and/or seminal work in information systems development and software engineering which directly address the theoretical interventions introduced during the Dialogical AR Partnership at SSC. Thus, it was only through action and reflection in the Dialogical AR cycles that the importance of these

precedents was fully evident. As the summary interpretations and recommendations culminating in the designed artifact are discussed in the remaining sections of this chapter, it is imperative that the importance of the work of Hazzan (2002, 2003 and 2005), Tomayko and Hazzan (2004), Nerur and Balijepally (2007), and Mathiassen and Purao (1998, 2002) is recognized and acknowledged. Furthermore, the specification of learning for the body of knowledge on *Reflective Practice* should and will address these important antecedents. The balance of this section will review key interpretations of the outcomes and consequences of action taken during the Dialogical AR Partnership at SSC.

6.4.1 Learning and Communication: Support for Reflective Practice

This section develops a synopsis of interpretations of the consequences of actions taken in the Dialogical AR Partnership at SSC. These actions were in response to interventions pertaining to *Reflective Practice* for individual learning and communication. Mårtensson's research at the Omega Corporation outlined by Mårtensson and Lee (2004) is used as a guide for these synopses.

Several elements of XP were explicitly presented as supportive and representative of Schön's *Reflective Practice*: *Daily Standup Meeting*, *The Planning Game*, *Pair Programming* and *Customer as Team Member*. The principle elements of *Reflective Practice* introduced to the SSC practitioners were: *Reflection-in-action*, *Reflection-on-action* and the *Ladder of Reflection*. The most relevant antecedent literature driving interventions related to *Reflective Practice* were those of Tomayko and Hazzan (2004) and Mathiassen and Purao (1998).

The following tables highlight, in summary, the Dialogical AR processes which informed the actions and interventions taken to address issues related to quality, productivity and skills and knowledge sharing, acquisition and transfer. These issues are ultimately considered as functions of learning and communication; learning and communication are inherent outcomes of using XP and provide redress for the balance of SSC's concerns. In most circumstances, the researcher's interpretations (towards second-order constructs) are presented as theories-in-use and examples of *Model I* behavior in some cases. As a result, the theory-driven interventions were attempts to encourage *Model II* behavior. While the terms *Model I* and *Model II* behavior were never used directly in the Dialogical AR Partnership, the concepts of *Single-Loop Learning* and *Double-Loop Learning* were presented to the practitioners several times during the Dialogical AR Partnership as a "hint" concerning the type of learning system that SSC should adopt.

Table 40 Considering Reflective Practice in Dialogical AR at SSC: Quality

Considering Reflective Practice in Dialogical AR at SSC: Quality	
***Issues from Diagnosis*:** Daphne is concerned about a range of issues related to maintaining the quality of her product.	
The Practitioners' Perspective	**The Researcher's Perspective**
***Espoused Theory*:** Daphne wants to adopt a software development method which will reveal and reinforce the quality inherent in her product ***Theory-in-use*:** Daphne does not need a method to know her product is the best, but she wants a method which minimizes the doubt in the mind of her current and potential clients.	***Interpretations towards Second-Order Construct Development:*** Daphne has sacrificed a lot of time, work and energy to develop a quality product and, as a result, is not quick to trust others with her reputation. Daphne believes that any method will mirror her already sound qualities and is likely to ignore aspects of a new method which contradict her present beliefs. Daphne is likely to consider new habits if their utility is absolutely proven and if they do not threaten her control over quality in her organization. ***Intervention to Introduce:*** XP is a method which is designed to deliver quality while also focusing on agility and learning. XP is well-suited to small teams and provides a focus on learning and reflection which will help SSC in the long-term
Dialog and Action	
In reflective dialog in the Dialogical AR Partnership the researcher discusses the possibility of adopting XP. The researcher stresses that a reflective and learning-focused method designed for small-team effectiveness would be a good approach. The partnership considers the historical context such that SSC has evolved from Daphne's hard work and the effect this has on her team. The researcher presents the principle elements of XP and focuses on those aspects which are relevant to reflection and quality: Daily Standup Meeting, User Stories, Release Planning, Planning Game, Iteration Planning and Acceptance testing. The researcher conducts exploratory demonstration, lecture and discussion sessions such that the SSC developers can explore and adopt XP. Daphne sees an immediate increase in developer productivity and expresses a desire to further engage XP. The researcher anticipates the opportunities for *Reflective Practice* and arranges to incorporate this approach in a subsequent iteration.	

Table 41 Considering Reflective Practice in Dialogical AR at SSC: Productivity

Considering Reflective Practice in Dialogical AR at SSC: Productivity	
***Issue from Diagnosis*:** Daphne is concerned that too much time is wasted in revisiting old problems and desires a software development method which focuses on efficiency and reuse.	
The Practitioners' Perspective	**The Researcher's Perspective**
***Espoused Theory*:** Daphne wants a method which will increase team productivity by incorporating past lessons and retaining team experience and knowledge. With these improvements she can increase billable hours. ***Theory-in-use*:** Daphne's wisdom, judgment, experience, knowledge, work ethic and commitment are greater than those of her employees. If she could transmit more of her own qualities to her employees and mould her employees to her working style, then productivity will increase as she will have "cloned" herself as much as possible.	***Interpretations towards Second-Order Construct Development:*** Daphne feels strongly about her own qualities and knows the most efficient way to do things. Daphne believes that the example of her habits, alone, is sufficient guidance for her employees and will increase productivity. Daphne desires a method which will not contradict her own views but will codify them for her employees. Daphne, alone, holds the key to productivity and just needs to get her developers to understand "her way." ***Intervention to Introduce:*** XP and *Reflective Practice* will create a team which, while thinking-in-action, will utilize reflection to explicate tacit knowledge and create an environment of open and honest communication which increases understanding and learning. Processes for learning and communication will open understanding and, in effect, increase productivity within that understanding as SSC adds to their "repertoire"
Dialog and Action	
XP holds that Pair Programming offers the greatest possibility of realizing raw improvements in productivity as the reflection and awareness among the developers engaged in Pair Programming is a synergistic multiplier. Daphne readily accepts and believes in the concept in synergy but points out the difficulty in creating all production code using XP due to the nature of their work and their team size. The researcher points out that others in the literature and in practice have encountered this and that it is okay to adapt the methods. Soon Daphne and the SSC developers realize that Pair Programming is useful for the creation of Spike Solutions which increase productivity though learning and R&D. The team finds as much productivity gains from User Stories and the Daily Standup Meeting as they do in Pair Programming. The researcher realizes that the sum of these activities can be presented as functions of *Reflection-in-action* and *Reflection-on-action* and begins to characterize the interventions in a manner which suggests and leads to *Reflective Practice*.	

Table 42 Considering Reflective Practice in Dialogical AR at SSC: Skills Development and Transfer

Considering Reflective Practice in Dialogical AR at SSC: Skills Development and Transfer	
***Issue from Diagnosis*:** Daphne wants the team to share their unique skills with each other and wants a method which ensures skills acquisition, transfer and retention.	
The Practitioners' Perspective	**The Researcher's Perspective**
***Espoused Theory*:** Daphne wants a method which will handle and manage skill in the team. She indicates that she has a lot to teach her employees and that she can learn from them as well. Further, she believes her employees have skills between them that are complementary. ***Theory-in-use*:** Daphne wants to "dump" her own abilities into her employees so that she can focus on growing the business. Daphne believes her employees must be incentivized to improve as they won't care about the business to the same extent that she does as they are not as vested as she is.	***Interpretations towards Second-Order Construct Development*:** Daphne maintains a critical attitude concerning the ability for her employees to reach her level of commitment and skill. Daphne believes that vested employees would step out and develop themselves. Daphne favors employees who accept her own skills and approach without question. Daphne believes those with their own style are wasting time as she has to continually show them "her style." ***Intervention to Introduce*:** XP calls for *Collective Code Ownership* and to *Move People Around* in their roles. This approach is bolstered by *Reflective Practice* as it enhances Pair Programming, which is an integral component of *Collective Code Ownership*.
Dialog and Action	
As SSC had already realized productivity gains and quality assurance in XP. The researcher discussed the possibilities for skills and knowledge sharing inherent in the development activities specified by XP. As SSC was not able to Pair Program in the manner prescribed by XP, their adoption of Collective Code Ownership was somewhat curtailed. Since SSC was Pairing for Spike Solutions, this represented their best chance to use the *Ladder of Reflection* to explicate tacit beliefs and habits in action. Thus, this exercise in *Reflection-in-action* lead to the use of blogs and wikis as tools for retaining *Reflection-in-action* in order to facilitate *Reflection-on-action* during the *Daily Standup Meeting*.	

6.4.2 Learning and Communication: Support for the Learning Organization

This section summarizes interpretations of the consequences of actions taken in the Dialogical AR Partnership at SSC; these actions are in response to interventions primarily related to *Organizational Learning*. Mårtensson's research at the Omega Corporation outlined by Mårtensson and Lee (2004) is used as a guide for these synopses.

The elements of XP explicitly presented as supportive and representative of Argyris and Schön's (1974, 1978, and 1996) Theory of action for *Organizational Learning*: *Daily Standup Meeting* and *Pair Programming*. The principle elements of Theory of Action introduced to the SSC practitioners were: *Single-Loop* and *Double-Loop Learning* and *Model I* and *Model II* behavior. The literature informing interventions related to *Organizational Learning* were those of Tomayko and Hazzan (2004) and Mathiassen and Purao (1998) and Nerur and Balijepally (2007).

The following tables highlight, in summary, the Dialogical AR processes which lead to the actions and interventions addressing issues related to learning and customer relations. These issues are ultimately considered as functions of learning and communication. In most circumstances, the researcher's interpretations and second-order constructs identified *Model I* behavior in the practitioners and also, in most cases, the theory-driven interventions were attempts to encourage *Model II* behavior. While the terms *Model I* and *Model II* behavior were never used directly in the Dialogical AR Partnership, the concepts of *Single-Loop Learning* and

Double-Loop Learning were presented to the practitioners several times during the Dialogical AR Partnership as a "hint" concerning the type of learning system that SSC should adopt.

Table 43 Considering Organizational Learning in Dialogical AR at SSC: Individual Learning

Considering Organizational Learning in Dialogical AR at SSC: Individual Learning	
***Issue from Diagnosis*:** Daphne wants a method which allows for team learning and development.	
The Practitioners' Perspective	**The Researcher's Perspective**
***Espoused Theory*:** Daphne believes that a new software development method should afford the team time for R&D and learning. Daphne also wants to capture team "know-how" to avoid "re-inventing the wheel." ***Theory-in-use*:** New skills are vital to maintain a quality product; however, this is no time for learning as there is just too much work. Daphne needs something that will systematically enhance learning retention as she feels she must revisit instruction to her employees at multiple times.	***Interpretations towards Second-Order Construct Development:*** With a focus on growing the business and ensuring a quality product by instilling her own values into her employees, Daphne leaves little time and energy to devote to learning. Daphne believes that systematic learning should transfer her own knowledge and experience to her employees. As SSC solves problems, they are not systematically retaining the learning which occurs during the problem-solving process. ***Intervention to Introduce:*** XP has several elements which focus on learning. Furthermore, *Reflective Practice* provides guidance on how thinking and *Reflection-in-action* can extend SSC's professional repertoire.
Dialog and Action	
In reflective dialog in the Dialogical AR Partnership the researcher emphasizes the portions of XP which directly address learning: Code Standards, Pair Programming, User Stories and the Daily Standup Meetings. The researcher stresses that a reflective and learning-focused method designed for small-team effectiveness would be a good approach. The partnership considers the historical context and social context whereby attitudes towards learning mostly reflect Daphne's own. Thus, while learning and staying on top of developments in the profession is very important to Daphne, growth in her business has restricted the free time available to learn new ideas, techniques and tools. The researcher suggests that XP has an embedded learning process which will help SSC. The researcher also presents the ideas of *Reflective Practice* and *Organizational Learning* as manifested in *Reflection-in-action*, *Reflection-on-action*, the *Ladder of Reflection* and *Single-Loop* and *Double-Loop Learning*.	

Table 44 Considering Organizational Learning in Dialogical AR at SSC: Team Learning

Considering Organizational Learning in Dialogical AR at SSC: Team Learning	
***Issue from Diagnosis*:** Daphne sees the benefits for learning inherent in XP but important lessons from recent crises are not systematically retained. Although Fred and Johnny have learned during Pair Programming to create Spike Solutions, Daphne is concerned that this learning is not retained for common use across the team.	
The Practitioners' Perspective	**The Researcher's Perspective**
***Espoused Theory*:** **Daphne believes that team learning needs to be preserved and distributed so that past lessons can be applied to present and future problems.** ***Theory-in-use*:** Employees are still "re-inventing the wheel" and taking too much time to solve problems that I have solved in the past. Since I don't know what Fred and Johnny are learning together, I don't know if it matches my own knowledge and patterns. When I show my employees how to do something, I want it to stick.	***Interpretations towards Second-Order Construct Development*:** Daphne is concerned that lessons the team learns are not being applied to present and future problems. Daphne wants learning to filter through her such that it is processed against her own framework of knowledge. Daphne will "get behind" team learning if she is aware of it and if it matches her own judgment. Daphne is convinced by success and will get behind any learning system that produces results. ***Intervention to Introduce*:** While XP provides good learning processes, a learning system based on *Reflective Practice* and *Organizational Learning* will establish feedback loops which utilize reflection to establish a team consciousness with respect to learning.
Dialog and Action	
The researcher first introduces the *Ladder of Reflection* during Pair Programming in order to develop patterns of reflection which explicate tacit knowledge and situates this knowledge in action. The researcher then asks SSC to consider two modern technologies, blogs and wikis, to enable *Reflection-in-action* and *Reflection-on-action* in a daily cycle. SSC developers "journal" into a personal blog and then contribute to a team blog at the end of the day. The next morning, prior to the Daily Standup, the team reflects-on-action from yesterday's team blog entries. The team also "feeds" a team wiki with on-the-spot *Reflection-in-action* and with "peer	

reviewed" *Reflection-on-action* after each Daily Standup Meeting. In this manner, the team grounds their learning in their shared social and historical context.

Table 45 Considering Organizational Learning in Dialogical AR at SSC: Strategic Partnerships

Considering Organizational Learning in Dialogical AR at SSC: Strategic Partnerships	
***Issue from Diagnosis*:** The team is in a negative and restrictive strategic partnership which repeats a series of mistakes across many projects. Over time, the short-comings of this relationship are negatively impacting SSC's reputation and hindering SSC's adoption of their new software development methodology	
The Practitioners' Perspective	**The Researcher's Perspective**
***Espoused Theory*:** I want our strategic partner to understand that we should be managing more of the project and quoting the estimates as we understand the technology and what we can deliver. Our strategic partners don't understand the "…intangible of IT." ***Theory-in-use*:** We are much more on-the-ball than our strategic partner and should have a larger role in project management and client interaction. Our strategic partner doesn't listen to us and learn from their mistakes – they are ruining our reputation by making us look bad due to their short-comings.	***Interpretations towards Second-Order Construct Development*:** In their relationship with their strategic partner, SSC is demonstrating a classic case of *Single-Loop Learning* whereupon SSC does not question their own fundamental assumptions and governing variables. As project failure due to misunderstanding, miscommunication and perhaps intransigence in their relationship to the strategic partner increases, SSC becomes more impatient and lays blame squarely at their partner's feet. SSC's *Model I* behavior is preventing any real solutions to emerge. The problem persists because SSC is aware of what is right but continues to repeat the same actions leading to failure ***Intervention to Introduce*:** SSC's learning system should include *Reflective Practice* as a means to explicitly identify and modify *Single-Loop Learning* and *Model I* behavior in favor of *Double-Loop Learning* and *Model II* behavior through *Reflective Practice*.
Dialog and Action	
SSC's use of the Reflective-Agile Learning Model and Method (RALMM) requires awareness of *Single-Loop Learning* and *Double-Loop Learning* and this topic is discussed openly and overtly during the practitioner-researcher dialogs. The researcher makes it clear the role their use of Collective Code Ownership, Pair Programming, the Daily Standup Meeting, blogging and the team wiki plays in facilitating and encouraging *Reflective Practice* and *Organizational Learning*. The researcher carefully reminds SSC that they must seek awareness through *Reflective Practice* in order to make the best use of RALMM.	

6.5 Towards a Reflective-Agile Epistemology and Paradigm for Learning

The preceding tables have summarized the key issues addressed during the Dialogical AR Partnership. As a process of interpretation, the researcher proposes that each of these issues can be addressed by the use of an agile method, such as XP, in conjunction with a learning system which specifically incorporates Schön's epistemology of *Reflective Practice* and Argyris and Schön's Theory of Action and *Organizational Learning*.

Thus, this dissertation makes the case that the development of a learning system is central to all of the issues which arose during the course of the Dialogical AR Partnership. Furthermore, the need for a learning system which works in conjunction with XP is demonstrated in the dialog and within the researcher's interpretation and second-order constructs. A system for reflective learning should sustain SSC beyond the particulars of any given method by utilizing reflective practice to incorporate learning and technique into SSC's professional and team repertoire.

SSC's learning system should facilitate *Reflection-in-action* and *Reflection-on-action* and agility to match the pace and style of their software development methodology. Thus, a *Reflective-Agile* system for learning suggests a mutually supporting and symbiotic relationship between a model and method for team productivity and quality and a model and method for team learning and development. The designed artifact proffered in this dissertation provides a combination of these systems grounded in theory. This approach is consistent with the views other scholars who have investigated and studied the philosophical grounding of agile methods

(Nerur et al. 2007; Tomayko et al. 2004) and grounding software and systems development in general in the epistemology of *Reflective Practice* (Mathiassen 1998; Mathiassen et al. 2002).

This dissertation's call for and development of a learning system which is both reflective and agile is also consistent with the theoretical positions from which the Reflective-Agile Learning Model and Method borrows and builds upon. As this dissertation proposes that a system of learning supersedes the particulars of any given method or technique, so too does Schön (1983, 1987). Among the principle intents of adopting an epistemology of *Reflective Practice* is to develop and build a repertoire from which an individual, or team, can draw from; the practitioners at SSC should learn and grow from their:

> *…intuitive understandings of the phenomena before them and construct new problems and models derived, not from application of research-based theories, but from their repertoires of familiar examples and themes. Through seeing as and doing as, they make and test new models of the situation… their on-the-spot experiments… also function as transforming moves and exploratory probes. (Schön 1983:166)*

The learning system that SSC has developed and tested within the Dialogical AR Partnership was seeded by theory but developed and honed in practice. This learning system should provide SSC the possibility to build and document individual and team repertoire which is situated in the historical and social context of the team. The learning system will facilitate and document new learning and extant learning for reflection. Should a new developer enter the team, he or she will encounter a living document of the team's repertoire manifested in the team's learning system.

Of course, the living development and documentation of the team's repertoire will not be of much use if this repertoire does not incorporate individual and *Organizational Learning*. If the team's learning system does not listen to the situation's *back-talk* by engaging in *Reflection-on-action*, then the conduits which feed the learning system will clog and/or dry up and cease to function. It is for this reason that individual *Reflective Practice* is so important for SSC's team

learning system. Were the team to share learning and repertoire bereft of reflective practice, the team would likely perpetuate extant models of understanding and reasoning even when the *back-talk* from the situation contradicts or challenges prevailing wisdom. The dialogical evidence and the researcher's interpretations verily highlight the degree to which SSC often ignored the *back-talk* in a situation by not sufficiently changing their own action strategies. As SSC gradually adopted XP and *Reflective Practice,* SSC's learning system incrementally developed such that reflection on *back-talk* precipitated changes in attitude and orientation towards chronic problems.

All is not "solved" with respect to many of SSC's problems and their use of their learning system is nascent and somewhat naïve. However, the learning system itself, if used regularly and diligently, should provide opportunities to reflect and to engage in *Double-Loop Learning* which will also focus SSC's understanding of their learning system and understanding of their team repertoire. Each of the practitioners at SSC are capable of using this system and do believe in the benefits which this system can bestow: Daphne has only the best interests of her customers, company and employees in her actions, yet she and her team can benefit from the reflection and focus of a learning system.

The Reflective-Agile Learning Model and Method presented in this dissertation places considerable focus on individual learning despite an intended primary goal of increasing team learning. This is so as *Organizational Learning* is somewhat nebulous in contrast to individual learning. Argyris and Schön recognize this paradox in their assessment and development of *Organizational Learning*:

> *The meaning of 'Organizational Learning' hinges… on the crucial issue of the levels of aggregation at which organizational phenomena are described and explained and at which prescriptions for organizational actions are directed… one cannot account for the*

> *observed higher-level phenomena of Organizational Learning… without referring to individual and inter-personal processes of inquiry. (Argyris et al. 1996:244)*

SSC's learning system might not be challenged by complex levels of aggregation, given the team's size, but their learning system does focus on the individual processes of learning.

Evidence from the Dialogical AR Partnership should reveal that both researcher and practitioners acted in focused inquiry and in an earnest desire to find the "answers" which would improve SSC's development and learning processes. In this partnership, both the practitioner and researcher "…share an interest in building explanatory models of organizational worlds" (Argyris et al. 1996:37). Argyris and Schön (1996) suggest that the most effective *Organizational Learning* systems can be realized through partnerships where the researcher can observe, encourage and document the team's natural processes of *Reflective Transfer*:

> *"The practitioners' causal inquiry does not yield general covering laws. Their situation-specific inferences of design, efficient, or pattern causality can be generalized only by a process we call 'reflective transfer' – 'transfer,' because the model is carried over from one organizational situation to another through a kind of seeing-as; 'reflective,' because the inquirer should attend critically to analogies and disanalogies between the familiar situation and the new one." (Argyris et al. 1996:43)*

Thus, Argyris and Schön suggest that the team develops and adds to their repertoire by a process of reflective transfer through experimentation, metaphor, analogy and problem setting. Schön (1983, 1987, 1994) has emphasized that *Reflective Practice*, as a matter of problem setting, is about *naming and framing* new situations against extant repertoire. Argyris and Schön (1996) also suggest that scholarly understanding and rigorous theory should develop by way of action research as the Dialogical AR Partnership draws the researcher into practice such that he or she can witness, first-hand, the means by which practice modifies and extends theory. According to Argyris and Schön (1996), this is not a process of using formal experimental and quasi-experimental designs intended to remove the researcher's influence and bias, but rather co-opt

the researcher into learning in practice (p.41). Thus, while the natural-science model holds that a researcher's causal inferences and explanations should result in generalizability, an action-science model holds that the practitioners' situation-specific models of causality, continually modified in an ongoing processes of "on-the-spot" experimentation and *Reflective Transfer*, afford the researcher insight into the real-time evolution of theory-in-practice.

Mårtensson and Lee (2004) distinguish Dialogical AR from Argyris and Schön's *Action Science*" and from Canonical AR in bringing Schutz' (1966) concepts of the *scientific attitude* and the *natural attitude of everyday life* into focus. The researcher elevates his or her participation in the Dialogical AR Partnership by "seeding" interventions which influence the practitioners' processes of causal experimentation with theory. It is through the *natural attitude* that the researcher harnesses the evidence gathered from the Dialogical AR Partnership as focused and scientific learning for a body of knowledge in one or more relevant fields of practice and scholarship. The Reflective-Agile Learning Model and Method, grounded in Schön's epistemology of *Reflective Practice*, is a principle outcome of this research and it serves as the basis for scientific and practical learning.

The remaining sections of this chapter discuss the principles and approaches used to validate an interpretive analysis of the evidence. As Argyris and Schön have identified the beneficial role that action research has for practical outcomes in *Organizational Learning* research, this section also discuss the implications of using Dialogical AR in this study and in the analysis of the dialogical evidence.

6.6 Validating the Interpretive Mode of Inquiry in Dialogical Action Research

In most research, the researcher is expected to validate his or her findings, conclusions and contributions as a means of rigorously assessing the quality of the research processes and outcomes. Some scholars believe that this activity is inappropriate for interpretive and qualitative work as the findings of this work are often subjective in nature and related to the idiosyncrasies and particulars of the research setting (Wolcott 2009). Despite such assertions, there is ample advice for reflecting on the quality of research outcomes in qualitative and interpretive research in the Information Systems literature (Klein et al. 1999; Lee 1991; Myers 1997; Myers et al. 2002; Walsham 2002). Additionally, the quality of action research is also addressed in the Information Systems literature (Avison et al. 2001; Avison et al. 1999; Baskerville 1997; Baskerville et al. 1998; Baskerville et al. 1996; Davison et al. 2004).

This section considers principles for interpretive field studies (Klein and Myers 1999) and principles for Canonical AR (Davison et al. 2004) for guidance to reflect on and assess the process and product of this research. While some scholars might frown on "checklist" approaches, principle-based articles are helpful to junior scholars in orienting and grounding their work. Thus, a set of principles for interpretive case studies and a set of principles for Canonical AR are used to compare and contrast the interpretations in this study in light of the outcomes and consequences of actions taken during the Dialogical AR Partnership at SSC.

This section will continue with the motivations for selecting a Dialogical AR approach in this study; this is followed by a discussion on the potential shortcomings of the Dialogical AR approach with respect to developing interpretations of the meanings of actions taken during the

Dialogical AR Partnership; next, Klein and Myers' (1999) principles for interpretive field studies are examined; and lastly, Davison et al.'s (2004) principles for Canonical AR are considered.

6.6.1 Framing the Dialogical Action Research Approach

Action research holds potential to address the ongoing controversy regarding rigor and relevance by providing a method of inquiry grounded in the realm of practice such that "…research informs practice and practice informs research synergistically" (Avison et al. 1999:94). While many scholars of Information Systems have presented action research as a research method capable of producing rigorous and relevant research outcomes (Baskerville 1999; Davison et al. 2004; Lau 1999; Mårtensson et al. 2004) , it is imperative that qualitative and interpretive inquiry also demonstrate non-trivial and rigorous conduct in research. This section begins by addressing this issue and also discusses other potential issues pertaining to the rigor and relevance of action research.

6.6.2 Considering the Potential Pitfalls of Dialogical Action Research

A danger in using action research lies not only in the fact that it is a qualitative research approach, but that it is as a qualitative approach which draws dangerously near to practice (Avison et al. 1999). Thus, a potential pitfall when using action research is a failure to clearly distinguish action research and consulting (Baskerville 1999; Mårtensson et al. 2004). A key aspect of Dialogical AR, which distinguishes Dialogical AR from consultancy, is the requirement to specify learning in the research outcomes. Specifically, the research must, through the application of theory in the Dialogical AR setting, specify learning for a body of

knowledge that the researcher's community of scholarship would recognize and accept (Davison et al. 2004; Mårtensson et al. 2004). Thus, the interpretation and analysis of evidence from the Dialogical AR setting should reveal efforts by the researcher to retain the rigor of methodical and scientific inquiry.

Despite advice and guidance to the contrary, it is difficult, when charged with the specification of learning, to avoid tutoring, educating and teaching the practitioner members of the Dialogical AR Team. It was often the case that an intervention called for a new technique which called upon the researcher to "teach" aspects of that intervention to the practitioners. Thus, while Mårtensson and Lee (2004) suggest that the researcher approach the partnership as an equal and to do his or her best to "…enter the world of the natives and to have dialogues situated in how they themselves saw their own world…" (p.517), it was, during the course of this research, often necessary to instruct.

Additionally, whereas the recommended advice from Mårtensson and Lee (2004) is to avoid directly exposing the practitioner to raw theory, the researcher did, by way of his role in the dialog, let raw theory "slip out." These utterances were always in response to input from the practitioners and often immediately re-iterated in metaphorical and symbolic terms salient to the practitioners' circumstances and given in the *natural attitude*.

While this research is not an ethnography, many aspects of the Dialogical AR process revealed "confessional" tones, from both the researcher and the practitioners, typically encountered in confessional ethnographies (Schultze 2000). During Dialogical AR process, the researcher adopted what Schutz calls the (1967) *natural attitude* during his visits where instruction, dialog and researcher/practitioners team-building took place. However, the *scientific*

attitude was frequently and necessarily adopted upon reflection once outside of the practitioners' environment and during the transcription and coding activities. Thus, during the course of the Dialogical AR Partnership, the researcher found it necessary to adopt a Janusian perspective in the face of the dichotomous reality of studying social phenomena in a participatory and interpretive mode of inquiry. The Janusian challenge is best described as "…the capacity to conceive and utilize two or more contradictory concepts, ideas or images simultaneously" (Blasko et al. 1986:43; Rothenberg 1971). Thus, while both the *scientific attitude* and the *natural attitude* are presented as discrete states of being during the course of this inquiry, the researcher's actual experience suggests that this distinction is clearer *post facto*. Schutz aptly describes the conundrum whereupon the Dialogical AR researcher must balance the natural attitude and the scientific attitude:

> *Having no 'here' within the social world the social scientist does not organize this world in layers around himself as the center. He can never enter as a consociate in an interaction pattern with one of the actors on the social scene without abandoning, at least temporarily, his scientific attitude. (Schutz 1962:40)*

Thus, in approaching the research problems in an interpretive mode, and in analysis of and reflection on qualitative data, the researcher is pressed to organize clear lines between research and action. This phenomenon was also most likely a by-product of using Lee's DSAR framework where the researcher must necessarily adopt the attitude of practice if he is to participate in designing. Lee (2007) scopes this problem out of the DSAR framework by suggesting that the practitioner would "…not accompany the practitioner back to his organization and, therefore, does not participate in taking the planned action…" (p.52) In theory, this clean and clear distinction makes sense as Lee (2007) uses Dialogical AR as his exemplary AR component of his framework. In this sense, the researcher found it necessary to adapt Lee's DSAR framework to suit the constraints of the research setting.

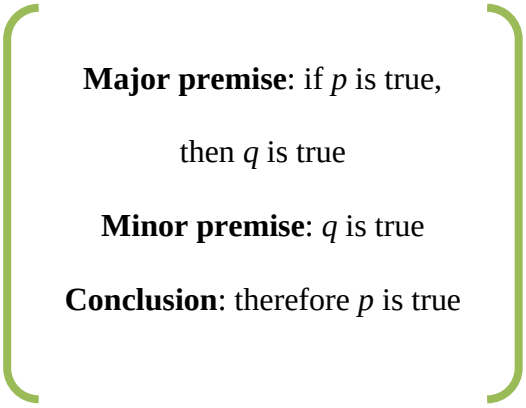
Lastly, Mårtensson and Lee (2004) stress the importance of conducting reflective dialog with the practitioners outside of their normal everyday setting. Ostensibly, this preference is specified in the description of the research method in order to encourage free and open reflection without the distractions of the normal working environment. While this was achieved a few times during the Dialogical AR Partnership at SSC, a small team operating in a small shop is constrained by the realities of deadlines, time and scarce resources. Often, Daphne could not justify being away from the office, despite indicating an early willingness to meet after hours or in a setting away from the office. This perceived scarcity and actual scarcity of time would become a recurring theme throughout my dialog with Daphne as the burden of running a small business was often the centerpiece of our dialog. Thus, this study may have progressed differently if it were possible to more frequently sequester the practitioners away from their daily setting. It was on more than one occasion that one or more of the practitioners were called away during our dialogues. The best that could be achieved, in terms of isolation, was the use of a common meeting room that occupants of SSC's premises (a small two-story office building) shared. The location of this meeting space was not ideal, but, given the circumstances, it had to (and did) suffice.

The researcher also faced issues and challenges related to induction, deduction and abduction. Despite the utility in generalizing to theory, the diagnosis-to-evaluation steps in the Dialogical AR cycle seemed prone to abductive logical conclusions despite bringing theory, *a priori*, to action planning. Lee and Hubona (2008) frame this issue against the formal logical constructs of *Modus Tollens* and *Modus Ponens* in syllogistic reasoning. The deductive logic of the syllogism makes for an appropriate scientific approach to research and validating the conclusions of research regardless of the epistemological stance taken by the researcher – hence

we have what Popper has referred to as Hypothetico-Deductive Logic as the basis for sound science (Lee 1989).

Of course, there are cases where valid research is not deductive in nature. Grounded theorists would arrive at their theories by way of induction where observations precede theory. Dialogical AR and action research in general, are somewhat prone to abduction which can also be described as the logical fallacy of ***affirming the consequent***. Were the proscribed goals of action research limited to theory building alone, abduction presents a weakness with respect to the rigor of any theory developed using Dialogical AR. It is not hard to see why Dialogical AR is prone to abductive reasoning as each iteration of the learning cycle feeds into subsequent iterations. It is possible, whilst moving between the natural attitude and the scientific attitude, to fall into abductive reasoning as demonstrated in Figure 50:

Figure 50 Abduction: the Logical Fallacy of Affirming the Consequent (Lee and Hubona 2008)



The possibility of falling into this reasoning is easy to understand given the longitudinal nature of the iterations in the Dialogical AR cycle. Thus, while the researcher's understanding of the social and historical context of the practitioners' situation may improve over time, it is possible that the researcher's interpretations alight upon him via abductive reasoning (Lee et al.

2008). While Lee and Hubona (2008) address this trap, and how to seek remedy, such remedy requires that an astute researcher would catch these mistakes on his or her own recognizance, which more easily said than done. The participatory nature of Dialogical AR may require somewhat extraordinary vigilance on the part of the researcher in order to avoid this logical fallacy.

The preceding discussion considered the potential pitfalls of using a Dialogical AR approach and recounted how some of these pitfalls were encountered and addressed in this research. The spirit and intent of utilizing Dialogical AR, as a method for conducting qualitative research, grounded in the interpretive mode of inquiry, was upheld throughout the researcher's field work in the practitioner setting.

6.6.3 A Principled Approach to Validating Interpretive Research Outcomes

Over the last 20 years, interpretive field studies have gained mainstream acceptance in Information Systems research. Many scholars of Information Systems have supported the legitimacy of using an interpretive approach to field studies as a worthy alternative to, or enhancement of, field studies conducting in a natural science mode of inquiry (Klein et al. 1999; Lee 1991; Lee 1999). Despite this acceptance, there is some confusion regarding how to critically evaluate interpretive fieldwork as the Interpretive model is not as well known as the Positivist model (Klein et al. 1999; Lee 1989). Thus, Klein and Myers (1999) provide a set of principles for conducting and evaluating interpretive field studies which are considered here.

Klein and Myers (1999), in offering a set of principles for interpretive work, do so in the hope of enlightening researchers about interpretive field work and highlight what an interpretive

mode of inquiry can and cannot do. Thus, as do Mårtensson and Lee (2004) regarding Dialogical AR, both Myers and Avison (2002) and Klein and Myers (1999) suggest that the aim of interpretive work is to achieve a rich understanding of context and process. In offering their guidelines, Klein and Myers (1999) present a synthesis of the literature on the interpretive paradigm in Information Systems research in the form of seven principles. Despite the potential drawbacks of using a "checklist" from a seminal paper, the potential benefits outweigh the risks. Thus, the Klein and Myers principles are enumerated and addressed as a "crosscheck" of the interpretive work in this dissertation.

Table 46 Summary of Principles of Interpretive Field Research

Summary of Principles of Interpretive Field Research	
1. The Fundamental Principle of the Hermeneutic Circle	This principle concerns the characteristic of the Hermeneutic circle where the researcher finds meaning in the whole-part comparison. Thus, understanding of the parts (originally, of a text) must be reconciled with an overall and "larger" meaning.
2. The Principle of Contextualization	This requires that the researcher always contrast interpretation with the historical and social context of the setting and situation under investigation. This is the "anti-generalization" clause.
3. The Principle of Interaction Between Researchers and the Subjects	This principle reminds the researcher that the interpretations represent a mutually and socially constructed reality only made possible by intersubjective understanding between the researcher and "subjects."
4. The Principle of Abstraction and Generalization	The researcher must relate and contrast the idiographic details of the setting and situation under investigation with a wider body of literature and understanding.
5. The Principle of Dialogical Reasoning	This principle refers to an iterative process by which the researcher reconciles between the assumptions of theory when entering the setting and situation under investigation and the actual outcomes of action.
6. The Principle of Multiple Interpretations	This principle encourages the researcher to remain open and sensitive to variances in the

Summary of Principles of Interpretive Field Research	
	first-level constructs conveyed by the participants in the interpretive study.
7. The Principle of Suspicion	This principle reminds the researcher to critically accept the interpretations and first-level constructs espoused by the participants as they may contain bias and distortion.

Table 46 provides useful advice, grounded in the literature, for interpretive researchers and those who will "consume" interpretive research. These principles will now be related and contrasted to this research in light of the evidence collected and the methods by which the evidence was obtained.

Principle of the Hermeneutic Circle

This principle is addressed in this research both in the mechanics of Dialogical AR and in the mode of inquiry chosen to analyze the dialogical evidence. Dialogical AR, given its iterative nature, encourages the researcher to engage in a dialogical partnership with the practitioner such that, together, the practitioner and researcher take various actions as both stimulus and response to a larger problem or set of problems. Thus, inherent within the Dialogical AR method is a whole-part comparison that is also characteristic of the Hermeneutic Circle. Furthermore, as a mode of analysis, the *constant comparison* activity in Strauss and Corbin's (1996) Grounded Theory, as a mode of analysis, encourages the researcher, while coding the dialogical evidence, to move from open and axial coding to categorization on by way of a process of constant comparison between new data and codes against existing codes and categories. This process allows for stronger categories and leads to theoretical sampling.

Principle of the Contextualization

Again, this research relies on Mårtensson and Lee (2004) in their requirement that Dialogical AR must consider the historical and social context inherent in the problem setting and situation under investigation. However, Mårtensson and Lee (2004) mean this in a bilateral sense such that the social and historical context is considered among all team members both from *praxis* and from *theoria*. Mårtensson and Lee (2004) advocate this approach to ensure knowledge heterogeneity and contextuality in the outcomes of Dialogical AR. This sensitivity to the context and history of the setting and situation under investigation ensures, in much the same manner as the principle of the Hermeneutic Circle, that analysis of the parts does not obscure the greater meaning embedded in the history and context of the situation as a whole. This principle has also aided a presentation of the evidence from the interpretive field study at SSC in a narrative fashion.

Principle of Researcher-Subject Interaction

Given the nature of Dialogical AR, this principle was virtually inescapable. In Dialogical AR, the research materials, or evidence, are invariably an outcome of a social construction which emerges during the Dialogical AR Partnership. Thus, that "… the 'data' are not just sitting there waiting to be gathered, like rocks on the seashore… Rather, *Interpretivism* suggests that facts are produced as part and parcel of the social interaction of the researchers with the participants" is assured in Dialogical AR (Klein et al. 1999:74) and was absolutely witnessed in the Dialogical AR Partnership. None of the Dialogical AR iterations would have been possible without meanings, and actions which arise from interpreting these meanings, which were socially constructed within the Dialogical AR Partnership.

Principle of Abstraction and Generalization

In its learning specification requirement, Dialogical AR ensures that some degree of generalization, specifically learning, ensues from the Dialogical AR Partnership. However, the idiographic nature of Dialogical AR restricts generalization. According to Lee and Baskerville (2003), it is most likely that interpretive field studies can only legitimately generalize to theoretical statements. This is congruent with Dialogical AR's requirement that the researcher specify learning, driven by interpretive analysis of the outcomes and consequences of action, for a body of literature pertaining to the setting and situation under investigation.

Principle of Dialogical Reasoning

With care not to conflate the use of the word "dialogical" used in this principle to Dialogical AR, this principle challenges the researcher to cyclically and iteratively question the prejudices they bring into the Dialogical AR Partnership. Moreover, this principle calls for transparency in reporting on research and to consider the fundamental philosophical assumptions which influenced the researcher upon entrance into the Dialogical AR Partnership. Also, this principle encourages the researcher to report how actual events challenged and changed these assumptions. In the case of SSC, the researcher was open with the practitioners at an early stage with respect to which theory was guiding his actions. This is not to say that the researcher gave a lengthy oratory on the finer points of the theories, but the researcher did paraphrase his fundamental philosophical assumptions, metaphorically and symbolically, as would be appropriate in the natural attitude of everyday life. Thus, by entering the Dialogical AR Partnership with *a priori* theory, Dialogical AR is distinguished from Grounded Theory and other forms of induction. Additionally, this principle addresses a reflective dialog the researcher conducts with his or her *a priori* philosophical assumptions during the course of the Dialogical AR Partnership. The very process by which this research was conducted strongly encouraged

such a dialog and, this dialog did, in fact, occur. Moreover, it will subsequently be demonstrated in this dissertation that the researcher's reflective dialog and "conversation with the materials" of his *a priori* theories did in fact alter his own understanding and assumptions based on these theories.

Principle of Multiple Interpretations

This principle and its manifestation in this research was a concern for the researcher. While this principle is fundamentally addressed in Dialogical AR such that the historical and social context is considered, in the case at SSC, the historical and social context was strongly biased by one individual in the Dialogical AR team, Daphne. Being that SSC is Daphne's company and that the history of the company largely centers on her, her first-order constructs resonated more strongly than those of her employees. The researcher, mindful of this social context, ensured that several dialogs were help with her developers alone where Daphne was not present. Daphne not only consented to this arrangement, but also encouraged it.

Principle of Suspicion

The researcher was often vexed with his own interpretations and the degree to which they were biased by Daphne's first-order constructs. Daphne's espoused theories of action were the prime mover for most interventions as, at the very least, these interventions were subject to her approval; to have introduced interventions without Daphne's blessing would have been damaging to the Dialogical AR Partnership. That being said, Daphne was always cooperative and giving of her time; (as was possible and practical) she believed that the Dialogical AR Partnership would bear fruit and benefit her company and development team. Klein and Myers (1999) caution against adopting a "false consciousness" which allows bias in one of among many

Espoused Theories (first-order constructs) dominate such that this perspective is assumed as absolute truth rather than one of among many which the researcher considers. Pleasantries, although useful for keeping the social order in the Dialogical AR Partnership and in adopting the natural attitude, are dangerous if not critically challenged in the scientific attitude.

Interdependence among the Principles

Klein and Myers (1999) assert a whole-part interdependence among these principles which is similar to and sympathetic to the Hermeneutic Circle. Remarkable is the extent to which Dialogical AR adheres to most of these principles by design. However, this is not likely accidental as Klein and Myers (1999) extensively cite Lee's (1989, 1991, 1994) work as influential in the development of their principles. It is of little surprise then, that Mårtensson and Lee's (2004) Dialogical AR is congruent to Klein and Myers' (1999) principles. Nonetheless, Klein and Myers' principles provide a useful post hoc checklist which provides occasion for reflection on the Dialogical AR effort at SSC as a whole.

6.6.4 A Principled Approach to Validating Dialogical Action Research Outcomes

In the spirit of Klein and Myers' (1999) principles for interpretive field studies, Davison et al. (2004) provide a similar set of principles to consider when conducting Canonical AR. Before the Dialogical AR iterations of this research are described and analyzed, the principles outlined by Davison Et. Al (2004) are considered. While there is danger that such a set of principles could be used in a manner that is too rote and automatic, "… adherence to a certain set of standards for conducting and reporting research in most (if not all) scientific disciplines has helped to establish their legitimacy and credibility" (Davison et al. 2004:83). Despite the

potential drawbacks of using a "checklist" from a seminal paper, the potential benefits outweigh the risks. Therefore, five principles listed by Davison et al. (2004), and their attendant criteria, are considered as the evidence from the Dialogical AR setting is discussed and outlined throughout the remainder of this chapter. The five principles for Canonical AR and their supporting criteria are listed below (Table 47):

Table 47 Principles and Criteria for Canonical AR (Davison et al. 2004)

Principles	Criteria
The Principle of the Researcher – Client Agreement	a. Did both the researcher and the client agree that Canonical AR was the appropriate approach for the organizational situation? b. Was the focus of the research project specified clearly and explicitly? c. Did the client make an explicit commitment to the project? d. Were the roles and responsibilities of the researcher and client organization members specified explicitly? e. Were project objectives and evaluation measures specified explicitly? f. Were the data collection and analysis methods specified explicitly?
The Principle of the Cyclical Process Model	a. Did the project follow the CPM or justify any deviation from it? b. Did the researcher conduct an independent diagnosis of the organizational situation? c. Were the planned actions based explicitly on the results of the diagnosis? d. Were the planned actions implemented and evaluated? e. Did the researcher reflect on the outcomes of the intervention? f. Was this reflection followed by an explicit decision on whether or not to proceed through an additional process cycle? g. Were both the exit of the researcher and the conclusion of the project due to either the project objectives being met or some other clearly articulated justification?
The Principle of Theory	a. Were the project activities guided by a theory or set of theories? b. Was the domain of investigation, and the specific problem setting, relevant and significant to the interests of the researcher's community of peers as well as the client? c. Was a theoretically based model used to derive the causes of the observed problem? d. Did the planned intervention follow from this theoretically based model? e. Was the guiding theory, or any other theory, used to evaluate the outcomes

Principles	Criteria
	of the intervention?
The Principle of Change through Action	a. Were both the researcher and client motivated to improve the situation? b. Were the problem and its hypothesized cause(s) specified as a result of the diagnosis? c. Were the planned actions designed to address the hypothesized cause(s)? d. Did the client approve the planned actions before they were implemented? e. Was the organization situation assessed comprehensively both before and after the intervention? f. Were the timing and nature of the actions taken clearly and completely documented?
The Principle of Learning through Reflection	a. Did the researcher provide progress reports to the client and organizational members? b. Did both the researcher and the client reflect upon the outcomes of the project? c. Were the research activities and outcomes reported clearly and completely? d. Were the results considered in terms of implications for further action in this situation? e. Were the results considered in terms of implications for action to be taken in related research domains? f. Were the results considered in terms of implications for the research community (general knowledge, informing/re-informing theory)? g. Were the results considered in terms of the general applicability of Canonical AR?

In the following sections of this chapter, it will be demonstrated that these principles and criteria are reflected in this study.

Principle of the Researcher-Client Agreement

The researcher respected the sanctity of the practitioners' privacy and sought practitioner partnership at all times. The mechanics and protocols of Dialogical AR were explained to the practitioners at multiple intervals such that our dialog and subsequent actions were understood in the larger context of the aims of the research and the research method. Thus, informed consent

was solicited, agreed upon and signed by the participants whereupon nature of Dialogical AR was made clear to the practitioners. During the month of July 2008, as the researcher conducted his initial observations, the researcher called upon at an SSC company meeting and took 20 minutes to explain his purpose and what the practitioners might gain by entering into the Dialogical AR Partnership. From this point forward, the practitioners were given regular reminders concerning their role in the partnership and how Dialogical AR would guide many of our interactions.

As to evaluation measures for the research outcomes, such measures were clearly stated upfront: SSC desired a method to address quality, productivity, learning, skills development and client communication and the researcher desired a basis for demonstrating the applicability of *Reflective Practice* as a theoretical basis for a learning system in the use of agile methods. These outcomes were explicit and yet also iteratively refined. While Davison et al. (2004) caution against emergent and post hoc measures of success, the emergent nature of using Dialogical AR and Lee's DSAR framework challenged this advice. However Davison et al.'s caution is taken under advisement and warrants discussion as a limitation and weakness of this dissertation. However, the advice seems somewhat out of step with the overall emergent intent of action research.

Principle of the Cyclical Process Model

Mårtensson and Lee (2004) do not wholly adopt the classic Canonical AR Cyclical Process Model (CPM) directly, but they do concede its influence and site Baskerville's (1999) illustration of the process. Davison et al. (2004) discuss and recommend an *a priori* and independent diagnosis of the organizational situation which is what was done in this research as

the researcher devoted some of June 2008 and all of July 2008 for this purpose. Thus, the Dialogical AR Partnership began subsequent to a base-line "read" of the organizational situation and reflection on practitioner-researcher interaction prior to entering the CPM. Thus, entrance into the Dialogical AR Partnership proceeded as a result of the initial and independent investigation and "profiling" of the organization as is recommended by Davison et al. (2004). All other recommendations inherent within this principle - action planning, action taking, and reflective evaluation - were a natural result of the Dialogical AR process. Davison et al. also call on the researcher to state explicit reasons for continuing with subsequent cycles and explicit reasons for exiting the CPM. Both of these issues are addressed and evident in the discussion of the evidence.

Principle of Theory

Throughout the Dialogical AR Partnership, the researcher was clear about which set of theories were guiding his interventions. The selected bundle of theory is amply supported in the literature and sufficiently and successfully guided the research process. Manifested within these theories are several models which predict and illustrate the parameters of the observed problems and how the theories provide redress of these problems. Over time, the appropriateness of the theories of Argyris and Schön (1974, 1978, and 1996) and Schön (1983, 1987) was increasingly evident. Accordingly, the interventions were rooted in the guiding theory and constituted the structure of the designed artifact. Again, the advice from Davison et al. (2004) in this area is good but somewhat restrictive with respect to the researcher's ability to react to emergent themes which arise in dialog. However, the overarching principle of the necessary role of theory is not disputed and is evident in the conduct of Dialogical AR in this research.

Principle of Change through Action

Davison et al. (2004) speak to the necessity of change in action in doing action research. Socially constructed meaning is at the heart of interpretive research (Klein et al. 1999) and mutually sanctioned change is at the heart of Dialogical AR (Mårtensson et al. 2004). Within the Dialogical AR Partnership, all interventions were the result of agreement and understanding in the Dialogical AR Partnership: problems were framed against shared understanding between partners, planned actions were taken within this understanding and client consent arose naturally in the dialog. Again, it is possible that Davison et al.'s (2004) criteria do not directly map to Dialogical AR as their assumptions concern Canonical AR which Mårtensson and Lee (2004) carefully distinguish Dialogical AR from. Thus, the principle of change through action, as it pertains to practitioner involvement is virtually assured in Dialogical AR. With respect to "before and after" readings and a chronological account, each of these are addressed and present in the dialogical evidence.

Principle of Learning through Reflection

Davison et al.'s (2004) last principle focuses on the specification of learning as a critical by-product of any action research effort. Within the tenets of Dialogical AR, this principle is not in dispute as Dialogical AR ensures this principle within its structure. Without regular reflective dialog in the Dialogical AR Partnership, Dialogical AR does not progress. Thus, reporting progress and outcomes, client reflection and planning for additional and future action are all embedded. In dialog, the practitioners at SSC were always aware of our mutual progress and always apprised, also through dialog, of where we were at, where we had been and where we were headed. Davison et al. (2004) also address the specification of learning for the research

community with respect to informing and re-informing theory and this advice is addressed subsequently in this report. However, the researcher's activities related to transcription, coding and comparison provided ongoing opportunities for reflection on the effects of the theory-driven interventions and in planning future dialog and intervention. In this sense, this particular research effort is compliant with this principle.

Reflecting on the Principles

Davison et al.'s (2004) proffered principles provide welcome structure and direction as action research gains traction as a means for investigating Information Systems phenomena. Their motivation, similar to that of Klein and Myers (1999) is sound as they seek to increase understanding in the "consumption" of rigorous and relevant inquiry conducted using action research. Thus, Davison et al. (2004) seek to distinguish quality work and introduce a means for establishing rigor and standards in the use of action research. Whether these standards are entirely appropriate for Dialogical AR is not certain and certainly warrants further discussion on the particulars of Dialogical AR as relates to Canonical AR.

Considering the Particulars of Dialogical AR

There are distinguishing principles of Dialogical AR which set it apart from the Canonical AR discussed by Davison et al. (2004). Mårtensson and Lee (2004) place, above and beyond Canonical AR, principles for rigor and relevance rooted in the epistemology of phenomenology where clear distinction is made between the natural and scientific attitude. In Dialogical AR, both the natural attitude of the practitioner and the scientific attitude of the researcher are required to create a partnership whose union provides synergistic advantage. Furthermore, in Dialogical AR the researcher approaches the research project with equanimity

such that the needs of all participants are addressed in good faith. Apart from achieving progress primarily through (but not limited to) reflective dialog, Dialogical AR also holds equality within the research team as sacrosanct in terms of knowledge heterogeneity and contextuality. The distinguishing features of Dialogical AR are shown in Table 48.

Table 48 Distinguishing Features of Dialogical Action Research (Mårtensson and Lee 2004)

Concept	Relevance
Adopting the Scientific Attitude	As relates to Schutz and phenomenology, this allows the Dialogical AR team to utilize the researcher's expertise in order to discover second-level constructs through observation and subjective interpretation.
Adopting the Natural Attitude of Everyday Life	As relates to Schutz and phenomenology, this allows the Dialogical AR team to utilize the practitioner's expertise in order to reveal and examine first-level constructs through action, dialog and participation.
Accepting the Role of Social and Historical Context	Recognition that both sides of the Dialogical AR team bring their own ontology, epistemology, history and context to the team.
Understanding the Social and Historical Context	Insofar as the researcher bears the burden to specify learning for theory and practice, it is also imperative that the researcher understand the social and historical context in order to utilize interpretations and analysis which leads to learning.

The body of this chapter has addressed and provided an argument that it has been in compliance and adherence to the distinguishing features of Dialogical AR in the case at SSC.

6.7 Reflections on the Evidence

As a participant in the Dialogical AR Partnership, the perspective of the researcher is well represented in the narrative descriptions of the dialogical and observational evidence presented in Chapters 5 and 6. However, despite 30 plus hours of transcribed dialog and despite using ample passages of the transcripts in this dissertation, it is still possible that the voices of

practitioners have not been sufficiently heard. The researcher, charged with the task of diagnosing, suggesting theory-driven interventions, interpreting the evidence, and developing second-order constructs, casts what may appear as a series of overly-critical assessments of the practitioners' actions. This was, perhaps, most apparent to the researcher during coding and analysis and during transcription. As the researcher was charged with "teaching" XP, there were times when it seemed like an inordinate portion of the dialog was soliloquy from the researcher. At other times, the researcher was somewhat impatient when the practitioners did not try or adopt interventions to the researcher's expectations. In these ways, the Dialogical AR Partnership exposed the human face of Dialogical AR and qualitative and interpretive work in general, and presented the pitfalls discussed in a previous section of this chapter.

As the practitioners at SSC cannot speak for themselves in this report, I will attempt to speak on their behalf and in their defense as many of the "theories-in-use" ascribed to the practitioners in the researcher's interpretations, casts a negative light on the practitioners. To the contrary, I firmly assert that practitioners conducted themselves professionally at all times and acted in a sincere desire to adopt a new methods which they believed would benefit them. Daphne, despite carrying the pressure and burden of running her business, demonstrated caring and consideration towards her employees which could not be construed as less than sincere. As the greatest sign of their sincerity, the practitioners were gracious with their time and in sharing their everyday life with the researcher. If an intervention or concept was partially considered or attempted, it was not entirely out of disinterest, but arose out of a constant and increasing level of business. In considering their own situation and offering a first-hand view of their processes to the researcher, the practitioners aptly engaged in reflective processes and demonstrated true interest in their own development and in the development of our partnership.

Fred and Johnny provided their time and insight during several sessions of *Pair Programming* which provided a "toe hold" onto which the researcher could position and develop the most important theory-driven interventions concerning *Reflective Practice*. Daphne, offered free and open access to her organization and employees and encouraged the practitioners to avail themselves of the opportunities for reflection made possible in our partnership. Daphne cared about what her employees thought and expressed her desire that her employees were open and honest in their reflections, especially those which arose in the dialogs in which she was not a participant.

As an interpretive researcher in the Dialogical AR team, I often felt like I was shining a flashlight into the dark and unkempt closets that are usually out-of-sight and out-of-mind. Such experiences are not always pleasant and can reveal things which the practitioners would rather not consider. Thus, as an honest exposure of the practitioners' habits and processes can cause embarrassment, it is important that the practitioners' identities are well obfuscated.

Also, as qualitative and interpretive research effort, the use of Dialogical AR encouraged a narrative presentation of the dialogical evidence. In this narrative, the particulars of SSC's business practices and development processes have been magnified to an extent which promotes reflection and analysis. When practitioners are thusly magnified and closely examined, positive and negative things become more readily apparent. From the natural attitude, this seems intrusive where interpretive conclusions and theory-driven interventions can seem presumptuous. However, the scientific attitude required close and methodical examination and reflection.

As the researcher, I acknowledge the effects of my participation in the partnership although it was short-lived by necessity and design. For the practitioners, the researcher's

presence is a temporary intrusion into their lives and a cause for self-examination and reflection. The researcher, in establishing informed consent as required by Internal Review Board policies, encouraged the practitioners to end the partnership if they were uncomfortable or if they felt the interventions too disruptive. I am grateful that the practitioners dutifully and diligently continued the partnership and I am pleased with the success of our partnership.

CHAPTER 7 Analysis of the Designed Artifact

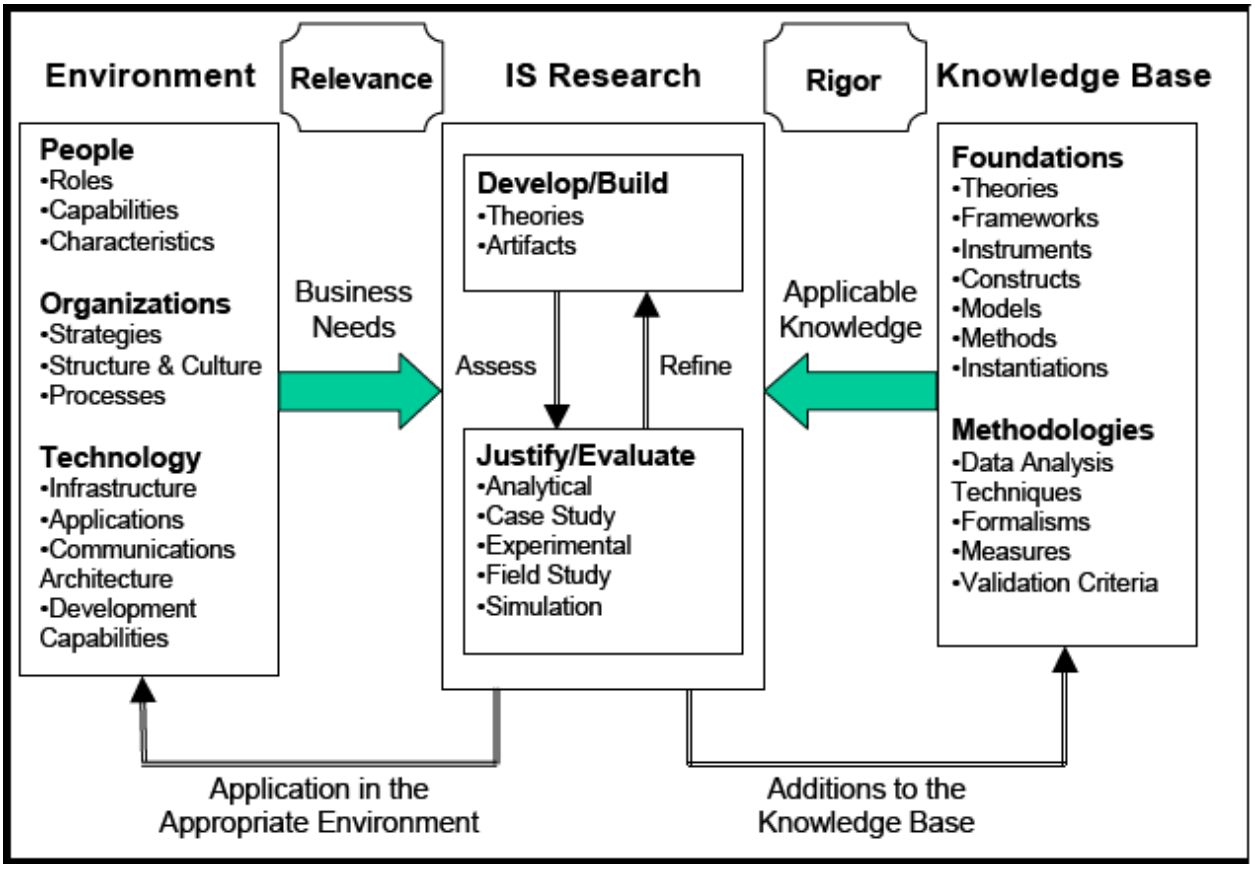
The most evident outcome and by-product of this research is the designed artifact: the Reflective-Agile Learning Model and Method (RALMM) for small-team and small-shop software development. This outcome, as a product of design and action research, is an embodiment of the practical and scholarly outcomes from the Dialogical AR Partnership and constitutes a centerpiece about which the answers to the research questions and the contributions of this research can be discussed. The designed artifact, RALMM, is a useful outcome and contribution to the practitioner; to scholarship; Dialogical AR; XP and agile methods; and to *Reflective Practice* and the emergent epistemology of reflective design.

The sections of this chapter describe and evaluate the designed artifact and proceed as follows. First, an explication of the Reflective-Agile Learning Model and Method including how RALMM relates to and improves on XP. Next, RALMM is considered as a manifestation and validation of the theories introduced. This followed by validation of the designed artifact according to accepted guidelines for design science. The chapter concludes with a discussion of the paradigmatic and methodological implications of the designed artifact as developed in the context of a qualitative and interpretive mode and method of inquiry.

7.1 A Reflective-Agile Learning Model and Method Methodology

The researcher entered into the Dialogical AR Partnership with clear motivations and expectations to demonstrate the applicability of theories from Argyris and Schön (1974, 1978, 1996) and Schön (1983, 1987) as relates to the effectiveness of agile methods in the small-team and small-shop software development environment. These motivations are consistent with Hevner et al.'s (2004) Information Systems Research Framework (Figure 51) whereupon business needs in an organization precipitate the development of theories or artifacts of design which can be iteratively assessed and refined such that the original business-oriented problem is addressed by the design artifact. However, to distinguish what appears, on the surface, to be mere consulting, Hevner et al. (2004) also point out the influence of scholarly knowledge and theory in the process of design. Also, Hevner et al. (2004) suggest that the outcomes of design research are meant to inform and re-inform a body of scholarly knowledge, which ensures that design science efforts are also research efforts.

Figure 51 Hevner et al.'s Information Systems Research Framework (2004)



What is immediately evident in Hevner et al.'s (2004) model is the striking similarity it has to action research in terms of the mutual goal of relevance in real-world problem solving and rigor in the specification of knowledge useful to a community of scholarship and science. Furthermore, this research utilizes Lee's (2007) DSAR framework which recognizes the symbiotic and synergistic possibilities in "actuating" design science with action research. Other scholars have also recognized the complementary nature of design science and Action Research (Järvinen 2007), which provides further validation for Lee's (2007 DSAR framework.

Recognizing the congruent relationship between Hevner et al.'s model of design science research with respect to rigor and relevance and the pursuit of these goals in action research, Lee's (2007) DSAR framework is used in this research in order to "actuate" design science.

That is, Dialogical AR provides more specified methodological guidance from which design science can be conducted.

The remainder of this section will present and outline the Reflective-Agile Learning Model and Method and present an account on how the DSAR framework was used to iteratively develop RALMM in the context of the Dialogical AR Partnership.

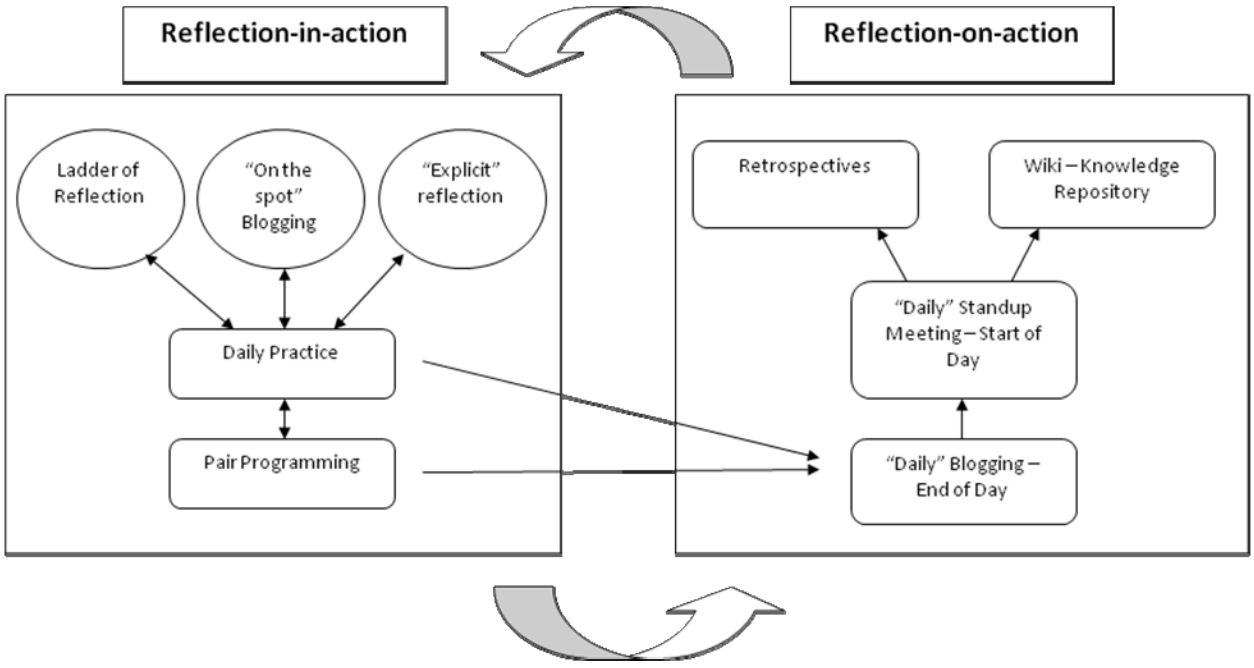
7.2 Elements of the Artifact

The Reflective-Agile Learning Model and Method is a manifestation of Schön's *Reflective Practice* (1983, 1987) and Argyris and Schön's (1974, 1978, 1996) theory of action as they pertain to *Organizational Learning*. This "bundle" of theory is applied to the *Collective Code Ownership* element of XP, specifically to *Pair Programming* and the *Daily Standup Meeting*, such that an explicit process of learning through the shared context of common experience and through an explication of tacit knowledge into a knowledge repository was possible. This daily learning process is enabled through the use of common Internet and World Wide Web communication tools such as Weblogs (blogs) and Wiki Wikis (wikis). Additionally, the developers regularly utilized Schön's (1987) *Ladder of Reflection* in order to develop shared context and to "uncover" tacit knowledge. Furthermore, the entire process cumulatively adds to individual and team repertoire in situations where, whether as individuals or as a group, the practitioners experience learning while solving new and novel problems or otherwise develop coping strategies for uncertainty and change.

7.2.1 Explicating the Model

The general relationship between the introduced habits and technologies present in the mutually-designed and developed artifact and the theoretical inputs are shown in Figure 52.

Figure 52 Elements of the Reflective-Agile Learning Model



These elements (Figure 52) are the outcome of an iterative process of design, dialog and reflection undertaken within the Dialogical AR Partnership. This is to say that the model expressed in Figure 52 is not a post hoc derivation of the researcher's interpretation and analysis, but rather the result of reflective reasoning and experimentation within the Dialogical AR Partnership. As a model for reflective and agile learning, RALMM provides an abstract representation of the relationships and interactions among the elements in Figure 52, to be used as a learning system by the practitioners at SSC. The model describes learning activities related to Schön's *Reflection-in-action*, where the practitioners conduct on-the-spot experiments and

other “conversations” with the situation and materials in order to adapt to change, uncertainty and novelty.

As a follow-up and in order to complete the reflective cycle, the practitioners utilize their team and personal blogs to gather the reflections and reactions of the day for *Reflection-on-action*. An individual process of *Reflection-on-action* occurs as a practitioner reviews the previous day’s team and personal blog entries. A team process of *Reflection-on-action* occurs during the next *Daily Standup Meeting* when the team leader, called a “coach” in XP or a “scrum-master” in Scrum, focuses *Reflection-in-action* and *Reflection-on-action* in order to prepare the team for the new day’s activities. In their reflective dialog during their *Daily Standup Meeting*, the team “vets” reflections-in-action and reflections-on-action, gleaned from the previous day’s blogs and from the reflective dialog at hand, and accumulates their conclusions and learning into their team Wiki. This process is circular as the team then moves into their routines of daily practice informed by that morning’s reflective dialog. Subsequently, the practitioners’ daily *Reflection-in-action* modify that morning’s understanding as new experience and reflection changes and shapes the team’s repertoire on a daily basis. Thus, the Reflective-Agile Learning Model is the result of the sum of the theory-driven interventions, and reflective evaluations of these interventions.

7.2.2 Explicating the Method

While Figure 52 presents the principle elements of the Reflective-Agile Learning Model, it does not describe how the method by which this model is used in their daily practices where the model is used as an integral part of their use XP. Thus, when the model is used in specific

and explicit goal-seeking or algorithmic steps, the model is used as a method. Figure 53 presents the methodical steps, informed by the Reflective-Agile Learning Model, which explain how RALMM fits into and works alongside XP.

Figure 53 The Reflective-Agile Learning Model and Method evidenced in SSC's Adaptation of XP

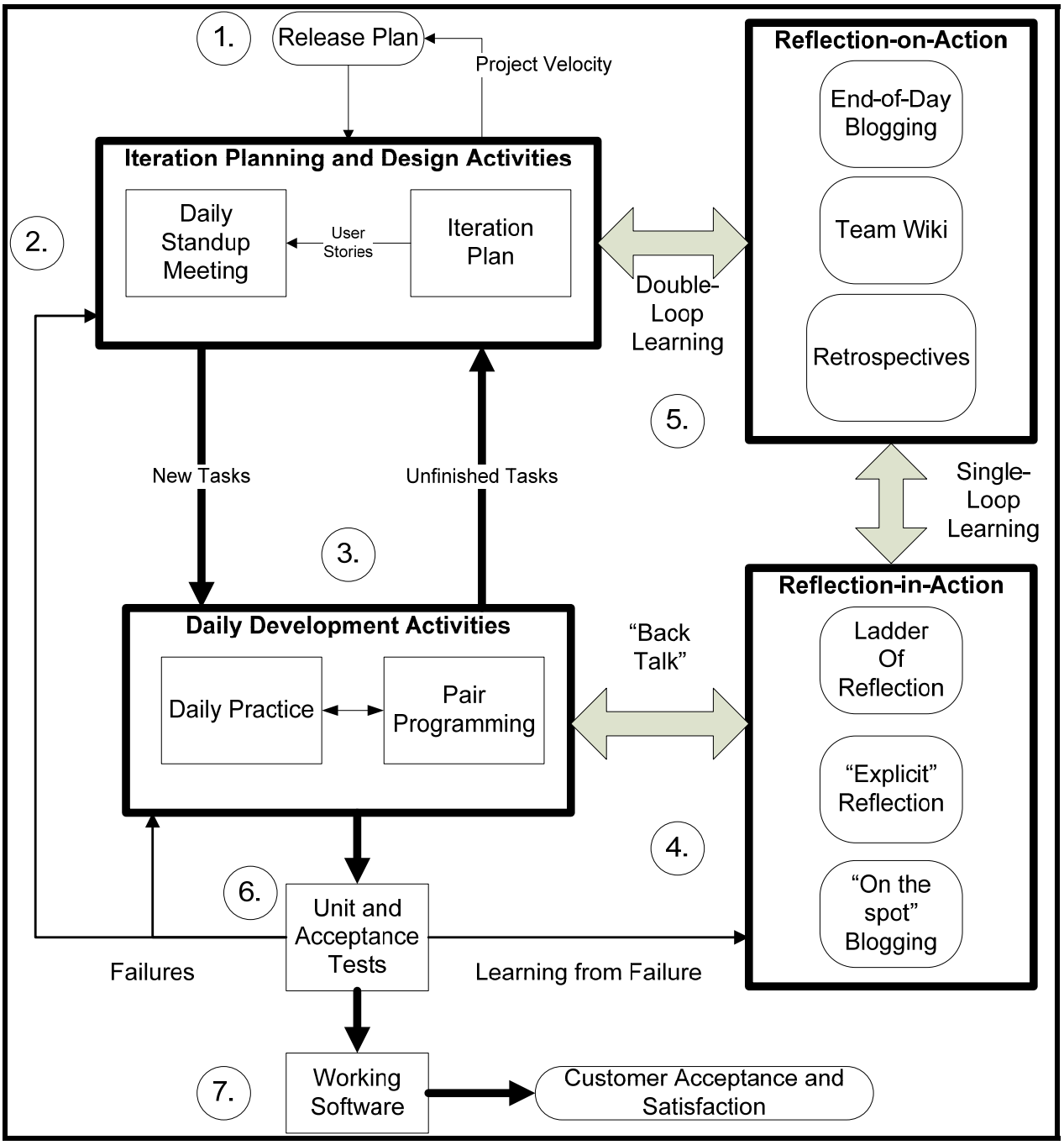


Figure 53 presents the Reflective-Agile Learning Model and Method as it is situated within SSC's use of XP at the conclusion of the Dialogical AR Partnership. This constitutes a learning method as the steps of the learning model are situated within SSC's software development process. As a method can be considered a sequence of steps to produce an intended outcome, the steps of the Reflective-Agile Learning Method are described in Table 49.

Table 49 Steps in the Reflective-Agile Learning Method

Steps of the Reflective-Agile Learning Method
1. Direction from the Release Plan – Overall project velocity and goals are expressed within the release plan which drives the tempo and direction of developer activity throughout the project.
2. Iteration Planning and Design Activities – Developer tasking arises from the Daily Standup Meeting and Iteration planning activities such as the Planning Game. The new tasks for daily development arise from this step. Also, individual and team learning influences this step by way of regular and daily activities related to *Reflection-on-action*.
3. Daily Development Activities – Developers acquire new tasks from planning activities and work towards the completion of these tasks by activities in daily practice and pair programming which constitute the team's Collective Code Ownership. These daily development activities become the causes for and inputs to *Reflection-in-action*. Thus, this step regularly feeds step 4 constantly as developers engage in their daily routines.
4. Reflection-in-Action – Developers use daily practice as the occasion to engage in *Reflection-in-action* especially in cases where novelty, change and uncertainty contradict or challenge the practitioners' extant repertoire. Through techniques such as the "Ladder or Reflection," explicit personal reflection and, in result of these, ad hoc and "on-the-spot" blogging, the developers engage in regular *Reflection-in-action* throughout the workday. *Reflection-in-action*, which constitutes and facilitates the developer's reflective "conversation with the materials" of designing is also an occasion for reflection to feed directly back into designing and action. The duo in engaged in pair programming will alter their designing as a result of their own reflective dialog and also as a result of their conversation with the materials.
5. Reflection-on-Action – Developers adopt daily habits which enable an *ex post facto* reflective process where *Reflection-in-action* and the consequences of daily action are considered for *Organizational Learning*. Being mindful and apprised of the requirement to engage in *Double-Loop Learning*, the practitioners first consider opportunities for *Single-Loop Learning* informed by their *Reflection-in-action*. As a result of *Reflection-on-action*, manifested in periodic Retrospectives, "feeding" and maintaining the team Wiki, and daily end-of-day blogging, the team engages in *Double-Loop Learning*.

Steps of the Reflective-Agile Learning Method

Double-Loop Learning, a byproduct of *Reflection-in-action* and *Reflection-on-action*, now informs the team's Daily Standup Meeting where new understanding and shared context is confirmed and incorporated into the daily plan. Team reflective dialog may then re-inform *Reflection-on-action* or modify tasks for daily development activity.

6. **Unit and Acceptance Testing –** An outcome of daily development activity are finished tasks which are then subjected to testing. The outcomes of these tests are also a part of the learning system. Acceptance Test failures commonly become new inputs for planning and design and Unit Test failures serve as a form of *back-talk* which the developers respond to in daily practice. In both cases, failure becomes an input to *Reflection-in-action* which ultimately informs *Reflection-on-action*.

7. **Small and Frequent Releases of Working Software** – An important outcome of the Reflective-Agile Learning Model and Method is small and frequent releases of working software which lead to customer acceptance and satisfaction. With both the software development and learning processes working together, the software development team increasingly ensures that this outcome is achieved regardless of changes in the environment.

This section now continues with a description on how the DSAR framework is used to create the Reflective-Agile Learning Model and Method.

7.3 Iteratively Designing the Artifact

Lee's (2007) DSAR framework is informed by the March and Smith (1995) framework for design science in information systems research. Lee's DSAR framework suggests how action research can drive design science research by using the iterative cycles of action research to undertake the steps described in the March and Smith (1995) framework. Thus, the DSAR model provides an explication of the design activities undertaken in each step of the Canonical AR cycle discussed in Baskerville (1999). The iterative design of the designed artifact is now accounted according to the four iterations of the Dialogical AR cycle (Figure 32). Each of the activities and outputs of design science research are also discussed. Table 50 provides a key to

the outcomes and activities within the DSAR framework undertaken during the Dialogical AR Partnership.

Table 50 Key to DSAR Activities in this Research

A. **Constructs**:
    a. Learning Organization
    b. Reflective Practice

B. **Model**: The working model of the reflective practitioner approach for the learning organization

C. **Method**: The reflective-agile method
    a. Extreme Programming
    b. Ladder of Reflection

D. **Instantiation**: The first introduction of the Reflective-Agile method

E. **Evaluation (Constructs and Model)**: A Strauss and Corbin (1998) approach to coding and categorizing

F. **Theorize (Model and Method)**: Use of the Dialogical AR cycle to diagnose, act, specify and learn.

The grayed-out cells in the DSAR framework in all subsequent tables represent the areas which were not addressed by a particular partner in the Dialogical AR Partnership. Also, each of the terms in the DSAR framework are expressed as defined by Lee (2007), who developed DSAR based on design science concepts in March and Smith (1995) and Hevner et al. (2004).

7.3.1 DSAR Iteration One: Early Discovery and Diagnosis

Table 51 DSAR Activities in Iteration One

Cycle One (July 2008 – September 2008)			**Design-Science Research Activities in Action-Research**			
			Build	**Evaluate**	**Theorize**	**Justify**
Action Researcher	***Design-Science Research Outputs***	Constructs	A	A,E		
		Model				
		Method				
		Instantiation				
Practitioner	***Design-Science Research Outputs***	Constructs				
		Model				
		Method				
		Instantiation				

During the first iteration, the Dialogical AR partners were primarily concerned with developing a base-line understanding of SSC's extant methods and determining the feasibility of introducing an agile software development method such as XP. At this stage, the theory-driven interventions served as a vocabulary for understanding learning processes as the practitioners explored XP. Thus, concepts such as *Reflective Practice, Organizational Learning*, Single and *Double-Loop Learning*, and *Model I*/*Model II* behavior were introduced into the vocabulary of the Dialogical AR Partnership.

Lee (2007) reminds the researcher using the DSAR framework to carefully build and evaluate constructs using careful field study and qualitative analysis such as the Grounded Theory techniques described by Strauss and Corbin (1998) and used in this dissertation. Thus the coding and analysis used to narrow SSC's issues to those related to learning and communication were informed both by the theory the researcher entered with at the onset of the Dialogical AR Partnership and by the results of the open and axial coding. Thus the constructs

developed were a result of the categories which emerged during Grounded Theory coding and those imputed by the researcher's theoretical perspective.

Given the iterative nature of Dialogical AR, coding and categorization "saturation" could only be reached for the current iteration; new information in subsequent iterations would have additional impact which influenced and changed codes and categories. Yet, as described by Strauss and Corbin, a longitudinal study such as this research will reach saturation over time. Thus, the first two iterations yielded the majority of categories to emerge over the course of the Dialogical AR Partnership. Table 52 demonstrates which activities within the DSAR occurred during the first iteration of the Dialogical AR cycle.

7.3.2 DSAR Iteration Two: Adopting XP

Table 52 DSAR Activities in Iteration Two

Cycle Two (October and November 2008)			**Design-Science Research Activities in Action-Research**			
			Build	**Evaluate**	**Theorize**	**Justify**
Action Researcher	***Design-Science Research Outputs***	Constructs		A,E		
		Model	B	B,E		
		Method				
		Instantiation				
Practitioner	***Design-Science Research Outputs***	Constructs				
		Model				
		Method				
		Instantiation				

As SSC were well into the process of adopting XP by the end of the 2nd Dialogical AR cycle, the researcher continued to introduce the Reflective-Agile constructs while also considering the effectiveness of these constructs as SSC adopted and adapted XP to their situation. Through the Dialogical AR Partnership, the researcher developed a clearer picture regarding the impact of his interventions while SSC learned and adopted XP. Of course XP has

its own set of constructs embedded within the method and, as agile methods in general are an important element of the Reflective-Agile Learning Model and Method, the XP constructs were also under evaluation. In all cases, the theory-driven constructs were used to guide interventions and not offered as solutions in and of themselves.

Within the second cycle, the researcher began to develop the Reflective-Agile Learning Model based on practitioner reaction, researcher interpretation and feedback in dialog. Elements of the model were supported and refined as the constructs were also iteratively refined. Therefore, the relationship between key elements in XP which were working for SSC – *Daily Standup Meeting*, *User Stories*, *Pair Programming*, *Spike Solutions* – were iteratively taking shape as the Dialogical AR cycles continued. The dialogical evidence also revealed that *Reflective Practice* contained important constructs which highlight and complement the constructs within Argyris and Schön's (1974, 1978, 1996) Theories of action for organizational where both served as essential guides for the Reflective-Agile Learning Model.

7.3.4 DSAR Iteration Three: Adapting XP

Figure 54 DSAR Activities in Iteration Three

Cycle Three (December 2008 and January 2009)			**Design-Science Research Activities in Action-Research**			
			Build	**Evaluate**	**Theorize**	**Justify**
Action Researcher	***Design-Science Research Outputs***	Constructs				
		Model				
		Method	C	C,E		
		Instantiation				
Practitioner	***Design-Science Research Outputs***	Constructs				
		Model				
		Method	C	C		
		Instantiation	D	D		

By the third iteration, the Reflective-Agile Learning Model had been thoroughly introduced and evaluated in the Dialogical AR Partnership. As the partners developed a clear understanding of the constructs and model, the partnership then developed the means by which the model would be "actuated" into steps which the practitioners could incorporate into their daily routine. Thus, the Reflective-Agile Learning Model would now incorporate a method by which the relationships suggested in the model could be made useful in the use of XP or some other agile method. In this sense, both sides of the partnership were actively involved in shaping the development of the method.

While the researcher was able to work with the practitioners to specify and refine a model and method, the method is best substantiated when the practitioners actually use the model and method in practice. Thus, an instantiation of the Reflective-Agile Learning Model and Method, whereupon the practitioners would use RALMM to some effect in their organization left for the practitioners to develop. Chapter 5 documents the degree to which SSC accepted the interventions and how thoroughly these interventions were attempted. By the end of the last iteration, the practitioners had tried all aspects of RALMM and yet also left room for further adoption and use of their new learning system.

7.3.5 DSAR Iteration Four: Towards Reflective Practice

Figure 55 DSAR Activities in Iteration Four

Cycle Four (January 2009 to February 2009)			**Design-Science Research Activities in Action-Research**			
			Build	**Evaluate**	**Theorize**	**Justify**
Action Researcher	***Design-Science Research Outputs***	Constructs			F	x
		Model			F	x
		Method			F	x
		Instantiation			F	x
Practitioner	***Design-Science Research Outputs***	Constructs				
		Model				
		Method				
		Instantiation				

Lee (2007) indicates that the *Theorize* and *Justify* steps in the DSAR framework are the sole purview of the researcher in the Dialogical AR Partnership – it is the researcher who must specify learning for the body of knowledge with which he entered into the Dialogical AR Partnership. As to the "theorize" activity, Lee (2007) offers the following: "I define the design-science research activity, *Theorize*, to refer to formulating an explanation for why a construct, model, method, or instantiation did not perform as expected" (p.56). Similarly, Lee quips on theorize: "I reserve the verb 'to theorize' to refer to what, in the *Build* and *Evaluate* activities, might have gone wrong and what might be done differently…" (p.56). Lastly, Lee (2007) offers that the "justify" research activity calls for the researcher to establish plausibility for his own explanation.

March and Smith (1995) provide their Design Science Research Framework as a 4 by 4 matrix of design science activities which Lee's (2007) DSAR framework is built upon. March and Smith (1995) say the following concerning a researcher's use of their framework:

> *Researchers can build, evaluate, theorize about, or justify the theories about constructs, models, methods, or instantiations. Different cells have different objectives and different methods are appropriate in different cells. Research efforts often cover multiple cells. Evaluation of research should be based on the cell or cells in which the research lies.(March et al. 1995:260)*

With this in mind, it must be stated that this research has not engaged in the *Justify* activity and this goal is reserved for and discussed as future research.

Lee (2007) suggests that problems, shortcomings and unexpected behaviors should be the basis of the theorizing activity in the DSAR framework. As such, with respect to the model and method, the problems encountered related to *Reflection-in-action* were: that the practitioners wouldn't blog regularly and that the practitioners wouldn't *Pair Program* regularly. Each of these unexpected consequences of action is mostly related to the practitioners' lack of time. However, it is possible that other deeper causes related to Argyris and Schön's (1974) *Model I* and *Model II* behavior patterns as related to *Organizational Learning.* As the DSAR *Theorizing* research activity is directly related to Dialogical AR's *Specify Learning* step, each of these are deferred to the conclusion in Chapter 8.

7.4 Improvements over Time

Mårtensson and Lee (2004) make it clear that Dialogical AR must balance rigor and relevance in co-opting the researcher's expertise and practitioners' expertise to address a real-world problem faced by the practitioner and seek remedy for the problem to the practitioners' satisfaction. Furthermore, over time, both the researcher's and practitioners' expertise must be improved such that the researcher's theory-driven interventions enhance the practitioners' understanding of the problem and theory brought to the Dialogical AR Partnership by the

researcher tested (Mårtensson et al. 2004:518). Thus, the evaluation criteria for Dialogical AR are shown in Table 53:

Table 53 Evaluating the Outcomes of Dialogical AR

Dialogical AR Evaluation Criteria	Description
Criterion I	The practitioner considers the real world problem facing him or her to be solved or satisfactorily remedied
Criterion II	There be an improvement in the practitioner's expertise
Criterion III	There be an improvement in the scientific researcher's expertise

Chapter Six has provided evaluation criteria for the conduct of this research by an accepted set of principles for interpretive field studies (Klein et al. 1999). However, the research must also be evaluated with respect to the designed artifact where the design and use of the artifact ground any improvements to the researcher's theoretical understanding. Thus, successes and failures in the design and use of the RALMM is used to inform an explanation of how the researcher's expertise was improved. Mårtensson and Lee (2004) are emphatic that unexpected outcomes and failures should provide the basis for adjustments to the original theory and that subsequent iterations of the Dialogical AR cycle provide a basis for testing the researcher's adjusted theories or second-order constructs.

Figure 53 suggests that improvements to the practitioners' problem and expertise and the researcher's expertise iteratively improve upon each cycle of the Dialogical AR process. Thus, at the conclusions of successive iterations, changes in expertise are demonstrated and described. This section will now continue, in the style of Mårtensson and Lee (2004), and describe what

overall improvements were realized by February 2009, at the end of the Dialogical AR Partnership versus the conditions which were observed in July of 2008.

7.4.1 How the Designed Artifact Remedies the Real World Problem

By February 2009, when the researcher exited the Dialogical AR Partnership, the practitioners at SSC were quite pleased with their progress and their new methods for software development and learning. As demonstrated in Chapters 5 and 6 of this dissertation, most of SSC's stated problems were reduced, by way of the Dialogical AR method, to matters pertaining to learning and communication (see Figure 48). Additionally, an evaluation of the practitioners' feedback in dialog suggested that the introduction of XP met with measurable success and the addition of RALMM was also well-received. Thus the sum of interactions, dialogs and interventions during the Dialogical AR Partnership met, in the practitioners' words, their stated desires for a software development method. Furthermore, RALMM also addressed the new issues which arose by way of the researcher's interpretation of the practitioners' theories-in-use.

Therefore, there were three primary real-world problems to emerge during the Dialogical AR Partnership which are listed in Table 54.

Table 54 Improvements to the Real World Problem

Real World Problem	Baseline conditions (T=July 2008)	How RALMM addresses the problem (T=February 2009)
SSC's desire for a method	SSC has no documented or espoused method beyond that which "lives in Daphne's" head and is transmitted informally and by rote.	While XP is the software development method adopted, RALMM provides a learning system which will allow SSC to adopt or adapt multiple agile methods.
SSC's need for a learning system	SSC espouses the importance of learning but has no explicit system in use and often find themselves revisiting problems previously "solved"	As the researcher recognized that SSC would be best served by a system for *Organizational Learning* founded on reflective practice, RALMM provides a theory-driven process to enhance reflection in the use of XP
SSC's commitment and continuity	SSC is committed to finding an appropriate software development method, but not cognizant of the role of reflection or learning in the use of software development methods.	SSC would need to commit to the theory-driven interventions beyond the Dialogical AR Partnership. RALMM, being a daily process which is easy to use and directly tied to XP, should be sustainable.

Of these three real-world problems, the first ("SSC's desire for a method") is SSC's espoused real world problem and the second two ("SSC's need for a learning system" and "SSC's commitment") were identified by the researcher in his interpretation of the dialogical evidence.

One issue which remains is that SSC did not consistently use RALMM and omitted steps in the method when they were pressed for time. It is not clear that more time would have helped in this regard, but the researcher was satisfied with the designed artifact and accepts that RALMM needs more rigorous validation and justification in a new research setting. As is evidenced in Chapter 6, the practitioners' goals for the Dialogical AR Partnership were met as

they did adopt a software development method and were very pleased with the success of the method.

7.4.2 How the Designed Artifact Improves the Practitioners' Expertise

At the onset of the Dialogical AR Partnership and in diagnosis, the researcher found the practitioners to be subject-matter experts with respect to their business; a business which had grown in accounts, receivables and employees from 2005 until the start of the Dialogical AR Partnership in July of 2008. However, the practitioners' knowledge dissemination and management strategy consisted of the developers learning and emulating Daphne's style. This method had worked to a degree but was also frustrating to the Daphne and her developers. In diagnosis, Daphne had indicated a desire to find a team method and team "way" and yet continued to build the company and team in a manner which was reflective of her own values. Therefore, while the practitioners possessed subject-matter expertise and technical expertise, they lacked expertise in software development methods and the means to determine methods appropriate to their organizational size and structure. Additionally, the practitioners lacked a reliable system for managing their client relationships and strategic partnerships. In the earlier Dialogical AR cycles, the practitioners often expressed dissatisfaction with customer communication; clients and customers where frequently the source of changes which would cause their projects to go over time and over budget. Lastly, the practitioners lacked expertise which would formalize their learning and communication processes and allow the practitioners to adapt successfully to change.

Table 55 How the Designed Artifact Improves the Practitioners' Expertise

Practitioners' Areas of Understanding	Baseline understanding (T=July 2008)	Improved understanding (T=February 2009)
SSC's knowledge of Agile methods	Only Fred and Johnny had even heard of agile methods previously. Though SSC was using a form of the waterfall model of the SDLC, SSC could not articulate this.	XP and RALMM have transformed SSC's understanding of the SDLC such the practitioners could not readily recall their old approaches. XP delivers a framework well-suited to SSC's small-shop setting.
SSC's model for efficient communication and learning	SSC's approach to customer communication is to provide the best customer service possible, yet SSC are not able to react to change nor are they able to retain past lessons as they react to change.	XP and RALMM have truly created an agile team which is customer-focused where working software is regularly released. RALMM provides the means for retaining knowledge and learning and XP provides a model for customer interaction which promotes effective customer communication.
SSC's understanding of Reflective Practice	SSC have a top-down and hierarchical system of governance and organizing which emanates from Daphne. As a team, little to no reflective is undertaken regularly.	By design, the use of XP and RALMM promote regular intervals of *Reflective Practice* for *Double-Loop Learning*. The team overtly understands the role of reflection inherent in XP and the role RALMM plays in explicating Schön's *Reflective Practice*.

Over time and subsequent to the Dialogical AR Partnership, the practitioners' use of XP and RALMM had improved their expertise in agile software development methods, *Reflective Practice* and *Organizational Learning*. The practitioners are now aware of the benefits afforded to them in selecting a development methodology suited to small teams and small shops. SSC's software development process now includes an *Organizational Learning* system which captures and reconsiders learning in the promotion and use of *Reflective Practice*. Furthermore, RALMM

has bestowed confidence on the practitioners such that methodological adaptation, deviation and hybridization can be freely considered and pursued as necessary. Thus, RALMM and *Reflective Practice* are at now the heart of SSC's daily structures of designing and learning.

7.4.3 How the Designed Artifact Improves the Researcher's Expertise

The researcher entered into the Dialogical AR Partnership fully aware of the potential benefits XP would likely pose for a small shop such as SSC (Auer et al. 2002; Boehm 2002a; Cao et al. 2008; Highsmith 2002) and the potential for Schön's (1983, 1987) *Reflective Practice* and Argyris and Schön's (1974, 1978, 1996) Theories of action for *Organizational Learning* would hold for the reflective learning component of XP (Cao et al. 2007; Fægri 2008; Hazzan et al. 2003; McAvoy et al. 2007; Nerur et al. 2007; Tomayko et al. 2004). Guidance and antecedent work from Hazzan (2003) and Tomayko and Hazzan (2004) presented an opportunity to use Dialogical AR to design an XP variant which appropriates *Reflective Practice*. Guidance from McAvoy and Tomayko and Hazzan (2004) Butler (2007), Nerur and Balijepally (2007), Cao and Ramesh (2007) and Fægri (2008) suggested that Argyris and Schön's organizational theories could provide a model to explain the reflective learning component of XP from a theoretical perspective. The researcher's expertise was tested in the application of these theories through interventions in the Dialogical AR Partnership and in development of the design science artifact. Additionally, the researcher's learning could be specified in terms of the artifact and in any phenomenon observed which contradicted the predictions of the introduced theories.

While the design and use of RALMM mostly met with success, the duration and regularity of the practitioners' use of RALMM was cause for concern. As the researcher did not

intend to justify RALMM in this research effort, further use and refinement of RALMM was not necessary as the original goals for the Dialogical AR Partnership had been met: the practitioners and researcher each had what they needed just due to RALMMs existence. The practitioners were happy as, in XP and RALMM, they have a method which assists in maintaining the quality assurance, customer relations and productivity that Daphne and the practitioners desired. In RALMM, the researcher now has a design science artifact whose iterative development and testing in the Dialogical AR Partnership provides an occasion to sufficiently test the predictions and effectiveness of the introduced theories.

Limitations to Reflection-in-practice

The principle surprise or disconnect the researcher experienced involved the practitioners' failure to embrace Reflection-in-practice. However, Schön (1983) predicts and explains why an organization's historical and social context might inhibit *Reflection-in-action* (p.263): In cases where *Organizational Learning* system does not challenge existing top-down hierarchical structures of power, then *Reflection-in-action* may not introduce learning and develop new repertoire in the team. As a result, if the team's learning system is inhibited by a predisposition to what Schön (1973) calls *Dynamic Conservatism*, then *Reflection-in-action* is partially or wholly neutered and RALMM's function will not benefit the team.

In the case of SSC, the primacy of Daphne's perspective and influence is a function of the historical and social context of the company in which Daphne has assumed the risk to start her own company and grow that company. As such, Daphne's role as manager and authority figure somewhat influences her ability to engage in *Reflection-in-action* and *Model II* behavior,

or, perhaps inhibits the ability for her developers to engage in *Reflection-in-action* which promotes *Double-Loop Learning*. Schön has anticipated this possibility:

> *The Reflection-in-action of managers is distinctive, in that they operate in an organizational context and deal with organizational phenomena. They draw on repertoires cumulatively developed organizational knowledge, which they transform in the context of some unique situation. And as they function as agents of organizational learning, they contribute to the store of organizational knowledge… But managers function as agents or organizational learning within an organizational learning system, within a system of games and norms which both guide and limit the directions of organizational inquiry… As a consequence, the organizational learning system becomes immune to Reflection-in-action. (Schön 1983:265)*

Therefore, if and how Daphne chooses to reflect-in-action has inordinate influence on how her developers reflect-in-action. In SSC's case, blogging and an open discussion of blogging (both important components of RALMM) in the *Daily Standup Meeting* occurred very infrequently from December 2008 to February 2009. Daphne and her developers espoused to have valued using daily blogging for *Reflection-in-action*, but they were not observed to have blogged enough to substantiate their value of this practice. Further, *Pair Programming*, another essential element of RALMM, was infrequently engaged and only as a matter of convenience.

In many cases, this recalcitrance might have been a simple function of the practitioner's attention to more pressing matters of business. However, feedback from earlier Dialogical AR cycles revealed an extant learning system which emanated nearly solely from Daphne to her employees which was very hierarchical, very top-down. Accordingly, if this old learning system lingered and informed the practitioners' *Theory-in-use*, even as the practitioners adopted and professed praise and confidence in XP and RALMM, then the mechanisms for *Double-Loop Learning* were not truly utilized and RALMM could not function effectively.

The Learning Paradox

Argyris and Schön (1996) also predict and characterize this problem in what they term the *Learning Paradox*. This disconnect arises in what is *discussable* and *not-discussable* in the context of organizational norms. In most cases, a small shop will have an involved owner who, rightfully, insists on maintaining the deciding say on many issues. With this in mind, there is doubt concerning whether the team will engage in true *Reflection-in-action* and *Double-Loop Learning* as some topics are not really up for discussion. If the learning system only focuses on what is discussable, then most interventions will only be applied to discussable problems. This was evident in the Dialogical AR Partnership where SSC did not directly deal with their strategic partnership problems despite repeated failures wrought by poor and uninformed decisions made on the part of the strategic partner. In any case, as the team further bypasses matters not discussable, disincentive to bring these issues into the open arises as it is unpleasant for the practitioners to admit that they have bypassed these issues. Argyris and Schön describe practitioners caught in this situation as being *double bound*: they are unable to address the undiscussable and yet suffer and erode their learning system further when they do not discuss important matters.

Argyris and Schön offer several suggestions to address the learning paradox and each are centered on achieving *Model II* behavior. Each of the suggestions center on achieving awareness of this issue and each requires change and enlightenment in top management. With a small-shop like SSC, it is possible that the inertia of the team lead and company owner is so strong that the developers have little hope of breaking it. In one sense, another round of Dialogical AR would provide a good means to address this topic, but the researcher would have to identify and share this problem with the practitioners. In the case of SSC, the practitioners' immediate problems

were addressed and the Dialogical AR Partnership had terminated. However, SSC did discuss a willingness to start the process in the future as a means for longitudinal review.

A related issue was SSC's failure to realize true *Collective Code Ownership.* As the developers could not or would not regularly engage in *Pair Programming,* it is feasible that *Collective Code Ownership* will remain unachievable. Further, team equality, implicit in Collective Code Ownership, may not always be possible in the case of small-shop software development. Daphne had indicated clear limits to the collectivization allowable at SSC and, considering the social and historical context, this seems rational: Daphne alone has assumed the risk or running her small business and she is reluctant to relinquish control. It may be the case that *Collective Code Ownership* would be more possible if company ownership were equal among the practitioners.

It is also possible to view the learning paradox as a kind of Groupthink where the practitioners make poor decisions, despite contradicting evidence, so as not to disagree with the group (Esser 1998; Whyte 1989). In this light, what is and is not discussable narrows the universe of discourse for the team and stunts the growth possible when using RALMM. While Groupthink is commonly reserved to describe why simple steps are not taken to prevent disastrous situations, such as in the case of the Challenger accident in 1986, in this case, the practitioners at SSC may be so molded to Daphne's patterns and values that *Reflection-in-action* will not produce results that contradict Daphne's thinking. Again, Daphne espouses the value of learning inherent in XP and RALMM, but she did not, through her example and leadership, promote the *Collective Code Ownership* which truly enables RALMM as a learning system.

Argyris and Schön (1996) have encouraged more research into the learning paradox and such research, specific in this case to small-shop software development method, may have implications for RALMM. The learning paradox suggests that RALMM requires further adjustment in light the findings of this research. While the learning paradox is well-covered in various literatures, only one other example has been found which addresses the learning paradox in the use of agile methods in the small-shop and/or small-team setting (McAvoy et al. 2007). This paucity alone suggests there is room for further research in this area. Given the emphasis agile methods place on learning, and given how the identification of the learning paradox has improved this researcher's knowledge and understanding of *Reflective Practice*, further research in this area is warranted and perhaps essential to further development of RALMM.

Argyris and Schön both describe the interplay between *Reflective Practice* and an *Organizational Learning* system. The learning paradox is essentially concerned with taking individual learning processes and using them to promote *Organizational Learning*. Thus, SSC can "…become a learning organization if and only if there is at least a strong integrator to assure the transition from individual learning to team and *Organizational Learning*" (Bratianu 2007:375). RALMM proposes to be a model and method which constitutes this "strong integrator" but RALMM itself is vulnerable to the learning paradox in the case where the team, for various reasons, systematically fails to discuss matters in a candid manner.

The researcher's expertise, with respect to the theoretical perspectives he brought to the Dialogical AR Partnership, was in agile methods, *Reflective Practice* and *Organizational Learning*. These are each wide and vast areas of knowledge of which the researcher possessed greater knowledge in comparison to the practitioners, but certainly not perfect knowledge. Despite Schön and Argyris and Schön's address and outline of challenges to the successful use

of *Reflection-in-action*, it was only within the Dialogical AR Partnership that these threats became apparent to the researcher and, specifically, as relates to the designed artifact.

Table 56 How the Designed Artifact Improves the Researcher's Expertise

Researcher's areas of understanding	Baseline understanding (T=July 2008)	Improved understanding (T=February 2009)
Agile methods and XP	Researcher has an understanding of agile methods and XP from the literature alone.	Having assisted SSC in learning XP, the practitioner has an understanding of agile methods which is grounded in the relevance of practice.
Reflective Practice (Schön 1983, 1987)	Researcher has a robust understanding of *Reflective Practice* and its impacts in various professions. Researcher understands the context of antecedent work in Software Engineering and Agile Methods.	Researcher has a new understanding of *Reflective Practice* grounded in the context of practice. Researcher, in practice, understands the limitations to *Reflection-in-action* and their implications for further development of RALMM.
Theories of action and organizational learning (Argyris and Schön 1974, 1978, 1996)	Researcher has a reasonable understanding of Theories of Action and *Organizational Learning*. Researcher understands the context of antecedent work in Software Engineering and Agile Methods.	Researcher has a new understanding of Theories of Action and *Organizational Learning* grounded in the context of practice. Researcher, in practice, understands the implications of the *Learning Paradox* as it relates to *Reflection-in-action* and the further development of RALMM.

It is remarkable that, as an action research effort designed to develop a design science artifact, this research has been so focused on organizational issues and that the completely successful use of RALMM has also been confounded by the issues related to social and historical context. Hevner et al. (2004) mention that design science research should "control" for organizational issues by focusing on the artifact in and of itself:

> *We conceive of IT artifacts not as independent of people or the organizational and social contexts in which they are used but as interdependent and coequal with them in meeting*

> *business needs. We acknowledge that perceptions and fit with an organization are crucial to the successful development and implementation of an information system. (p.83)*

In the case of an artifact designed to direct *Organizational Learning*, it might be impossible to avoid organizational issues.

Possible Solution Approaches to the Learning Paradox

Oddly, recursive and "meta" approaches to reflection may offer a partial solution to the learning paradox. The "boss" or power-broker in a small shop might come to recognize problems associated with the learning paradox if the shortcomings which arise from the paradox can be demonstrated as deleterious to the "bottom line" of efficiency or profitability. In SSC's case, the practitioners did not warm up to XP, and subsequently RALMM, until their use of both methods yielded demonstrable benefit to productivity. One "meta" solution approach to the learning paradox lies in the spirit of Schön's (1987) *Ladder of Reflection* which, as a concept, was well-received by the SSC practitioners and by Daphne. As the *Ladder of Reflection* involves multiple "meta" levels of reflection, revelations concerning the learning paradox may come to light if the meta-levels of reflection were applied to the problem.

In a case study very similar to this research, McAvoy and Butler (2007) examine a case where a small team (twice the size of SSC and which exists in the context of a larger company) fails to adopt XP. McAvoy and Butler (2007) identify the root cause of XP adoption failure in their study in terms of *Model I* and *Model II* behavior and what they call the *Abilene Paradox* which is essentially the same concept discussed by Argyris and Schön as the *Learning Paradox*. In both paradoxes, problems arise due to "…an inability to manage agreement" (McAvoy et al. 2007:556). SSC, however, is a slightly different case as they are small-shop and small-team. Further, Daphne is the *de facto* power broker whose participation in the use of XP or RALMM

directly affects her team's ability to adopt these methods. From the perspective of either paradox, the developers at SSC are unwilling to question or challenge Daphne and this presents a challenge to effective *Reflection-in-action*.

While McAvoy and Butler (2007) ultimately reach the same conclusion as that reached in this research, that the phenomenon requires further study, their framework utilizing the characteristics of *Model I* and *Model II* behavior in contrast to the practitioners' actions in the field study is useful to understand why SSC was reluctant to adopt practices in RALMM related to *Reflection-in-action*. Table 57 shows which *Model I*/*Waterfall*/*Single-Loop Learning* and *Model II*/*Agile*/*Double-Loop Learning* action strategies were initially observed at SSC in July 2008 (Time 1).

Table 57 Action Strategies Observed in July 2008

Single-Loop Learning	**(Model I-Waterfall)**	**Double-Loop Learning**	**(Model II-Agile)**
Model I Governing Variables	**Action Strategies observed at SSC**	**Model II Governing Variables**	**Action Strategies observed at SSC**
Plan for project scope, cost and schedule	These activities were all done upfront by Daphne	***Valid information***	Information is shared regularly between Daphne and "project managers" and at weekly meetings. Most things are on a "need to know basis."
Maximize winning and minimize losing	-	***Free and informed choice/open debate***	Somewhat observed at weekly meetings.
Minimize negative feelings	Very little reflective discussion observed	***Internal commitment***	Daphne cites this as a weakness she'd like a method to address
Be rational	Daphne unilaterally protects her interests as a primary theory of action		

Table 58 shows which *Model I*/Waterfall/*Single-Loop Learning* and Model II/Agile/*Double-Loop Learning* action strategies were initially observed at SSC in February 2009 (at Time 2).

Table 58 Action Strategies Observed in February 2009

Single-Loop Learning	**(Model I-Waterfall)**	**Double-Loop Learning**	**(Model II-Agile)**
Model I Governing Variables	**Action Strategies observed at SSC**	**Model II Governing Variables**	**Action Strategies observed at SSC**
Plan for project scope, cost and schedule	The initial stages of project negotiation were still conducted in this mode	***Valid information***	Iterative development, emerging requirements and frequent customer feedback were all observed.
Maximize winning and minimize losing	-	***Free and informed choice/open debate***	Collective Code Ownership within the team was not fully evident, but customer coordination was vastly improved.
Minimize negative feelings	Behavior was still observed; it is speculated, as evidenced in the *Learning Paradox* as a cause for the lack of adoption of Reflection-in-practice in RALMM.	***Internal commitment***	Small and frequent releases of working software became norm. Continuous reflective and retrospective testing processes were well under way and showing steady improvement.
Be rational	Daphne continued a de facto top-down style and saw the elements of XP and RALMM as a means to better monitor and control team productivity and learning		

Similar to the "meta" levels of reflection in Schön's *Ladder of Reflection*, McAvoy and Butler (2007) promote *Triple-Loop Learning* (Romme et al. 1999) as a response to the *Learning and Abilene Paradox*. *Triple-Loop Learning* represents a check-and-balance "meta" approach which is characterized as "…learning how to learn" (McAvoy et al. 2007:555). Addressing the issue of Daphne's influence regarding what is *discussable* constitutes a third loop of reflection: a reflection on the reflection learning. There is no doubt that adopting *Triple-Loop Learning* requires sophistication and SSC is not likely to respond to this addition unless the benefits are made clear.

Without further research, it is not certain that any of these possible remedies to the *Learning Paradox* have merit. McAvoy and Butler (2007) offer *Triple-Loop Learning* as a possible solution approach in the case of XP adoption failure in small team situated in a larger organization. In the case of SSC, XP was fundamentally adopted, but aspects of RALMM were not given full consideration. Due to this fact, it is very difficult to make any conclusive statements which could then serve as theoretical propositions. This section has presented the researcher's reflections on the matter by considering a similar case study using a similar theoretical framework and by also considering the ideas informing the *Ladder of Reflection*. Since a simple remedy of "more time" may have revealed regular use of RALMM, issues related to the *Reflection-in-action* in the practitioners' use of RALMM require further study. Opportunities for further study are discussed in the next chapter.

7.5 How the Designed Artifact Addresses the Research Questions

As a result of the Dialogical AR Partnership, the researcher's improved expertise will, by intent and design, be only generalizable to theory rather than populations (Lee et al. 2003). Thus, two generalizations are offered in analysis of the research outcomes. First, *Reflective Practice* has appropriately offered an explanation of how the learning processes in XP are cyclical, reflective and sustainable. Second, *Reflective Practice* suggests the general structure of RALMM, which is meant to model an *Organizational Learning* system in the use of an agile method in a small-shop software development environment. Thus, the answers to each of the research questions are addressed IN the designed artifact.

7.5.1 How the Designed Artifact Addresses Research Question One

Research Question One	"What is an explanation for how the professional practice of small-team software development in the small-shop environment benefits or does not benefit from the introduction of *Reflective Practice*, and what contributions can this explanation make to the body of theory on the professional practice of small-team software development?"

RALMM is the manifestation of a body of theory from Argyris and Schön (1974, 1978, 1996) and Schön (1983, 1987) on *Reflective Practice* as it relates to individual and *Organizational Learning*. What does the epistemology of *Reflective Practice* imply for our theoretical understanding of the professional practice of software development in the small-team setting? Is it possible that agile success, as supported by the body of theory from Schön and Argyris, represents paradigmatic shift? Nerur and Balijepally (2007) would answer in the affirmative and so too does this dissertation. Schön has carefully outlined and positioned

Reflective Practice as an epistemology in diametric and polar opposition to *Technical Rationality* with respect to learning and change. The evidence from the Dialogical AR setting, and the qualities and properties of the designed artifact, demonstrates that *Reflective Practice* provides a theoretical basis for a small shop's use of XP. Thus, if *Reflective Practice* is the case against *Technical Rationality*, and if engineering, as a frame and metaphor for the professional practice of software development is a discipline grounded in *Technical Rationality*, then what conclusions can we draw concerning agile methods? This research, in generalizing to theory, has made the case that agile methods represent a paradigmatic break away from engineering and, as such, are better explained by the epistemology of *Reflective Practice*.

What does this imply for how software professionals are trained? And, what does this imply for the literature on software engineering? If *Reflective Practice* is an appropriate theoretical lens from which agile methods as paradigm-change can be understood, then is it appropriate to refer to agile methods as a software engineering methodology? In 1968, software engineering was a practitioner response to perceived chaos. 30 years, later, can we say that agile was a response to software engineering? The literature and the outcomes of this research support the position that, in *Reflective Practice*, we have an alternative epistemology for agile methods beyond that of the engineering metaphor and *Technical Rationality*. With *Reflective Practice* we have a philosophical and ontological basis to develop the metaphor of agility in its own right as being distinctive from software engineering in general. *Reflective Practice* allows for long since forbidden disciplines and concerns, such as art and craft, as accepted informants to agile methods and equal partners at the table.

Of course there are enlightened scholars and practitioners in software engineering who openly acknowledged the role of art and craft in software development, but what net effect have

these voices had on the profession? Does *Reflective Practice* have an equal setting at the table as does *Technical Rationality*? Are art and craft commonly mentioned in professional training for software developers? Are art and craft commonly mentioned academic journals on the subject of software development, or as an accepted approach from which developers are molded? The theoretical contribution of this work then is to have established, with empirical evidence, the direct applicability of *Reflective Practice* to the use of an agile method and to provide a design science artifact which represents and demonstrates this relationship. This work joins a growing chorus of voices from practitioners and scholars who assert that some forms of software development, especially in light of a global economy and a flattened world, require a new philosophical approach such that the impact of these new methods can be fully understood. Thus, by *naming and framing* agile software development methods as Reflective practice, as opposed to software engineering and *Technical Rationality*, we can understand this phenomenon from a fresh perspective (Schön et al. 1994).

Hopefully, this fresh perspective percolates into the infrastructure of our systems and institutions for training and educating software development professionals as we now have cause to move beyond *Technical Rationality* alone and into a higher and fuller comprehension of the art, craft, science and design of software development.

7.5.2 How the Designed Artifact Addresses Research Question Two

Research Question Two	"What is an explanation for how *Reflective Practice* improves or otherwise changes the use of agile software development methods in a small-team and small-shop software development environment, and what contributions can this explanation make to the body of theory on agile software development methods for small-team and small-shop software development?"

While the implications of *Reflective Practice* for agile methods and XP in particular have been examined by Hazzan (2002 and Tomayko and Hazzan (2003, 2004a, 2004b, 2005), neither have offered an empirical study which demonstrates the plausibility of their recommendations, nor have they offered a design science artifact which demonstrates the effectiveness of *Reflective Practice*. Thus, in this dissertation, RALMM improves XP by adding a theory-based learning system to XP. RALMM, as a manifestation of *Reflective Practice*, demonstrates how *Reflection-in-action* and *Reflection-on-action* "actuate" and explicate the learning systems inherent within XP by expressing them overtly and formally and by casting them in the light of theory. Furthermore, Theories of action – *Espoused Theory* and *Theory-in-use* – present an opportunity to understand and guide *Reflection-in-action* and *Reflection-on-action* to facilitate both *Single-Loop* and *Double-Loop Learning*.

Implications of the research outcomes for the body of theory on agile methods are both clear and unclear. It is clear that theorizing on agile software development methods is relatively new, and that this research adds to extant theoretical explanations of agile success in the literature. Also, in this sense, RALMM and findings thereon are consistent with the emerging notion (Cao et al. 2008; Fægri 2008; McAvoy et al. 2007; McAvoy et al. 2008; Nerur et al. 2007) that the paradigmatic shift evidenced in agile methods is very congruent to and explained by *Reflective Practice* (Schön 1983, 1987) and the Theory of Action perspective on *Organizational Learning* (Argyris and Schön 1974, 1978, 1996).

7.5.3 How the Designed Artifact Addresses Research Question Three

Research Question Three	"What is a design science artifact which provides the benefits of agile software development practices and *Reflective Practice* while satisfying accepted guidance for the development, implementation and evaluation of the design science artifact?"

This chapter makes the connection between *Reflective Practice* and agile software development practices evident and inherent in RALMM. Thus, RALMM, *per se*, is offered in answer to research question three: RALMM captures and conveys all of the benefits of XP for small teams and small shops and offers *Reflective Practice* as a theoretical and practical perspective which guides a small-team and small-shop *Organizational Learning* system. However, as this research question also requests satisfaction of accepted guidance for the development, implementation and evaluation of the design science artifact, this chapter concludes with an assessment of whether RALMM's development using DSAR in a Dialogical AR Partnership has produced valid design science artifact.

7.6 Validating the DSAR Framework as Interpretative Research

Hevner et al. (2004), in recognition of Klein and Myers' (1999) guidance for Interpretive researchers have also put forth a set of guidelines to inform "…the community of IS researchers and practitioners on how to conduct, evaluate, and present design-science research" (Hevner et al. 2004:77). In using Lee's (2007) DSAR framework, the research method used is subject to guidelines for both interpretive and qualitative research, and, via Dialogical AR, subject to guidelines for design science research. While serving these two masters is a challenge, having overlapping sets of guidelines can serve to strengthen the quality of this work, garner recognition for adherence to more than one rigorous research process, and perhaps broaden the appeal of the research to a wider audience. In any case, the risks and weaknesses of taking a "checklist" approach are accepted in favor of the potential benefits of using Hevner et al.'s (2004) criteria.

Hevner et al. (2004) corroborate March and Smith's (1995) Design Science Research Framework in considering that the outputs of design science research are typically constructs, models, methods and instantiations. RALMM is offered as a model and method although RALMM could also be considered at the construct and instantiation level as well. At the construct level, RALMM simply brings together the constructs already apparent in agile, *Reflective Practice* and Theories of action for *Organizational Learning*. As it has been used in practice by the practitioners at SSC, RALMM has also been instantiated.

Hevner et al. (2004) caution the design science researcher against making their research effort the routine "…application of existing knowledge to organizational problems, such as constructing a financial or marketing information system using best practice artifacts (constructs, models, methods, and instantiations) existing in the knowledge base" (p. 81). Rather, the design science researcher must address "…important unsolved problems in unique or innovative ways or solved problems in more effective or efficient ways. The key differentiator between routine design and design research is the clear identification of a contribution to the archival knowledge base of foundations and methodologies" (p.81). Therefore, the goals of action research and design science are so congruent that there is little wonder that an increasing number of researchers are suggesting a combination of design science and action research efforts (Järvinen 2005; Järvinen 2007; Lee 2007).

While bringing *Reflective Practice* to XP is a meaningful contribution in and of itself, the conflagration of an epistemology of *Reflective Practice* and a set of design principles and methods also emphasizing reflection, provides a theoretical basis for a reflection on agile methods which also supports agile methods as paradigm shift. So, an important unresolved and remaining problem is the lack theorizing on agile methods success and, attendant to this

theorizing, a lack of improvements to agile methods which reflect theory. This is not surprising as agile methods are a practitioner revolution; it is up to researchers, perhaps using action research and design science, to move our scholarly understanding of agile phenomenon forward through controlled and scientific inquiry and in the application of theory or the development of theory as relates to success and failure in the use of agile methods.

Grounding an understanding of how *Reflective Practice* benefits and improves an agile software development method is consistent with Hevner et al. (2004): "The fundamental principle of design-science research from which our seven guidelines are derived is that knowledge and understanding of a design problem and its solution are acquired in the building and application of an artifact" (p.82). The remainder of this section contrasts this research effort with Hevner et al.'s (2004) guidelines in order to assess the quality of the artifact and the quality of the research.

7.6.1 Guideline: Design as an Artifact

Many feel that the IT artifact is and should remain at the heart of research in Information Systems (Orlikowski et al. 2001). Indeed, according to this principle, it is imperative that RALMM be formally stated and that RALMM addresses the concepts of construct, model and method. This chapter provides evidence suggesting that RALMM is a model, method and instantiation. However, in presenting RALMM as an artifact of design science research in Information Systems, an important question arises: are software development models and methods also an IT artifact? Moreover, is a model and method of an *Organizational Learning* system an IT artifact? Taken in a reflective and interpretive mode, these questions really beg the

question: Are models and designs developed to guide and control the designers also designs? In a personal conversation in Hevner in 2007, this question was answered in the affirmative: models and methods designed to guide the actions of designers are also designs themselves (Hevner 2007).

It would seem that design science research in information systems may not, despite the nobility in the effort, be able to successfully remove the "people factor" from design research. Perhaps this is appropriate, as "…technology and behavior are not dichotomous in an information system" (Hevner et al. 2004: 77) and are thus inseparable. In the DSAR framework, Lee (2007) provides some redress of this dilemma as Lee positions action itself as an artifact such that designs and designing are the byproducts of action. Consequently, an artifact is not a natural phenomenon that is "out there" waiting to be objectively discovered and analyzed; rather, an artifact is the result of the human process of design. RALMM is also, as an artifact of design, a prescription for how to engender an *Organizational Learning* system in the use of XP. RALMM is also consistent with Hevner et al.'s (2004) assertion that most instantiations are prototypes which demonstrate feasibility. As a prototypical instantiation, RALMM, as designed and developed within the Dialogical AR Partnership and in the specific case of SSC, can only hope to generalize to the theories guiding the design. Moreover, that is to say, RALMM, as a prototypical artifact, has no "population" for generalization. Therefore, it is safe to conclude that RALMM is a prototypical artifact which demonstrates the feasibility of applying *Reflective Practice* to XP.

7.6.2 Guideline: Problem Relevance

It is imperative that RALMM yield utility, but it also must demonstrate relevance to a larger and recognized problem. It would be easy to suggest that the need for RALMM is evident in the emergence of the agile methods alone. However, a better characterization of the problem is that RALMM addresses the epistemological ramifications of the emergence of agile methods. As demonstrated previously in this chapter, there is growing consent that *Reflective Practice* is an appropriate theoretical basis for the use of agile methods in the small-team and small-shop setting. Thus the relevance of the problem addressed has been demonstrated, at the very least, in the literature. Perhaps more significantly, *Reflective Practice* has also demonstrated its viability in addressing failures in the use of agile methods and offers predictive guidance on how to address this failure. Specifically, this research has demonstrated a difference in the goals of XP and the goals of RALMM and has described the current state of SSC developers' use of these methods. In the case of this research, the cooperative design and development of RALMM in the Dialogical AR Partnership allowed for the discovery of new areas of research. This makes the case that design science research can be exploratory in the sense that Grounded Theory is exploratory.

7.6.3 Guideline: Design Evaluation

The approach to artifact evaluation and validation in this research is practically pre-ordained as observational and descriptive given the use of the DSAR framework (Lee 2007) and Dialogical AR. Both design science and action research require that the researcher evaluate the

outcomes research efforts. In Dialogical AR, the researcher and practitioner, in dialog, evaluate which theory-driven interventions worked and did not work; subsequently, the researcher uses the outcomes of these interventions as a test of theory and a test of the design. As this research uses Lee's (2007) DSAR framework, Dialogical AR itself constitutes the evaluation approach for assessing the designed artifact; an evaluation of the Dialogical AR outcomes is the subject of Chapter 6. In this sense, RALMM was designed, developed and studied in an in-depth business environment as recommended by Hevner et al. (2004:86).

7.6.4 Guideline: Research Contributions

This chapter has discussed the contributions of this research effort extensively. The problem addressed did not necessarily imply that current models for learning in the use of agile methods were deficient; rather the problem was based on the fact that researchers in Information Systems and Software Development have not adequately theorized on success and failure in the use of agile methods with respect to learning in the small-team and small-shop setting. Thus, the research effort to realize and instantiate RALMM has provided evidence that *Reflective Practice* and Theories of action for *Organizational Learning* are theoretical lenses most appropriate in order to study and understand agile methods as paradigmatic shift.

7.6.5 Guideline: Research Rigor

Hevner et al. (2004) require that design science research include the rigor of scientific and controlled inquiry in order to distinguish the design effort from the "rudimentary" practice of design. Lee (2007) asserts that the DSAR framework, as "actuated" by the Dialogical AR

Partnership, ensures that both rigor and relevance are "built in." That both rigor and relevance in inherent in the use of action research in Information Systems research is well established in the literature: (Avison et al. 1999; Baskerville 1997; Baskerville 1999; Baskerville et al. 1996; Checkland et al. 1998; Davison et al. 2004; Mårtensson et al. 2004; McKay et al. 2001; Straub et al. 1998). Lee (2007) and Mårtensson and Lee (2004) do caution the researcher to take the *scientific attitude* while conducting action research to ensure that the allowances for rigor built into the structure of action research is best realized. Further, Dialogical AR recognizes the knowledge heterogeneity possible when the scientific attitude of the researcher (rigor) is joined with the natural attitude of everyday life (relevance) of the practitioner (Lee 2007:46; Mårtensson et al. 2004). Thus, a dutiful and strict division of labor in a Dialogical AR Partnership produces the rigor (and relevance) Hevner et al. (2004) indicate is required to demonstrate a clear contribution to one or more scholarly bodies of knowledge in result of the research effort.

7.6.6 Guideline: Design as a Search Process

Hevner et al. (2004) have asserted that design science is inherently iterative. Therefore, in exploring the possibilities in prototypes, design science efforts are less likely to present "finished" products; rather, the design and use of the artifact is likely to lead to further questions. Thus, Hevner et al. (2004) describe design science as a search process which is likely ongoing as each subsequent iteration and instantiation of an artifact shapes new understandings of the problem and problem space. This is not to say that a single artifact cannot produce a satisfactory solution to a problem – RALMM and XP have solved several of SSC's problems – but it does suggest that, as a result of a design science research effort, a designed artifact should also convey

new knowledge to the research community and, ideally, provide new opportunities for research. As research is never "done," so too is it likely that designing and design research is never "done." Thus, Hevner et al. (2004) describe a design/test cycle that produces an artifact for a single research effort and also describes the overall cycle of design research where outcomes lead to new problems which lead to new outcomes and so forth.

Figure 56 Emergence in the Design/Test Cycle (Hevner et al. 2004)

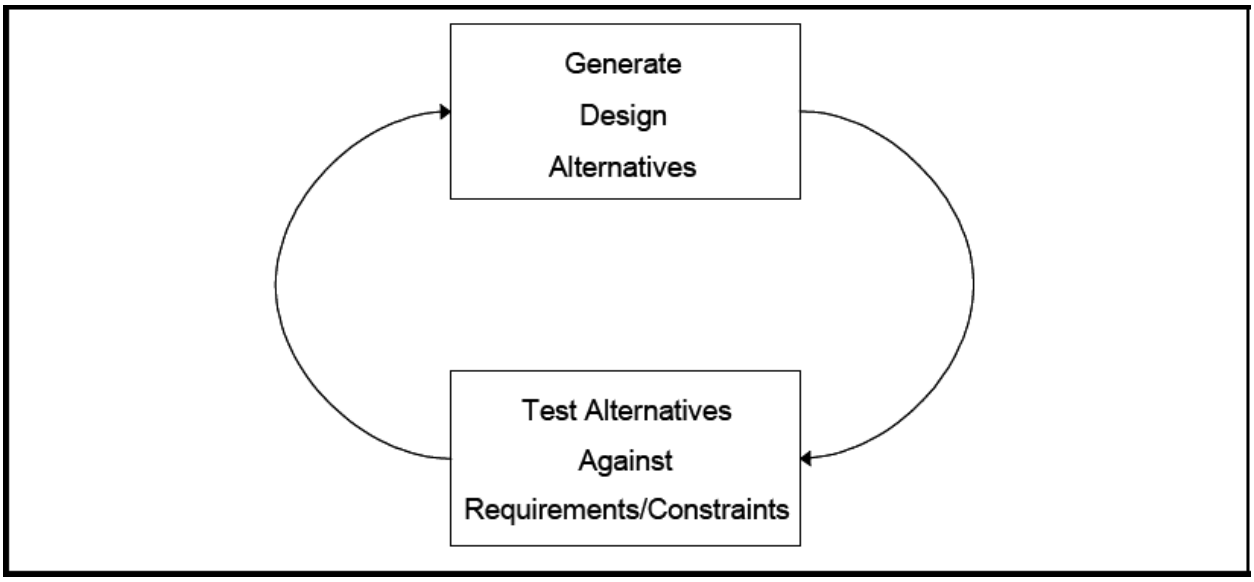


Figure 56 shows Hevner et al.'s (2004) design/test cycle. It is clear that the DSAR framework (Lee 2007) is well-suited to facilitate this cycle as it bears resemblance to the cycles of Canonical AR or the general processes of Dialogical AR. In designing the model, the problem space is illustrated and illuminated within the Dialogical AR Partnership such that the artifact is recognized as a valid solution by the practitioner and in contrast to one or more prevailing bodies of literature. The practitioners' acceptance of the designed artifact, as model, method and instantiation, demonstrates that RALMM's effectiveness and utility. Moreover, successful use of the designed artifact also supports the body of theory introduced by the researcher where the theory explains why RALMM does work when used effectively and consistently or under what

conditions RALMM does not work. Furthermore, the practitioners' inconsistent and at times resistant use of elements of XP and RALMM "…provides a context for additional research aimed at more fully explicating the resultant phenomena" (Hevner et al. 2004:90). Also, the Dialogical AR process does not develop an optimized solution in careful consideration of all known and unknown variables; rather, the Dialogical AR process has created a solution iteratively crafted to the practitioners' problems at hand.

7.6.7 Guideline: Communication of Research

Hevner et al. (2004) remind the researcher that the designed artifact must be clearly conveyed in a manner which could be useful to practitioners, researchers, and managers. As RALMM was created iteratively within the Dialogical AR Partnership, the processes by which RALMM was constructed and evaluated are implicitly embedded in the research approach. This dissertation extensively reports on the steps of and results of the Dialogical AR Partnership in Chapters 5 and 6. This chapter has described the qualities of the designed artifact as the product of the DSAR framework (Lee 2007). Next, the final chapter will conclude by discussing the contributions of the research with respect to practitioners, managers and researchers.

CHAPTER 8 Conclusion

It has been a decade (give or take) since the onset of the agile software development phenomenon and surprisingly little theory has been used to understand the success of agile methods. As small-team and small-shop software development continues to flourish, the need for theorizing on agile success for small teams is warranted and somewhat overdue. This research does not instigate the idea that Schön's epistemology of *Reflective Practice* provides an appropriate theoretical lens from which agile success can be better understood, however, this research moves that agenda forward in providing empirical support for the relationship between agile methods and *Reflective Practice*. The evidence from the Dialogical AR Partnership corroborates the effectiveness of using *Reflective Practice* to understand and guide learning and communication for small-team and small-shop software development. As theoretical constructs, *Reflective Practice* and Theories of action for *Organizational Learning* have produced an effective design science artifact for communication and learning.

While agile methods have been adopted and practiced in a large variety of organizational settings and applied to a variety of problem domains, this research has demonstrated that agile methods are effective in an area of compelling growth and impact: small-team and small-shop software development for the Internet and the World Wide Web. It is far easier to envision small agile teams developing innovative and impactful Web 2.0 (and beyond) applications that it is a large team using heavyweight methods and tightly controlled processes.

An important outcome of this research is not only to have solved the practitioners' problems, but to have improved the researcher's expertise; on this matter, Mårtensson and Lee (2004) offer the following:

> *As in all forms of action research, every intervention that the practitioner makes and that follows from the researcher's theory-based dialogue with the practitioner provides, in the scientific researcher's eyes, an empirical test of the theory. Interventions that yield organizational results that the researcher's theory does not anticipate would provide an opportunity for improving the theory following the researcher's and practitioner's reflections. The theory would be continually improved, following the researcher's reflections, and new interventions would be continually made until the practitioner deems his or her problems to be sufficiently addressed. For the researcher, the product would be a theory that has been improved and that has survived the latest attempts of empirical testing in the field. (p.519)*

To this end, the researcher is also encouraged to develop second-level constructs and interpretations on the occasion that the researcher's theory-driven interventions produce unexpected results. This chapter will focus on the primary unexpected result, which was a difference between the practitioners' *Espoused Theory*, that an agile method and system for team and *Organizational Learning* would be beneficial, and their *Theory-in-use*, whereby the practitioners selectively submitted to XP and RALMM; there are elements of both XP and RALMM which the practitioners tried only partially or not at all. At the conclusion of the Dialogical AR Partnership, the researcher's reflections on the research outcomes increasingly identified this disconnect between perceived benefit and actual use as an encompassing theme for many of the "surprises" encountered where the practitioners' actions did not seem consistent with theory.

This chapter will also discuss the contributions this work has made in the areas or research, method and practice. For clarity, sections 8.1 to 8.4 in this chapter should be considered against the following statements of contribution:

- **Contribution to Practice**: RALMM is a theory-driven learning system which can improve learning and productivity in the use of an agile method in a small-team or small-shop environment
- **Contribution to Method**: RALMM was developed in a Dialogical Action Research partnership utilizing Lee's (2007) Design Science and Action Research framework. As this is the first application of DSAR using Dialogical AR, this research can be used to reflect on the effectiveness of the approach
- **Contribution to Theory**: RALMM, in its iterative development, demonstrates the positive and explanatory effects of Schön's epistemology of Reflective Practice and Argyris and Schön's Theory of Action to better understand and facilitate organizational learning in the use of an agile method in the small-shop environment

To conclude this dissertation, this chapter will first examine what has been learned as a result of the research and discuss the contributions of the research. The specification of learning and contributions of the research will be discussed with respect to the problem solution, improvements to the practitioners' expertise and improvements to the researcher's expertise. The chapter then goes on to discuss the limitations and weaknesses of the research followed by future directions and opportunities for the research program. Last, some final and summary thoughts are offered.

8.1 Specification of Learning

Action research focuses on learning as a principle research output; Mårtensson and Lee (2004:510) describe the Canonical AR cycle as a "learning cycle." Furthermore, in Dialogical AR, learning is not an activity independently and separately engaged by the researcher and practitioners respectively, rather, learning in the Dialogical AR cycle occurs in the interactions between *theoria* and *praxis*:

> *The practitioner does not reflect or learn by himself; instead, it is through a one-on-one dialogue that the researcher purposely encourages and guides the practitioner to reflect and learn (apart from and in addition to her own reflection and learning as a researcher). (Mårtensson et al. 2004:511)*

The emphasis on learning inherent in action research cannot be overstated; the literature is replete with exhortations to ensure bi-lateral learning in action research studies (Baskerville 1997; Baskerville et al. 1998; Checkland et al. 1998; Mårtensson et al. 2004). Argyris and Schön (1996:3) describe learning as a product and process and this research has shown that Dialogical AR fulfills both definitions. Often, the learning process "…occurs whenever errors are detected and corrected, or when a match between intentions and consequences is produced for the first time" (Argyris 1995:20). Thus, when using Dialogical AR, the means by which a researcher demonstrates learning and, by association, identifies the contributions of his research, is by addressing errors and discussing corrective actions.

In specifying learning and implications of the research outcomes, it should be noted that the researcher's overall understanding of the research problem improved and changed during the course of and subsequent to the Dialogical AR Partnership. Antecedent and helpful sources in the literature became clearer to the researcher throughout the research effort and will likely continue past this dissertation. I do not see this as a negative as a dissertation should open up a research program for its writer and not serve as a punctual statement.

8.2 Lessons from the Problem Solution

The most striking lesson in the problem solution, XP and RALMM and the practitioners' use thereon, was the role of the Weblog (blog) and the Wiki Wiki/CMS (wiki). SSC had been using blogs informally at the start of the Dialogical AR Partnership, but had moved over to a

comprehensive CMS, TikiWiki[20], which combined blogs, wikis and many other content and configuration management tools. The idea of using blogs, wikis and content management systems to supplement and augment an agile software development methodology is certainly not new (Chau et al. 2004; Decker et al. 2005; Erickson et al. 2005; Lindstrom et al. 2004; Newkirk 2002). Further, others have suggested the use of blogs and wikis as a learning and knowledge-sharing system in the use of XP and other agile methods (Chun 2004; Pinna et al. 2003; Sharp et al. 2004). Moreover, the use of bogs and wikis to support small-team web development and learning is also mentioned in the literature (Grisham et al. 2005; Maurer et al. 2002). However, the application of these technologies set against the theoretical interventions of *Reflective Practice* and *Organizational Learning*, gives a deeper meaning to the use of these technologies.

Additionally, the "backstory" on the relationship between wikis and XP is worth mentioning. Ward Cunningham has been instrumental in the development of XP, agile methods and Wikis (Leuf et al. 2001). Thus, the relationship between wiki use and XP is not a casual one. The XP method itself was iteratively honed and perfected using wikis in a manner not unlike the iterative development of RALMM in the Dialogical AR Partnership. Kent Beck (1999) frequently consulted with Ward Cunningham as XP took form and utilized Cunningham's Portland Pattern Repository Wiki (Cunningham 2008) as a means of soliciting input and feedback. Furthermore, a set of design principles for Wikis is available at the Portland Pattern Repository Wiki (Cunningham 2008); the historical and social context in the connection between XP and Wikis was not lost on the researcher as RALMM took shape and served to confirm that these technologies were good choices for inclusion in RALMM.

[20] http://info.tikiwiki.org/

Lastly, now that the designed artifact is instantiated and in use in a practitioner environment, we can classify XP + RALMM according to the Hirschheim and Klein (1989) four-paradigm model. On the subjectivist-objectivist continuum, XP + RALMM is decidedly subjectivist as the methods are more conducive to an outlook on reality which "…seeks to understand the basis of human life by delving into the depths of subjective experience of individuals" (Hirschheim et al. 1989:1201). It is difficult to locate XP + RALMM on the order-conflict continuum as XP + RALMM are methods for software development and learning related to both product and process. Thus, XP + RALMM could be located in the "Social Relativist" quadrant of the Hirschheim and Klein (1989) paradigm model for its characteristic as a software development method which assists "…order, stability, integration, consensus and functional coordination" (Hirschheim et al. 1989:1201). On the other hand, XP + RALMM would also be considered *Neohumanist* in its role as a learning method which considers the "…change, conflict, disintegration and coercion…" (Hirschheim et al. 1989:1201) inherent in the problem space. The utility of the Hirschheim and Klein (1989) analysis lies in its address of the paradigmatic aspects of an Information Systems Development methodology. This is congruent to this dissertation's theorizing on an appropriate paradigm for agile methods.

8.3 Improvements to the Practitioners' Expertise

While the obvious improvement to the practitioners' expertise was their adoption and adaptation of XP to facilitate a team method, the most remarkable improvement in the practitioners' expertise was their ability to act reflectively. At the conclusion of the Dialogical AR Partnership, and through the use of RALMM and XP, the practitioners used reflection to improve learning and communication amongst themselves, to improve learning and

communication with respect to their client and partner relationships, and to improve learning and communication used to reinforce their own learning system. Thus, the greatest outcome of the research, with respect to improving the practitioners' expertise, would be *Reflective Practice*. While the practitioners have more progress to make in this area, RALMM remains in place and is available to guide and encourage *Reflection-in-action* and *Reflection-on-action* in the practitioners' daily use of XP.

A remarkable side-effect of the practitioners' newfound reflective practices was their improved ability to hold open and honest reflective conversations. The researcher observed the practitioners use their *Daily Standup Meeting* to address internal issues related to personal and power dynamics which may yield progress in the long term. This new habit may be partially attributable to the frequent reflective dialogs conducted in the Dialogical AR Partnership. As mentioned previously in Chapter 6, the reflective dialogs at times assumed a therapeutic, confessional and autobiographical feel. This phenomenon is sometimes known as "autobiographical narrative" where the confessional and subjective nature of dialog allows the practitioners to tell their story (Burman et al. 1993; Kock 2003; Marshall 2004).

The practitioners' relaxed and confessional tone is presaged in Mårtensson's case study where his dialogical setting was ensconced and removed from the practitioner's daily stimuli (Mårtensson et al. 2004). Further, in the case of SSC, these therapeutic dialogs often involved extensive use of symbolism and analogy: used by the researcher to convey and relate the theoretical interventions and used by the practitioners to relate their circumstances and problems. There is significant power in symbolism as relates to describing and explaining the behaviors of practitioners, clients and partners (Hirschheim et al. 2004). Remarkably, a significant portion of

the transcribed dialog and notes reveal more discussion concerning internal and external interpersonal problems than on issues related to software development methods.

Lastly, there was some discrepancy in the practitioners' *Espoused Theory*, whereby the practitioners professed to have forgotten about their old methods since realizing success with XP, and their *Theory-in-use* whereby the practitioners' selectively adopted the parts of XP that were immediately compelling and required the least change in their habits. Thus, in some cases, the practitioners resisted change by seemingly holding on to their extant methods. Taking an interpretive perspective, it is likely that the practitioners have many valid reasons for their partial adoption and further effort in the Dialogical AR Partnership may have revealed these reasons more clearly. This phenomenon is discussed further in this chapter.

8.4 Improvements to the Researcher's Expertise

Mårtensson and Lee (2004) and Lee (2007) echo Argyris (1995) in their suggestion that learning is rooted in error recognition and corrective responses to error recognition. As the researcher brings theoretical expertise to the Dialogical AR Partnership, the researcher is expected to report on the effects of theoretical interventions in the Dialogical AR Partnership where the consequences of action did not corroborate the theory's predictions. Mårtensson and Lee (2004) suggest three domains for improvement when using Dialogical AR: improvements in the researcher's expertise, improvements in the practitioners' expertise and improvements to the "real-world" problem. Previous sections have discussed improvements for the practitioners' expertise and to their "real-world" problem; this section will discuss various opportunities for learning as a result of improvements in the researcher's expertise.

Again, it should be noted that the researcher's overall understanding of the research improved and changed during the course of and subsequent to the Dialogical AR Partnership. New insights and new sources from the literature (and new literatures) presented themselves throughout the research effort. In some cases, the very theories brought to the research setting were revealed to the researcher in a new light over the course of and subsequent to the Dialogical AR Partnership. Most opportunities for learning were realized in the practitioners' usage and adoption patterns and habits as they were exposed to XP and RALMM. The remaining subsections in this section discuss the implications for the various theories and techniques which were introduced into the Dialogical AR Partnership by the researcher.

8.4.1 Implications for the Body of Knowledge on Agile Methods

Kent Beck (1999) realized that XP would subject software developers adopting the method to quite a bit of change: "If you want to try XP, for goodness sake don't try to swallow it all at once… pick the worst problem in your current process and try solving it the XP way" (p.77). Abrahamsson et al. (2002) also addresses partial XP adoption and question whether teams who partially adopt and adapt XP are still practicing XP. Even the practitioner literature on XP varies in advice for adopting XP where no two adoption plans are the same (Jeffries et al. 2001). This is despite the fact that XP is perhaps the most widely used and documented of the agile methods, with the longest history of use (Abrahamsson et al. 2002; Williams et al. 2000b). Additionally, that SSC, as a small-shop, were able to quickly derive very noticeable benefit from XP, despite their somewhat selective, inconsistent and ad hoc adoption pattern, is also predicted and discussed in the literature (Maurer et al. 2002). Thus, partial method adoption and method adaptation is rather encouraged and anticipated in the agile methods literature. However, the

researcher assumed that adoption and adaptation would be the result of the practitioners' careful and personal consideration of the methods in their own independent investigation; this was not the case.

A remarkable aspect of SSC's adoption and adaptation of XP is the means by which the practitioners at SSC learned the technique. Right up until the last dialog, when the researcher exited the Dialogical AR cycle and thus discontinued the Dialogical AR Partnership, none of the practitioners at SSC professed to have taken a single independent action to "learn" XP on their own. They professed to have read none of the supplemental books and articles given to them throughout the Dialogical AR Partnership; they did not visit any websites or even browse simple entries on the topic on the various free online encyclopedias commonly available.

By the end of the Dialogical AR Partnership, I was somewhat taken aback by this news and also quite puzzled about it. Thus, a principle question arose: "How could practitioners realize so much benefit from a method that they researched very little on their own?" Again, I think this behavior can be attributed to an over-reliance on the guidance of the researcher where the researcher may have fallen into more of a "teaching" mode than Dialogical AR calls for. This problem is further discussed in the next subsection.

8.4.2 Implications for the Body of Knowledge on Action Research

Given the researcher's previous educator–student relationship with the practitioners years prior to the Dialogical AR Partnership, it is possible that the power structures inherent in the educator-student relationship unknowingly and inadvertently biased both the researcher and the practitioners. In reviewing the transcripts of the dialogs, a didactic and pedagogic tone is

discernable from the researcher when new concepts, such as XP and *Reflective Practice*, were introduced in the Dialogical AR Partnership. Thus, as Mårtensson and Lee (2004) encourage that the social and historical context influencing both the researcher and practitioner be carefully considered, several of the discrepancies between theory and practice may have been partially due to the very nature of the Dialogical AR Partnership.

Of course, knowledge of XP and the how the practitioners might adopt and adapt the techniques and processes prescribed by XP were not going to materialize on their own; the researcher, dutifully fulfilling the responsibilities required on his end of the Dialogical AR Partnership, undertook to "teach" XP to the practitioners. However, teaching and lecturing practitioners in a Dialogical AR Partnership is outside the pale of Mårtensson and Lee's description of the research approach when taken literally; indeed it is proscribed to do so:

> *A related point is that, in dialogical AR, the researcher (e.g., Mårtensson) does not attempt to "educate" the practitioner (the managing director), but rather conducts himself as an equal and does his best to enter the world of the natives (the people at Omega Corporation) and to have dialogues situated in how they themselves saw their own world.(Mårtensson et al. 2004:517)*

Therefore, these "pedagogical" activities are far more representative to what Baskerville (1999) and Davison et al. (2004) refer to as Canonical AR, which perhaps focuses more acutely on the participatory aspects of action research. There are times when the researcher had to teach by example and "get down into" the detailed matter concerning the tools and techniques the practitioners would need to learn in order to adopt XP. I do not regret nor do I especially apologize for these deviations, but it is important to note that, in order to facilitate SSC's adoption, and subsequent adaptation of XP, I had to "get my hands dirty" above and beyond "just" dialog.

In any case, I feel that the iterative and reflective nature of the Dialogical AR method itself allowed for the team to adopt and adapt XP with only input from the researcher during the partnership. Thus, this research provides further empirical evidence that Dialogical AR and other forms of AR can produce research results that are relevant. There is no question that Dialogical AR has produced a meaningful and useful artifact of design in the case of SSC.

8.4.3 Implications for the Body of Knowledge on Reflective Practice

Schön (1987) uses the coach and student relationship extensively in order to illustrate how *Reflective Practice* influences and facilitates learning. One of the principle theory-driven interventions used during the research, the *Ladder of Reflection*, is illustrated as a transaction between coach and student. Furthermore, *Pair Programming* is further opportunity to explore the coach/student relationship to increased the effectiveness of XP (Williams et al. 2000a). Tomayko and Hazzan (2004) also focus on the coach/student relationship such that their model demonstrating how *Reflective Practice* may improve *Pair Programming* informed the interventions introduced in this research in. Thus, SSC's reticence in their adoption of *Pair Programming* is puzzling as RALMM relies on *Pair Programming* to realize the benefits of the coach/student or strategic/tactical relationship possible in the pair.

In SSC's case, *Pair Programming* was adopted as a means of learning new skills and developing *Spike Solutions*. While this partial adoption is condoned in the literature on XP, the practitioners try to use *Pair Programming* to develop all production code – they rejected this out of hand. While there were several legitimate reasons for this, one interesting reason may be attributable to the fact that the researcher did not observe Daphne engage in Pair Programming

with her employees. In diagnosis, among Daphne's concerns was that her employees develop knowledge and skills equal to or surpassing her own. Thus, when Daphne and her developers were first introduced to *Pair Programming*, Daphne indicated that she had "done that" numerous times when she sat down to teach her employees some technique or approach. However, XP and RALMM suggest that *Pair Programming* is useful beyond simple knowledge transfer; *Pair Programming* is an opportunity to develop shared context and understanding within the team.

SSC may benefit most effectively from their use of XP and RALMM if and when Daphne is willing to pair with her developers on a regular basis. The researcher observed Fred and Johnny used the *Ladder of Reflection* while *Pair Programming* on several occasions. However, as the power-broker in her organization, Daphne's example would have better promoted *Pair Programming* in the team. Daphne's desire for her employees to transfer her knowledge to her employees would be better realized if she were to engage her employees in the coach-student role more frequently using *Pair Programming* and the *Ladder of Reflection*.

Excepting obvious reasons, such as lack of time, another possible explanation SSC's reticence would be the conundrum whereby Daphne functions both as a peer and the boss in the agile team. Given her desire to teach her people her own skills and knowledge, I did not expect that Daphne would shun *Pair Programming*. *Reflective Practice* has ample room for learning in the coach-student relationship and using this relationship can help Daphne achieve her goals. In this case, it seems like RALMM may need modification in order address these issues. RALMM needs an additional mechanism where the consequences the reluctance to use *Pair Programming* for boss to employee (coach to student) learning would be more evident in practice.

8.4.4 Implications for the Body of Knowledge on Small Teams

The role of power, and interpersonal dynamics related to power, in Information Systems Development is a well-understood topic (Markus 2002). At SSC, the researcher observed a hierarchical structure where Daphne, as the boss, was also the *de facto* development and team lead. This presents complications affecting the degree to which the practitioners develop a sustainable team culture. Gorla et al. (2004) anticipated this problem to a degree: "…a small team needs personality heterogeneity between a team leader and other team members" (p. 34). Daphne clearly displays the qualities of a strong leader but her strong leadership might prevent rather than encourage open reflection in the team. Gorla et al. (2004) recommend Myers-Briggs (MTBI) or some other personality assessment which will improve understanding in the team such that each personality type is understood. At several points along the way during the Dialogical AR Partnership, Daphne openly reflected on the qualities and traits inherent in her employees and considered her perceptions of these traits as she assigned work.

Again, in light of these discrepancies, it may be the case that RALMM needs recalibration to accommodate an additional level of learning which the practitioners can use to monitor the team's learning process itself. I have no doubt that if Daphne were more aware of the effects and implications of her authority as it relates to the team's ability to engage in reflection, she would be willing to adjust her actions to avoid impeding her team's learning system. That is, these discrepancies should not imply that Daphne did not have the best interests of her company and her employees at heart.

8.4.5 Implications for the Body of Knowledge on Organizational and Team Learning

Schön (1973) characterizes resistance to change as *Dynamic Conservatism*, where actors in the system resist change by clinging to their old habits and patterns regardless of their *Espoused Theory* (i.e. "a learning system is good," "*Reflective Practice* helps us," etc.). In SSC's case, the inertia of Daphne's existing way of doing things presents a strong headwind against change. Yes, XP and RALMM were successful, but the researcher did not observe fundamental changes which suggest that all of SSC's pre-*Reflective Practice* habits had been examined and reconsidered. Thus, despite the benefits of XP and RALMM, Daphne continued to fundamentally believe and trust in the stable state of her extant processes, those which she had created for herself and her company through years of dedicated hard work. Thus, when the new habits and procedures in XP and RALMM were introduced, subtle and tacit responses to these new habits were used in an attempt to retain SSC's former stable state (pre *Reflective Practice*). In this sense, SSC's stable state is predicated on beliefs related to the structure of their organization, the tacit theories guiding their actions and the technologies and techniques which they had become accustomed to (Schön 1973). The irony in Daphne's *Espoused Theory* (we need a method) lies in the fact that Daphne desired a new method less for the purposes of change but more as a means to confirm that her previous practices were sound (her *Theory-in-use*). Throughout the Dialogical AR Partnership, Daphne asserted that her methods and processes were already correct and that she simply wanted to confirm this to herself and others. In this, we can see that agile methods and *Reflective Practice* pose changes to SSC's stable state that will take time to "sink in."

The ironic disconnect between Daphne desiring a new method and then rejecting and/or ignoring some of the fundamental lessons inherent in her new method has been anticipated in the literature. Some have suggested that *Triple-Loop Learning* or *Deutero-learning* can help the learning organization recognize and ameliorate the effects of dynamic conservatism, the learning paradox and the *Abilene Paradox* (Argyris 1995; Argyris 2003; Argyris et al. 1996; Romme et al. 1999; Visser 2007).

> *A learning system… must be one in which dynamic conservatism operates at such a level and in such a way as to permit change of state without intolerable threat to the essential functions the system fulfils for the self. Our systems need to maintain their identity, and their ability to support the self-identity of those who belong to them, but they must at the same time be capable of transforming themselves. (Schön 1973: 57)*

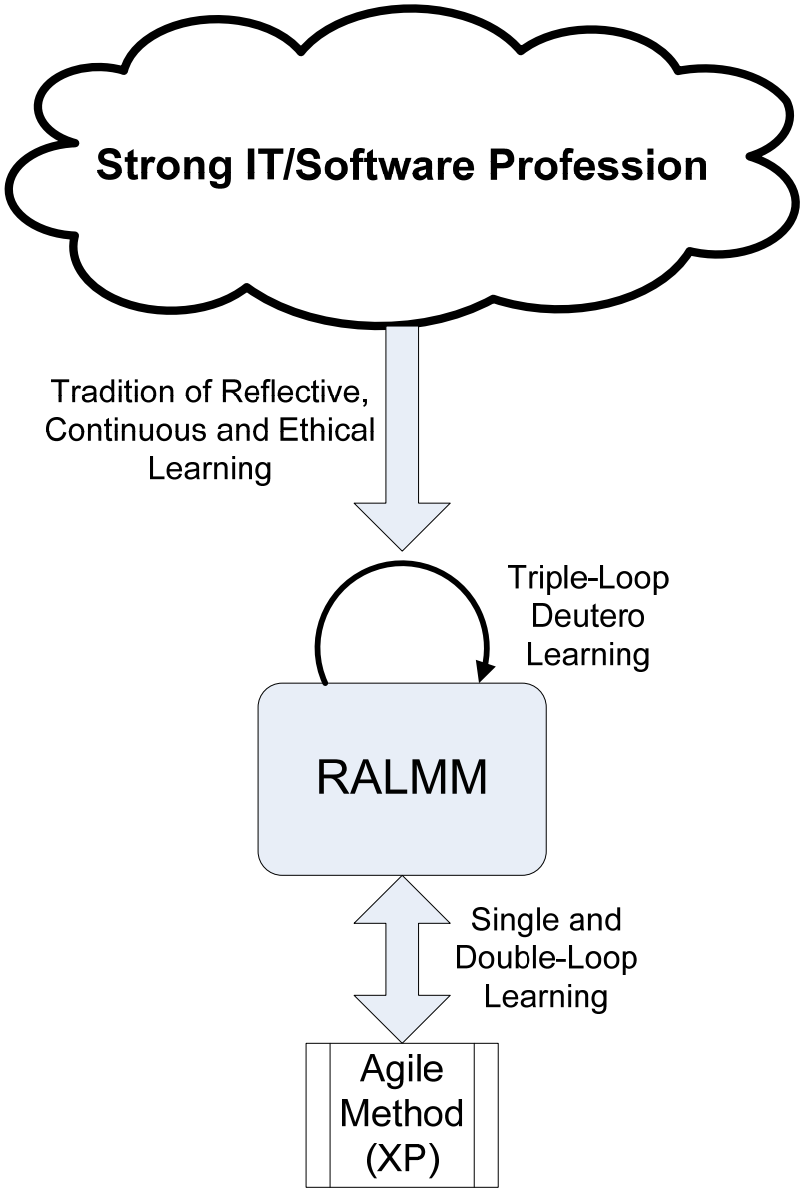
Based on this *back-talk* from the Dialogical AR Partnership, it is now clear that RALMM is incomplete and a work-in-progress as RALMM must adjust to accommodate these problems. This is so as RALMM needs to provide a mechanism for reflection on the adequacy and effectiveness of RALMM itself; RALMM, too, should be dynamically adjustable. Just as the *Ladder of Reflection* contains meta-levels of reflection; the learning organization needs meta-levels for reflective learning. Argyris and Schön propose that *Deutero-learning* (1996:28), in its capacity to enable *second-order* learning, or learning about the process of learning, would be appropriate to this task.

One approach to address the need for *Deutero-learning* in RALMM is to focus on professional practice in small-shop software development: whither the profession of IT? Debate concerning whether IT (taken in a systemic sense to include software, hardware and organizational deployment) is supported by a valid profession is well-known and ongoing (Denning 2001a; Denning 2001b; Denning et al. 1989; Knight et al. 2002; McConnell et al. 1999). One premise of this dissertation has been that small teams in the small-shop setting

require additional professional guidance for learning as they lack the reinforcing effects of a larger organization. However, does a doctor or lawyer working in their equivalent of a "small shop" suffer from their relative isolation? Generally not. It seems as though these professions have such a strong tradition of ongoing, ethical and reflective learning that these professionals can comfortably operate in relative isolation. IT professionals, and by association, agile/small-shop software development professionals lack strong professional guidance which has some effect on their learning systems.

Therefore, one propositional recommendation from this work is the necessity of a stronger professional "guild" for software developers, especially so for agile software developers in small teams. If an IT professional's education included formal and doctrinal guidance concerning continuous, reflective and ethical learning, the practitioners at SSC might have better recognized the necessity of taking occasional *second-order* readings of their own learning process. Pursuant to this, the researcher arranged that the last practitioner-researcher dialog coincide with a "retreat" for the practitioners. This retreat took place outside of SSC's offices and included a general reflective retrospective for the purposes of *Deutero-learning*. Figure 57 demonstrates a proposed relationship between professional guidance on reflective learning, *Deutero-learning* and RALMM.

Figure 57 Model for the application of Deutero-learning in RALMM



Making this connection so late in the research process was somewhat disconcerting: how could I have missed such an important aspect of the theoretical knowledge I was supposedly an expert of? Argyris acknowledges *Deutero-learning* but seems to subordinate to *Single-Loop* and *Double-Loop Learning*, which may have encouraged the researcher to miss its importance:

> *Sometimes Double-Loop Learning is equated with Bateson's 'Deuterolearning'. Schön and I have expressed our intellectual indebtedness to Bateson (Argyris and Schön 1974).*

> *However, we also made a distinction between Double-Loop Learning and Deutero-learning. We understood Deutero-learning to mean second-order learning, reflecting on the first-order actions. Deutero-learning can occur by going meta on single or Double-Loop Learning. The distinction is important because the knowledge and skills required to produce Double-Loop Learning are significantly greater and more complicated than those required for Deutero-learning on Single-Loop issues. (Argyris 2003:1178)*

It should be noted that the late "discovery" of *Triple-Loop* and *Deutero-learning* was made possible by the Dialogical AR Partnership, leading the researcher to reconsider both *Dynamic Conservatism*, and *Deutero-learning*. I didn't expect these things until I encountered them – thus new propositions or new theoretical perspectives from the literature were needed in contrast to the emergent particulars of the case.

8.5 Limitations and Weaknesses

No research effort is without compromises, limitations and weaknesses. In order for research to be useful and meaningful, an open critique on the limitations of a study can bring about opportunities for future work and also encourage rigorous reflection on the assumptions and conclusions of the study. Given that the long-term ambitions of this research were high and given the emergent nature of the results and of the research method, a discussion on the limitations and weaknesses of this research will, if anything, provide a basis for further research.

Among the most obvious apparent weaknesses and limitations of the research is the issue of the generalizability of the results. While some might question the validity and generalizability given the single-site and longitudinal nature of this Dialogical AR study, however, critiques of this nature may be misplaced when applied to action research. In any case, the chosen research design will certainly raise generalizability questions for some readers. While this issue was addressed in the literature review, any conclusions targeted to the population of small teams in

the small web-development shop, small teams in general and software development in general would be tenuous. At best, this work could be generalized to the theories of Argyris and Schön, the epistemology of *Reflective Practice* and the general body of emergent theorizing on the philosophical basis for agile success in small teams. Also, with respect to the theories applied, there are many *Organizational Learning* designs and theories which were not considered as alternatives in this research.

In terms of the research approach, this study was conducted using a variant of action research which has not been widely used. Furthermore, this study may be the only instantiation of Lee's (2007) DSAR framework. Thus, while neither of the methodological guides used in this research have been validated by extensive use, they are based upon sound logic, theory and science. While the mixed-method approach provided an opportunity for rich and qualitative evidence supporting the designed artifact, using both design science and action research demanded that the researcher serve despite their complementary nature (Lee 2007).

Although the designed artifact is a valid contribution to XP and provides substantial support for the role which *Reflective Practice* can play in the use of agile methods, on its own merits, the Reflective-Agile Learning Model and Method is not a significant technological contribution. The technologies used to describe the learning method are quite common and their inclusion could be construed as trivial. The intent of including blogs and wikis was not to suggest a radical innovation of technology, but to actuate or "operationalize" the use of *Reflective Practice* to facilitate the organizational and team learning implicit in the literature on XP. Further, RALMM's utility in the case of SSC does not imply that RALMM would be useful or successful in other teams or in the use of other agile methods. However, RALMM does confirm the theories used to drive the interventions leading to the iterative development of

RALMM. Clearly more research is needed to: carry the model to new settings; add *Deutero-learning* to the model; and to justify the model.

Among its weaknesses, this research is perhaps too ambitious, theorizing towards a paradigm for agile methods in small teams, and, accordingly, scoped too broadly. While the research questions and the use of the DSAR framework ensured that the outcomes of the case and the designed artifact would produce meaningful, relevant outcomes, the broader program implied in discussion throughout this dissertation is just that: broad. Furthermore, while the evidence from the Dialogical AR Partnership and the designed artifact has addressed the research questions, the audience for the results of this study may not be sufficiently defined. Practitioners may find some use in the designed artifact, but the utility of the artifact is heavily tied to the theoretical propositions embedded in the development of the artifact. This artifact is not likely to produce any patents nor would it be something which can potentially be packaged, licensed or sold. In this sense, the designed artifact is, in the parlance of XP, an *Architectural Spike* designed to test an idea and meant to be prototypical at best. Many agile teams use blogs and wikis and other research has suggested the use of the *Ladder of Reflection* and of *Reflective Practice* in general.

Perhaps the most important weakness and limitation of this research is its attempt to provide a broad nexus of ideas which likely represent a lifetime of research in their own right. To distill the work of Schutz, Schön and Argyris in a truly meaningful way constitutes a lifetime of scholarship in its own right. Similarly it is somewhat impossible to completely digest all of agile methods for the purposes of theory. Thus, this research cannot claim to have validated this body of theory and has presented a limited, yet valid, falsification test of the introduced theory. The subject of and theorizing on software development methods in general is its own vast

horizon of work and opportunity in terms of developing, testing and applying theory. These weaknesses extend this study's use of action research and its appeal to the emergent science of design.

Lastly, a very important weakness and limitation of this research has close parallels with weaknesses related to agile methods and small teams: the skill of those using the method. Given that this report is a dissertation written by a doctoral candidate, the outcomes of the research may have been quite different at the hands of a seasoned and experienced qualitative researcher. Just as Cockburn (2000) reminds those considering agile methods that an above-average skill level is preferred, similar exhortations have been made with respect to qualitative and interpretive methods.

> *An individual's level of skill, knowledge and experience in using qualitative research methods is a significant influence in deciding whether or not to employ them in IS research. … A researcher's skill in using qualitative methods is an influencing factor not only during the dissertation but also through her or his career. (Trauth 2001:8)*

A possible solution approach to this dilemma lies in one of the ultimate outcomes and conclusions of this research: to use reflection and adopt *Deutero-learning* (Lee 2001). While a qualitative research approach such as Dialogical AR is risky, research, in itself, is a risky endeavor with no guarantee of reward and qualitative research more so (Applegate et al. 1999; Baskerville 2001). Lee (2001) has also addressed the cart-and-horse dilemma of skill and qualitative research by challenging qualitative researchers to "practice what they preach" by taking an interpretive read of their own research activities. Thus, while it was a weakness, threat and limitation to have used a qualitative approach to the research topic in this dissertation, the aims of the research were worthy and the evidence from the case study is compelling.

8.6 Directions for Future Research

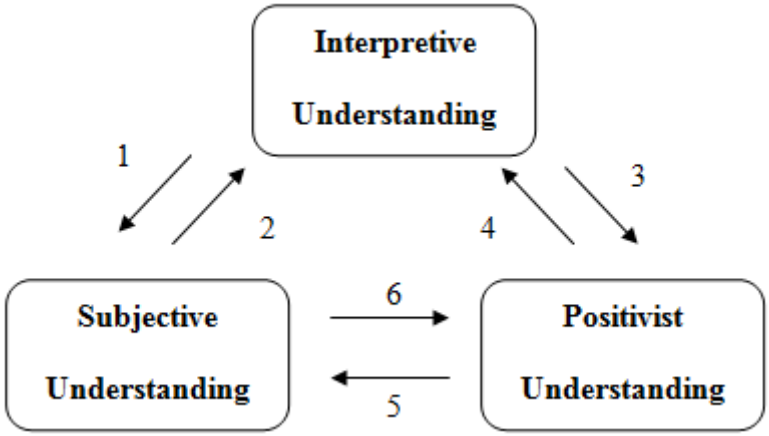
Many of the weaknesses and limitations of the previous section provide an opening for future directions and opportunities. Thus, many of the findings, conclusions and contributions of this study are simply openings for further inquiry. This discussion has clearly shown that the designed artifact requires more refinement and a more rigorous validation approach. The artifact could also be expressed more formally such that others could also validate the utility of the artifact. For instance, the constructs could be more clearly defined as independent variables such that hypothesized dependent variables could be proposed and the relationships between these variables tested and specified in one or more statistical models as is typically possible in the use of structural equation modeling using LISREL and/or PLS path modeling. Optionally, an additional in-depth, single-site study of another small team using another agile method may provide a clean slate such that the artifact and the concepts supporting the artifact can be refined. An important stipulation for any future single-site study would entail the requirement that the team are already regular users of an agile method and already understand the processes of learning and reflection inherent to that method. In the short term, an opportunity for embarking on such a study exists. The researcher has access to a willing small team using Scrum in the setting of a medium-sized software company. Additionally, there are many opportunities to test the viability of the designed artifact with small teams operating within larger companies.

The use of the ladder of reflection in this study and in the designed artifact is one among many possible uses of Schön's ideas from his 1983 and 1987 work. As a theoretical basis for the learning processes embedded in agile methods, a more extensive investigation and appropriation

of Schön's work would improve the designed artifact and provide a more generalized interface to agile methods beyond the specifics of XP.

Lee (1991) provides insight to guide efforts to move RALMM forward as a research program which continues to explore agile methods from the Reflective paradigm for learning. While an interpretive mode of inquiry may be more appropriate for research which is exploratory in nature (searching for theory), an interpretive mode can complement other modes of inquiry. Lee (1991) is careful to remind researchers, from both a *Positivist* and *Interpretative* paradigm, that methods from each tradition can be used for mutual enhancement without sacrificing rigor. As is the case with Dialogical AR, Lee presents an integrated framework for *Positivist* and *Interpretive* approaches to research which is grounded in Schutz' phenomenology (Lee 1991: 351). Lee extends Schutz' first-level and second-level constructs by adding a third level which we can liken unto the theorizing and justifying stages of the DSAR framework. In order to understand, test, and perhaps generalize, the assumptions and interpretations made in developing second-level constructs, the researcher will theorize and justify the second-level constructs. For simplicity, Lee (1991) calls these three levels the subjective understanding (level 1), the interpretive understanding (level 2) and the positivist understanding (level 3). Figure 58 demonstrates Lee's integrated framework for positivist and interpretive approaches to research.

Figure 58 Lee's Integrated Interpretive and Positivist Framework (1991)



Lee's (1991) integrated framework recognizes the benefits of interpretive approaches where: a researcher first observes the practitioner's everyday sense-making and meanings (arrow 1 in Figure 58); and, having developed a second-level interpretive understanding, the researcher may test the validity of his second-level understanding against the subjective understanding (arrow 2 in Figure 58). From the interpretive understanding, the beginnings of the positivist understanding are developed. First, a valid interpretive understanding provides the independent and dependent variables for statistical inference testing in lieu of controlled experiment (arrow 3 in Figure 58). Before experimentation (usually with a quasi-experimental design), a researcher would want to test the validity of the interpretive understanding within a positivist model - again, this is done with usually independent variables, dependent variables and a statistical model of their relationships (arrow 4 in Figure 58). The purpose of steps 3 and 4 are to develop the positivist understanding based on an interpretive understanding. Once the researcher is convinced, through scientific experiment, that the positivist understanding sufficiently and accurately incorporates the subjective understanding, then the positivist understanding can be tested against the subjective understanding (arrows 5 and 6 in Figure 58).

Lee's integrated framework is instructive in this dissertation as it proposes an approach by which various research traditions might be integrated. Whereas Lee's (1991) framework recognizes the contextuality of knowledge, it may not fully appreciate the heterogeneity of knowledge as stated in Dialogical AR. The idea of the positivist understanding, of course, in keeping with the philosophy and epistemology of *Positivism*, is that the rules of formal and hypothetico-deductive logic are beyond the everyday understanding and attitude of the subject practitioner. However, Lee demonstrates the value of the subjective understanding and how this understanding serves as the basis for action-planning/theorizing and future research. Thus, this dissertation recognizes the value of Lee's integrated framework insofar as the Positivist understanding has utility in providing confirmatory tests for the interpretive understanding. As a design science artifact, RALMM has not been justified; future research is required to justify RALMM and is an obvious next step.

8.7 Final Reflections

The outcomes of this research have convinced the researcher that Schön's epistemology of *Reflective Practice* and Argyris and Schön's theories of action for *Organizational Learning*, taken together, are an appropriate theoretical lens from which the emergent Reflective-Agile paradigm can be understood. The research experience also confirmed that Dialogical AR was the correct research method given the goals of the research and the research questions. As a qualitative and iterative research method, Dialogical AR enabled incremental and emergent learning for both parties in the Dialogical AR Partnership. The researcher was somewhat surprised in just how effective this process was as his own understanding of introduced theories evolved during the course of the research. The literatures consulted one year prior assumed new

meanings subsequent to the Dialogical AR Partnership. Thus, the reflective and interpretive mode of inquiry made possible by Dialogical AR "opened doors" for learning in practice and in theory. For instance, the importance of *Deutero-learning* was a late-breaking revelation and was a concept whose importance was not evident until the context of the research had evolved. However, as a solution approach to emerging issues, how *Deutero-learning* and *Triple-Loop Learning* confirmed that the body of theory selected for this research was appropriate.

Reflection was a key enabler for understanding, learning and designing in this dissertation. Throughout the study, it was more apparent that there are objective and subjective uses of reflection and each use pattern played a role in the research. Work towards a problem solution caused objective reflection in which the Dialogical AR partners were focused on tools, technologies and techniques to solve "real-world" problems. Much of this reflection was *Reflection-in-action* and *Knowing-in-action*. However, subjective reflection was there during dialogs and present when the researcher transcribed and reviewed field notes: subjective reflection encouraged those involved to examine how we were learning and growing. Furthermore, the use of reflection in this research has opened up more questions than answers. Perhaps that is the point of research.

Literature Cited

Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J. "Agile software development methods: review and analysis," VTT Publications.

Abrahamsson, P., Warsta, J., Siponen, M.T., and Ronkainen, J. "New Directions on Agile Methods: A Comparative Analysis," in: *Proceedings of the 25th International Conference on Software Engineering*, Portland, Oregon, 2003, pp. 244-254.

Agarwal, R., and Jr., H.C.L. "The Information Systems Identity Crisis: Focusing on High-Visibility and High-Impact Research," *MIS Quarterly* (29:3), September 2005, pp 381-398.

Ahern, D., Clouse, A., and Turner, R. *CMMI Distilled: A Practical Introduction to Integrated Process Development* Addison-Wesley, Boston, 2001.

Aiken, J. "Technical and Human Perspectives on Pair Programming," *ACM SIGSOFT Software Engineering Notes* (29:5) 2004.

Anderson, W.L., and Crocca, W.T. "Engineering practice and codevelopment of product prototypes," *Communications of the ACM* (36:6), June 1993, pp 49-56.

Angus, D.L. *Professionalism and the Public Good: A Brief History of Teacher Certification* Thomas B. Fordham Foundation, Washington, D.C., 2001.

Ante, S.E. "Shawn Fanning," in: *BusinessWeek*, 2000.

Applegate, L.M. "Rigor and Relevance in MIS Research Introduction," *MIS Quarterly* (23:1), March 1999, pp 1-2.

Applegate, L.M., and King, J.L. "Rigor and Relevance: Careers on the Line," *MIS Quarterly* (23:1), March 1999, pp 17-18.

Argyris, C. "Action science and organizational learning," *Journal of Managerial Psychology* (10:6) 1995, pp 20-27.

Argyris, C. "A Life Full of Learning," *Organization Studies* (24:7) 2003, pp 1178-1192.

Argyris, C., and Schon, D. *Organizational Learning: A theory of action perspective* Addison Wesley, Reading, MA, 1978.

Argyris, C., and Schon, D. *Organizational learning II: Theory, method, and practice* Addison-Wesley, Reading, MA, 1996.

Argyris, C., and Schön, D. *Theory in Practice: Increasing Professional Awareness* Jossey-Bass, San Francisco, 1974, p. 224.

Atwood, M.E., McCain, K.W., and Williams, J.C. "How does the design community think about design?," in: *Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques*, B. Verplank and A. Sutcliffe (eds.), Association for Computing Machinery, London, England, 2002, pp. 125-132.

Auer, K., and Miller, R. *Extreme Programming Applied: Playing to Win* Addison-Wesley, Boston, 2002.

Avison, D., Baskerville, R., and Myers, M. "Controlling action research projects," *Information Technology & People* (14:1) 2001, p 28.

Avison, D., Lau, F., Myers, M., and Nielsen, P.A. "Action Research," *Communications of the ACM* (42:1), January 1999, pp 94-97.

Bagert, D.J. "Viewpoint: taking the lead in licensing software engineers," *Communications of the ACM* (42:4), April 1999, pp 27-29.

Barley, S.R. "Technology as an Occasion for Structuring: Evidence from Observations of CT Scanners and the Social Order of Radiology Departments," *Administrative Science Quarterly* (31:1), March 1986, pp 78-108.

Barrett, F.J., and Cooperrider, D.L. "Generative Metaphor Intervention: A New Approach for Working with Systems Divided by Conflict and Caught in Defensive Perception," in: *Appreciative Inquiry: An Emerging Direction for Organization Development*, D.L. Cooperrider, J. Peter F. Sorensen, T.F. Yaeger and D. Whitney (eds.), Stipes Publishing L.L.C., Champaign IL, 2001.

Baskerville, R. "Distinguishing Action Research From Participative Case Studies," *Journal of Systems and Information Technology* (1:1) 1997, pp 25-45.

Baskerville, R. "Conducting Action Research: High Risk and High Reward in Theory and Practice," in: *Qualitative Research in IS: Issues and Trends*, E.M. Trauth (ed.), Idea Group Publishing, Hershey, PA, 2001, pp. 192-217.

Baskerville, R., and Myers, M.D. "Special Issue on Action Research in Information Systems: Making IS Research Relevant to Practice - Foreword," *MIS Quarterly* (28:3), September 2004, pp 329-335.

Baskerville, R., and Wood-Harper, T. "Diversity in Information Systems Action Research Methods," *European Journal of Information Systems* (7:2) 1998, pp 90-107.

Baskerville, R.L. "Investigating Information Systems with Action Research," *Communications of the Association for Information Systems* (2:1999), October 1999, p 32.

Baskerville, R.L., and Wood-Harper, A.T. "A critical perspective on action research as a method for information systems research," *Journal of Information Technology* (11) 1996, pp 235-246.

Bass, L., Clements, P., and Kazman, R. *Software Architecture in Practice, Second Edition* Addison-Wesley, Boston, 2003, p. 560.

Battelle, J. "The Birth of Google," in: *Wired Magazine*, 2005.

Baugh, S.G., and Roberts, R.M. "Professional and Organizational Commitment Among Engineers: Conflicting or Complementing?," *IEEE Transactions on Engineering Management* (41:2), May 1994, pp 108-114.

Beck, K. *Extreme Programming Explained* Addison-Wesley, Reading, MA, 1999.

Benbasat, I., and Zmud, R.W. "Empirical Research in Information Systems: The Practice of Relevance," *MIS Quarterly* (23:1), March 1999, pp 3-16.

Benbasat, I., and Zmud, R.W. "The Identity Crisis with the IS Discipline: Defining and Communicating the Discipline's Core Properties," *MIS Quarterly* (27:2), June 2003, pp 183-194.

Bentley, T.J. "Systems Development Life Cycle," *Management Accounting* (68:11), December 1990, p 13.

Berger, P.L., and Luckmann, T. *The Social Construction of Reality* Anchor Books, New York, 1967, p. 219.

Blackburn, J.D., Scudder, G.D., and Wassenhove, L.N.V. "Improving Speed and Productivity of Software Development: A Global Survey of Software Developers," *IEEE Transactions on Software Engineering* (22:12), December 1996, pp 875-885.

Blasko, V.J., and Mokwa, M.P. "Creativity in Advertising: A Janusian Perspective," *Journal of Advertising* (15:4) 1986, pp 43-50.

Blevis, E., Lim, Y.-K., and Stolterman, E. "Regarding Software as a Material of Design," in: *WonderGround - 2006 Design Research Society International Conference in Lisbon*, Design Research Society, Lisbon, Portugal, 2006, p. 18.

Blum, B.I. "A taxonomy of software development methods," *Communications of the ACM* (37:11), November 1994, pp 82-94.

Boehm, B. "Get Ready for Agile Methods, With Care," in: *IEEE Computer*, 2002a.

Boehm, B. "Software engineering is a value-based contact sport," *IEEE Software* (19:5), September/October 2002b, pp 95-95.

Boehm, B. "Value-Based Software Engineering," *ACM SIGSOFT Software Engineering Notes* (28:2), March 2003, pp 1-12.

Boehm, B. "A View of 20th and 21st Century Software Engineering," in: *International Conference on Software Engineering*, Shanghai, China, 2006, p. 29.

Boehm, B.W. "Software Engineering," *IEEE Transactions on Computing* (25:12), December 1976, pp 1226-1241.

Boehm, B.W. "Software Engineering - As it is," in: *International Conference on Software Engineering*, Association for Computer Machinery, Munich, Germany, 1979.

Boehm, B.W. "A Spiral Model of Software Development and Enhancement," *IEEE Computer* (21:5), May 1988, pp 61-72.

Boehm, B.W. "Anchoring the software process," *IEEE Software* (13:4), July 1996, pp 73-82.

Boehm, B.W., and Papaccio, P.N. "Understanding and controlling software costs," *IEEE Transactions on Software Engineering* (14:10) 1988, pp 1462-1477.

Boehm, B.W., and Turner, R. "Observations on balancing discipline and agility," Proceedings of the Agile Development Conference, 2003, pp. 32-39.

Boehm, B.W., and Turner, R. *Balancing Agility and Discipline: A Guide for the Perplexed* Addison-Wesley, Boston, MA, 2004.

Bourque, P., Dupuis, R., Abran, A., Moore, J.W., and Tripp, L. "The Guide to the Software Engineering Body of Knowledge," *IEEE Software* (16:6), November/December 1999, pp 35-44.

Bourque, P., Dupuis, R., Abran, A., Moore, J.W., Tripp, L., and Wolff, S. "Fundamental principles of software engineering - a journey," *Journal of Systems and Software* (62:1), May 2002, pp 59-70.

Bratianu, C. "The Learning Paradox and the University," *Journal of Applied Quantitative Methods* (2:4), Winter 2007, pp 374-386.

Brien, A. "Professional Ethics and the Culture of Trust," *Journal of Business Ethics* (17:4), March 1998, pp 391-409.

Britcher, R.N. "Why (Some) Large Computer Projects Fail," in: *Software Runaways*, R.L. Glass (ed.), Prentice Hall, Englewood Cliffs, New Jersey, 1998.

Brooks, F.P. *The Mythical Man-Month* Addison-Wesley, Reading, MA, 1995, p. 322.

Brugha, C.M. "Implications from Decision Science for the Systems Development Life Cycle in Information Systems," *Information Systems Frontiers* (3:1), March 2001, pp 91-105.

Bryant, I., Usher, R., and Johnston, R. *Adult Education and the Postmodern Challenge: Learning Beyond the Limits* Routledge, London, 1997.

Burman, E., and Parker, I. *Discourse Analytic Research: Repertoires and Readings of Texts in Action* Routledge, New York, 1993, p. 401.

Burrell, G., and Morgan, G. *Sociological Paradigms and Organizational Analysis* Ashgate, Burlington, Vermont, 1979, p. 432.

Cao, L., and Ramesh, B. "Agile Software Development: Ad Hoc Practices or Sound Principles?," *IT Professional* (9:2), March-April 2007, pp 41-47.

Cao, L., and Ramesh, B. "Agile Requirements Engineering Practices: An Empirical Study," *IEEE Software* (25:1), January/February 2008 2008, pp 60-67.

Carayannis, E.G., and Sagi, J. "Dissecting the professional culture: insights from inside the IT "black box"," *Technovation* (21) 2001, pp 91-98.

Chapman, S. "Worldwide PC Numbers to Hit 1B in 2008, Forrester Says," in: *CIO*, International Data Group, 2007, p. 2.

Chau, T., and Maurer, F. "Knowledge Sharing in Agile Software Teams," in: *Logic versus Approximation*, Springer, Berlin, 2004, pp. 173-183.

Chau, T., Maurer, F., and Melnik, G. "Knowledge sharing: agile methods vs. Tayloristic methods," in: *Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Linz, Austria, 2003, pp. 302-307.

Checkland, P., and Holwell, S. "Action Research: Its Nature and Validity," *Systemic Practice and Action Research* (11:1), February 1998, pp 9-21.

Cheetham, G., and Chivers, G. "Towards a holistic model of professional competence," *Journal of European Industrial Training* (20:5) 1996.

Chun, A.H.W. "The Agile Teaching/Learning Methodology and Its e-Learning Platform," in: *Advances in Web-Based Learning – ICWL 2004*, W.L.Y. Shi and Q. Li (eds.), Springer Berlin, Heidelberg, 2004, pp. 11-18.

Cockburn, A. "Selecting a project's methodology," *IEEE Software* (17:4), July/August 2000, pp 64-71.

Cockburn, A. *Agile Software Development* Addison-Wesley, Boston, 2002.

Cockburn, A., and Highsmith, J. "Agile Software Development, the People Factor," *IEEE Computer* (34:11), November 2001, pp 131-133.

Constantine, L.L. "Process Agility and Software Reusability: Toward Lightweight Usage-Centered Design," *Information Age* (8:2) 2002, pp 1-10.

Cragg, P.B., and King, M. "Small-Firm Computing: Motivators and Inhibitors," *MIS Quarterly* (17:1), March 1993, pp 47-60.

Creswell, J.W. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*, (3rd ed.) Sage, Los Angeles, 2009, p. 260.

Cunningham, W. "Wiki Design Principles," Portland, Oregon, USA, 2008.

Cusumano, M.A. "Extreme Programming Compared with Microsoft-style Iterative Development," *Communications of the ACM* (50:10), October 2007, pp 15-18.

Cusumano, M.A., and Selby, R.W. "How Microsoft builds software," *Communications of the ACM* (40:6), June 1997, pp 53-61.

Daudelin, M.W. "Learning from experience through reflection," *Organizational Dynamics* (24:3), Winter 1996, pp 36-48.

Davenport, T.H., and Markus, M.L. "Rigor vs. Relevance Revisited: Response to Benbasat and Zmud," *MIS Quarterly* (23:1), March 1999, pp 19-23.

Davison, R.M. "Professional Ethics in Information Systems: A Personal Perspective," *Communications of the AIS* (3:8), April 2000, p 34.

Davison, R.M., Martinsons, M.G., and Kock, N. "Principles of canonical action research," *Information Systems Journal* (14) 2004, pp 65-86.

DeBaar, B. "Why Agile Popped Up on the Radar When it Did," *PM World Today* (11:11), November 2007, p 5.

Decker, B., Ras, E., Rech, J., Klein, B., Reuschling, C., Höcht, C., Kilian, L., Traphoener, R., and Haas, V. "A Framework for Agile Reuse in Software Engineering using Wiki Technology," in: *2nd Workshop on Knowledge Management for Distributed Agile Processes: Models, Techniques, and Infrastructure*, Kaiserslautern, Germany, 2005, pp. 1-4.

DeLone, W.H. "Determinants of Success for Computer Usage in Small Business," *MIS Quarterly* (12:1), March 1988, pp 51-61.

DeMarco, T., and Boehm, B. "The agile methods fray," *IEEE Computer* (35:6), June 2002, pp 90-92.

Denning, P.J. "Computer science and software engineering: filing for divorce?," *Communications of the ACM* (41:8), August 1998, p 128.

Denning, P.J. "The Profession of IT: Who Are We?," *Communications of the ACM* (44:2), February 2001a, pp 15-19.

Denning, P.J. "When IT Becomes a Profession," in: *The Invisible Future*, McGraw Hill, 2001b, p. 19.

Denning, P.J. "Recentering computer science," *Communications of the ACM* (48:11), November 2005, pp 15-19.

Denning, P.J., Comer, D.E., Gries, D., Mulder, M.C., Tucker, A., Turner, A.J., and Young, P.R. "Computing as a discipline," *IEEE Computer* (22:2), February 1989, pp 63-70.

Denning, P.J., and Dunham, R. "The Core of the Third-Wave Professional," *Communications of the ACM* (44:11), November 2001, pp 21-25.

Denning, P.J., and Riehle, R.D. "The Profession of IT: Is Software Engineering Engineering?," *Communications of the ACM* (52:3), March 2009.

DeRemer, F., and Kron, H. "Programming-in-the large versus programming-in-the-small," in: *Proceedings of the international conference on reliable software*, Los Angeles, California, 1975, pp. 114-121.

Dingwall, R.W.J. "Professions and Social Order in a Global Society," in: *Revista Electrónica de Investigación Educativa*, 2004.

Dorst, K., and Dijkhuis, J. "Comparing paradigms for describing design activity," *Design Studies* (16:2), April 1995, pp 261-274.

Dybå, T. "Improvisation in small software organizations," *IEEE Computer* (17:5), September/October 2000, pp 82-87.

El-Kadi, A. "Stop that divorce!," *Communications of the ACM* (42:12), December 1999, pp 27-28.

Elliot, R.K., and Jacobson, P.D. "The Evolution of the Knowledge Professional," *Accounting Horizons* (16:1), March 2002, pp 69-80.

Emery, F. "Characteristics of Socio-Technical Systems," in: *THE SOCIAL ENGAGEMENT OF SOCIAL SCIENCE - Volume ll - THE SOCIO-TECHNICAL SYSTEMS PERSPECTIVE*, E. Trist and H. Murray (eds.), University of Pennsylvania Press, Philadelphia, 1997.

Ensmenger, N.L. "The 'Question of Professionalism' in the Computer Fields," *IEEE Annals of the History of Computing* (23:4) 2001, pp 56-74.

Eppinger, S.D., Whitney, D.E., Smith, R.P., and Gebala, D.A. "A model-based method for organizing tasks in product development," *Research in Engineering Design* (6:1), March 1994, pp 1-13.

Erickson, J., Lyytinen, K., and Siau, K. "Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research," *Journal of Database Management* (16:4), October-December 2005, pp 88-100.

Esser, J.K. "Alive and well after 25 Years: A Review of Groupthink Research," *Organizational Behavior and Human Decision Processes* (73:2-3), February 1998, pp 116-141.

Euler, E.A., Jolly, S.D., and Curtis, H.H.L. "The Failures of the Mars Climate Orbiter and Mars Polar Lander: A Perspective from the People Involved," in: *24th Annual AAS Guidance and Control Conference*, American Astronautical Society, Breckenridge, Colorado, 2001.

Everett, J.E., and Watson, J. "Small Business Failure and External Risk Factors," *Small Business Economics* (11:4), December 1998, pp 371-391.

Fægri, T.E. "Exploratory knowledge creation in agile software development projects," in: *The 31st Information Systems Research Seminar in Scandinavia*, Åre Sweden, 2008.

Fallman, D. "Design-oriented human-computer interaction," in: *Proceedings of the SIGCHI conference on Human factors in computing systems*, G. Cockton and P. Korhonen (eds.), Association for Computing Machinery, Ft. Lauderdale, Florida, USA, 2003, pp. 225-232.

Faraj, S., and Sproull, L. "Coordinating Expertise in Software Development Teams," *Management Science* (46:12), December 2000, pp 1554-1568.

Fayad, M.E., Laitinen, M., and Ward, R.P. "Thinking objectively: software engineering in the small," *Communications of the ACM* (43:3), March 2000, pp 115-118.

Florac, W.A., Carleton, A.D., and Barnard, J.R. "Statistical Process Control: Analyzing a Space Shuttle Onboard Software Process," *IEEE Software* (17:4), July/August 2000, pp 97-106.

Ford, G. "A Mature Profession of Software Engineering," Software Engineering Institute, Carnegie Mellon University, Pittsburg.

Fowler, M. "The New Methodology," http://www.martinfowler.com/articles/newMethodology.html, 2005.

Fowler, M., and Highsmith, J. "The Agile Manifesto," in: *Software Development*, 2001, pp. 28-32.

Frakes, W.B., and Kang, K. "Software Reuse Research: Status and Future," *IEEE Transactions on Software Engineering* (31:7) 1995, pp 529-536.

Freidus, N. "Digital storytelling for reflective practice in communities of learners," *ACM SIGGROUP Bulletin* (23:2), August 2002, pp 24-26.

Garlan, D., Gluch, D.P., and Tomakyo, J.E. "Agents of Change: Educating Software Engineering Leaders," *IEEE Computer* (30:11), November 1997, pp 59-65.

Gelperin, D., and Hetzel, B. "The Growth of Software Testing," *Communications of the ACM* (31:6) 1988.

Giddens, A. *The Constitution of Society* University of California Press, Berkeley, CA, 1984, p. 402.

Ginige, A., and Murugesan, S. "Web engineering: an introduction," *IEEE Multimedia* (8:1), January-March 2001, pp 14-18.

Glass, R.L. *Software Runaways* Prentice Hall, Englewood Cliffs, New Jersey, 1998.

Glass, R.L. "Questioning the software engineering unquestionables," *IEEE Software* (20:3) 2003, pp 119-120.

Google "Life at Google," 2008.

Gorla, N., and Lam, Y.W. "Who should work with whom?: building effective software project teams," *Communications of the ACM* (47:6), June 2004, pp 79-82.

Gorman, R.A. *The Dual Vision: Alfred Schutz and the myth of phenomenological science* Routledge & Paul Kegan, London, 1977.

Gotterbarn, D., Miller, K., and Rogerson, S. "Software Engineering Code of Ethics is Approved," *Communications of the ACM* (42:10), October 1999, pp 102-107.

Gower, B. *Scientific Method: A Historical and Philosophical Introduction* Routledge, Ney York, 1996, p. 288.

Grisham, P.S., and Perry, D.E. "Customer relationships and Extreme Programming," in: *2005 workshop on Human and social factors of software engineering*, St. Louis, MO, USA, 2005, pp. 1-6.

Hazzan, O. "The reflective practitioner perspective in software engineering education," *The Journal of Systems and Software* (63:3) 2002, pp 161-171.

Hazzan, O., and Tomayko, J. "The Reflective Practitioner Perspective in eXtreme Programming," in: *Third XP Agile Universe Conference*, Springer, New Orleans, 2003, pp. 51-61.

Hazzan, O., and Tomayko, J. "The Reflective Practitioner Perspective in Software Engineering," in: *Conference on Human Factors in Computing Systems*, Vienna, Austria, 2004a, p. 4.

Hazzan, O., and Tomayko, J.E. "Human Aspects of Software Engineering: The Case of Extreme Programming," in: *5th International Conference on Extreme Programming and Agile Processes in Software Engineering, XP 2004*, J. Eckstein and H. Baumeister (eds.), Garmisch-Partenkirchen, Germany, 2004b, pp. 303-311.

Hazzan, O., and Tomayko, J.E. "Reflection Processes in the Teaching and Learning of Human Aspects of Software Engineering," in: *Proceedings of the 17th Conference on Software Engineering Education and Training*, IEEE Computer Society, Norfolk, Virginia, 2004c.

Hazzan, O., and Tomayko, J.E. "Reflection and Abstraction in Learning Software Engineering's Human Aspects," *IEEE Computer* (38:6), June 2005, pp 39-45.

Heiskanen, A. "Reflecting over a practice: Framing issues for scholar understanding," *Information Technology & People* (8:4) 1995.

Hesse-Biber, S., Dupuis, P., and Kinder, T.S. "HyperRESEARCH: A computer program for the analysis of qualitative data with an emphasis on hypothesis testing and multimedia analysis," *Qualitative Sociology* (14:4) 1991, pp 289-306.

Hevner, A. "Design Science," J. Babb (ed.), Richmond, Virginia, 2007.

Hevner, A.R., March, S.T., Park, J., and Ram, S. "Design Science in Information Systems Research," *MIS Quarterly* (28:1) 2004.

Highsmith, J. "What Is Agile Software Development?," *CrossTalk: The Journal of Defense Software Engineering*), October 2002, pp 4-9.

Highsmith, J., and Cockburn, A. "Agile software development: the business of innovation," *IEEE Computer* (34:9), September 2001, pp 120-127.

Highsmith, J., and Cockburn, A. "Agile Software Development: The Business of Innovation," *IEEE Computer*), September 2002.

Highsmith, J.A. *Adaptive Software Development: A Collaborative Approach* Dorset House, New York, 2000.

Hill, A.W., Dong, A., and Agogino, A.M. "Towards Computational Tools for Supporting the Reflective Team," in: *Artificial Intelligence in Design '02*, J.S. Gero (ed.), Kluwer Academic Publishers, Dordrecht, 2002, pp. 305-326.

Hirschheim, R., and Klein, H.J. "Four Paradigms of Information Systems Development," *Communications of the ACM* (32:10), October 1989, pp 1199-1216.

Hirschheim, R., and Klein, H.K. "Crisis in the IS Field? A Critical Reflection on the State of the Discipline," *Journal of the Association for Information Systems* (4:5), October 2003, pp 237-293.

Hirschheim, R., Klein, H.K., and Lyytinen, K. "Exploring the intellectual structures of information systems development: a social action theoretic analysis," *Accounting, Management and Information Technologies* (6:1-2), January-June 1996, pp 1-64.

Hirschheim, R., and Newman, M. "Symbolism and Information Systems Development: Myth, Metaphor and Magic," in: *Qualitative Research in Information Systems*, M. Myers and D. Avison (eds.), Sage Publications, London, 2004, pp. 241-275.

Hjørland, B., and Albrechtsen, H. "Toward a New Horizon in Information Science: Domain-Analysis," *Journal of the American Society for Information Science* (46:6) 1995, pp 400-425.

Holmes, N. "Fashioning a Foundation for the Computing Profession," *IEEE Computer* (33:7), July 2000, pp 97-98.

Humphrey, W. *A Discipline for Software Engineering: The Complete PSP Book* Addison-Wesley, Reading, MA, 1995.

Humphrey, W. *Introduction to Team Software Process* Addison-Wesley, Boston, 2000.

Iivari, J., Hirschheim, R., and Klein, H.K. "A Paradigmatic Analysis Contrasting Information Systems Development Approaches and Methodologies," *Information Systems Research* (9:2) 1998, pp 164-193.

Jackson, T.P. "Court's Findings of Fact: United States of American cs. Microsoft Corporation," in: *F3800*, U.S.D.C.f.t.D.o. Columbia (ed.), United States District Court for the District of Columbia, Washington, D.C., 1999.

Jacobson, I., Booch, G., and Rumbaugh, J. *The Unified Software Development Process* Addison-Wesley, Reading, MA, 1999.

Järvinen, P. "Action research as an approach in design science," Department of Computer Sciences - University of Tampere, Tampere, Finland, 2005, p. 16.

Järvinen, P. "Action Research is Similar to Design Science," *Quality & Quantity* (41) 2007, pp 37-54.

Jarvis, P. "Reflective practice and nursing," *Nurse Education Today* (12:3) 1992, pp 174-181.

Jeffries, R., Anderson, A., and Hendrickson, C. *Exreme Programming Installed* Addison-Wesley, Upper Saddle River, NJ, 2001.

Johnson, D.L., and Brodman, J.G. "Applying CMM Project Planning Practices to Diverse Environments," *IEEE Software* (17:4), July/August 2000, pp 40-47.

Jones, D. "The Interpretive Auditor," *Management Communication Quarterly* (15:3), February 2002, pp 466-471.

Jonsen, A.R., Clarence H. Braddock, I., and Edwards, K.A. "Professionalism," University of Washington, Seattle, 1998.

Kaboub, F. "Positivist and Hermeneutic Paradigms: A Critical Evaluation under the Structure of Scientific Practice," University of Missouri-Kansas City, Kansas City, MO, 2001.

Keen, P.G.W. "Information Systems and Organizational Change," *Communications of the ACM* (24:1), January 1981, pp 24-33.

Kelle, U. "Computer-Assisted Analysis of Qualitative Data," in: *Discussion paper series of the LSE Methodology Institute*, University of Bremen, Bremen, Germany, 1997, p. 24.

Klein, H.K., and Myers, M.D. "A Set of Principles for Conducting and Evaluating Intrepretive Field Studies in Information Systems," *MIS Quarterly* (23:1), March 1999, pp 67-94.

Klobas, J.E., and McGill, T. "Indentification of Technological Gatekeepers in the Information Technology Profession," *Journal of the American Society for Information Science* (46:8), September 1995.

Knight, J.C., and Leveson, N.G. "Should software engineers be licensed?," *Communications of the ACM* (45:11), November 2002, pp 87-90.

Kock, N. "Action research: lessons learned from a multi-iteration study of computer-mediated communication in groups," *IEEE Transactions on Professional Communication* (46:2), June 2003, pp 105-128.

Kostamovaara, H., and Seppänen, M. "Are services really that valuable? Two diverse cases in an intra-company technology transfer setting," in: *IEEE 2007 International Conference on Industrial Engineering and Engineering Managment*, M. Helander, M. Xie, R. Jiao and K.C. Tan (eds.), IEEE, Singapore, 2007, pp. 1433-1437.

Kraut, R.E., and Streeter, L.A. "Coordination in software development," *Communications of the ACM* (38:3), March 1995, pp 69-81.

Kuhn, S. "The software design studio: an exploration," *IEEE Software* (15:2), March-April 1998, pp 65-71.

Kuhn, T.S. *The Structure of Scientific Revolutions*, (3rd ed.) University of Chicago Press, Chicago, 1996, p. 212.

Lalle, B. "The management science researcher between theory and practice," *Organization Studies* (24:7), September 2003, pp 1097-1115.

Larman, C., and Basili, V.R. "Iterative and Incremental Development: A Brief History," *IEEE Computer* (36:6), June 2003, pp 47-56.

Larson, M.L. "The Rise of Professionalism," *Pediatrics* (63) 1979, p 490.

Lau, F. "Toward a framework for action research in information systems studies," *Information Technology & People* (12:2) 1999.

Lee, A.S. "A Scientific Methodoloyg for MIS Case Studies," *MIS Quarterly* (13:1), March 1989, pp 33-50.

Lee, A.S. "Integrating Positivist and Interpretive Approaches to Organizational Research," *Organization Science* (2:4), November 1991, pp 342-365.

Lee, A.S. "Rigor and Relevance in MIS Research: Beyond the Approach of Positivism Alone," *MIS Quarterly* (23:1), March 1999, p 4.

Lee, A.S. "Challenges to Qualitative Researchers in Information Systems," in: *Qualitative Research in IS: Issues and Trends,* E.M. Trauth (ed.), Idea Group Publishing, Hershey, PA, 2001, pp. 240-270.

Lee, A.S. "A Scientific Methodology for Scientific Case Studies," in: *Qualitative Research in Information Systems,* M.D. Myers and D. Avison (eds.), Sage, London, 2002, pp. 169-180.

Lee, A.S. "Thinking about Social Theory and Philosophy for Information Systems," in: *Social Theory and Philosophy for Information Systems,* J. Mingers and L. Willcocks (eds.), John Wiley & Sons, West Sussex, 2004, pp. 1-26.

Lee, A.S. "Action is an Artifact," in: *Information Systems Action Research,* N. Kock (ed.), Springer, New York, 2007, pp. 42-60.

Lee, A.S. "Three Wishes I Have for the Information Systems Field," in: *Eleventh Annual Conference of the Southern Association for Information Systems,* Richmond, VA, 2008.

Lee, A.S., and Baskerville, R.L. "Generalizing generalizability in information systems research," *Information Systems Research* (14:3) 2003, pp 221-245.

Lee, A.S., and Hubona, G.S. "A Scientific Basis for Rigor and Relevance in Information Systems Research," Virginia Commonwealth University, Richmond, Virginia, 2008, p. 69.

Lee, R. (ed.) *Information Technology for the Social Scientist.* Routledge, London, 1995.

Lehman, M.M. "Software's future: managing evolution," *IEEE Software* (15:1), January/February 1998, pp 40-44.

Leuf, B., and Cunningham, W. *The Wiki Way* Addison-Wesley, Boston, 2001.

Lewin, K. "Frontiers in Group Dynamics," *Human Relations* (1:2) 1947, pp 143-153.

Lindstrom, L., and Jeffries, R. "Extreme Programming and Agile Software Development Methodologies," *Information Systems Management* (21:3), Summer 2004, pp 41-60.

Lindvall, M., and Rus, I. "Process diversity in software development," *IEEE Software* (17:4), July/August 2000, pp 14-18.

Lohr, S. "Part Artist, Part Hacker, And Full-Time Programmer," in: *The New York Times,* New York, 1996.

Louridas, P., and Loucopoulos, P. "A generic model for reflective design," *ACM Transactions on Software Engineering and Methodology* (9:2), April 2000, pp 199-237.

March, S.T., and Smith, G.T. "Design and natural science research on information technology," *Desicion Support Systems* (15) 1995, pp 251-266.

Markus, M.L. "Power, Politics, and MIS Implementation," in: *Qualitative Research in Information Systems,* M.D. Myers and D. Avison (eds.), Sage, London, 2002, pp. 19-50.

Marshall, J. "Living systemic thinking," *Action Research* (2:3) 2004, pp 305-325.

Mårtensson, P., and Lee, A.S. "Dialogical Action Research at Omega Corporation," Virginia Commonwealth University, Richmond, Virginia, 2002, p. 34.

Mårtensson, P., and Lee, A.S. "Dialogical Action Research at Omega Corporation," *MIS Quarterly* (28:3) 2004.

Martin, C.D., and Martin, D.H. "Professional codes of conduct and computer ethics education," *ACM SIGSAC Review* (8:3), September 1990, pp 1-12.

Martin, R.C. *Agile Software Development: Principles, Patterns, and Practices* Prentice Hall, Upper Saddle River, 2003.

Mathiassen, L. "Reflective Systems Development," *Scandanavian Journal of Information Systems* (10:2) 1998, p 67.

Mathiassen, L., and Purao, S. "Educating reflective systems developers," *Information Systems Journal* (12:2), April 2002, pp 81-102.

Maurer, F., and Martel, S. "Extreme programming: Rapid development for Web-based applications," *IEEE Internet Computing* (6:1), January-February 2002, pp 86-90.

McAvoy, J., and Butler, T. "The impact of the Abilene Paradox on double-loop learning in an agile team," *Information and Software Technology* (49:6), June 2007, pp 552-563.

McAvoy, J., and Butler, T. "A Failure to Learn in a Software Development Team: The Unsuccessful Introduction of an Agile Method," in: *Information Systems Development: Challenges in Practice, Theory, and Education,* C. Barry, K. Conboy, M. Lang, G. Wojtkowski and W. Wojtkowski (eds.), Springer, New York, 2008.

McBreen, P. *Questioning Extreme Programming* Addison-Wesley Longman Publishing Co., Inc., 2002a, p. 201.

McBreen, P. *Software Craftsmanship* Addison-Wesley, Boston, 2002b, p. 187.

McCalla, G. "Software Engineering Requires Individual Professionalism," *Communications of the ACM* (45:11), November 2002, pp 98-101.

McConnell, S. *Code Complete* Microsoft Press, Redmond, WA, 2004a.

McConnell, S. *Professional Software Development* Addison-Wesley, Boston, 2004b.

McConnell, S., and Tripp, L. "Professional Software Engineering: Fact or Fiction?," *IEEE Software* (16:6), November/December 1999, pp 13-18.

McDonald, A., and Welland, R. "Web engineering in practice," in: *Proceedings of the Fourth Workshop on Web Engineering*, 2001a.

McDonald, A., and Welland, R. "Web Engineering in Practice," in: *Proceedings of the Fourth WWW10 Workshop on Web Engineering*, Hong Kong, China, 2001b, p. 10.

McGirt, E. "Hacker. Dropout. CEO.," in: *Fast Company.com*, 2007.

McKay, J., and Marshall, P. "The dual imperatives of action research," *Information Technology & People* (14:1) 2001, pp 46-59.

Melnik, G., and Maurer, F. "Introducing agile methods: three years of experience," in: *30th Euromicro Conference*, Rennes, France, 2004, pp. 334-341.

Mingers, J. "Combining IS research methods: Towards a pluralist methodology," *Information Systems Research* (12:3), September 2001, p 240.

Miser, H.J. "President's Symposium: Science and Professionalism in Operations Research," *Operations Research* (35:2), March - April 1987, pp 314-319.

Moore, J.E. "One Road to Turnover: An Examination of Work Exhaustion in Technology Professionals," *MIS Quarterly* (24:1), March 2000, pp 141-168.

Morse, J.M., Barrett, M., Mayan, M., Olson, K., and Spiers, J. "Verification Strategies for Establishing Reliability and Validity in Qualitative Research," *International Journal of Qualitative Methods* (1:2), Spring 2002, pp 1-19.

Myers, M.D. "Qualitative research in information systems," *MIS Quarterly* (21:2), June 1997, pp 241-242.

Myers, M.D., and Avison, D.E. "An Introduction to Qualitative Research in Information Systems," in: *Qualitative Research in Information Systems: A Reader*, M.D. Myers and D.E. Avison (eds.), Sage Publications, London, 2002.

Nerur, S., and Balijepally, V. "Theoretical Reflections on Agile Development Methodologies: The Traditional Goal of Optimization and Control is Making Way for Learning and Innovation," *Communications of the ACM* (50:3), March 2007, pp 79-83.

Nerur, S., Mahapatra, R., and Mangalaraj, G. "Challenges of Migrating to Agile Methodologies," *Communications of the ACM* (48:5), May 2005, pp 73-78.

Newkirk, J. "Introduction to agile processes and extreme programming," Proceedings of the 24th International Conference on Software Engineering Orlando, Florida, 2002, pp. 695-696.

Nueibeh, B. "Ariane 5: Who Dunnit?," *IEEE Software* (14:3), May/June 1997, pp 15-16.

Nunes, N.J., and Cunha, J.F. "Wisdom: A Software Engineering Method for Small Software Development Companies," *IEEE Software* (17:5), September/October 2000, pp 113-119.

Orden, A. "The emergence of a profession," *Communications of the ACM* (10:3), March 1967, pp 145-147.

Orlikowski, W. "The Duality of Technology: Rethinking the concept of technology in organizations," *Organization Science* (2:2) 1992, pp 398-427.

Orlikowski, W.J., and Barley, S.R. "Technology and Institutions: What Can Research on Information Technology and Research on Organizations Learn from Each Other?," *MIS Quarterly* (25:2) 2002.

Orlikowski, W.J., and Baroudi, J.J. "The Information Systems Professional: Myth or Reality?," Stern School of Business, New York University, New York.

Orlikowski, W.J., and Iacono, C.S. "Research Commentary: Desperately Seeking the "IT" in IT Research - A Call to Theorizing the IT Artifact," *Information Systems Research* (12:2), June 2001, pp 121-134.

Orlikowski, W.J., and Robey, D. "Information Technology and the Structuring of Organizations," *Information Systems Research* (2:2), June 1991, pp 143-169.

Oz, E. "Ethical Standards for Information Systems Professionals: A Case for a Unified Code," *MIS Quarterly* (16:4), December 1992, pp 423-433.

Palmer, S., and Felsing, J. *A Practical Guide to Feature-Driven Development* Prentice Hall, Upper Saddle River, NJ, 2002.

Parker, D.B. "Rules of Ethics in Information Processing," *Communications of the ACM* (11:3), March 1968, pp 198-201.

Parnas, D. "Why software developers should be licensed," *Engineering Dimensions* (22:3), May/June 2001, pp 36-39.

Parnas, D., and Clements, P. "A Rational Design Process: How and Why to Fake It," *IEEE Transactions on Software Engineering* (February) 1986, pp 251-257.

Parnas, D.L. "Software engineering programs are not computer science programs," *IEEE Software* (16:6), November/December 1999, pp 19-30.

Parnas, D.L. "Licensing Software Engineergs in Canada," *Communications of the ACM* (45:11) 2002, pp 96-98.

Paulk, M.C., and Institute, S.E. *The Capability Maturity Model: Guidelines for Improving the Software Process* Addison-Wesley, Reading, MA, 1995.

Pinna, S., Mauri, S., Lorrai, P., Marchesi, M., and Serra, N. "XPSwiki: An Agile Tool Supporting the Planning Game," in: *Extreme Programming and Agile Processes in Software Engineering*, Springer, Berlin, 2003.

Polanyi, M. *The Tacit Dimension* Peter Smith, Goucester, Mass., 1983.

Pour, G., Griss, M.L., and Lutz, M. "The Push to make Software Engineering Respectable," *IEEE Computer* (33:5), May 2000, pp 35-43.

Pressman, R.S. "Can Internet-Based Applications Be Engineered?," *IEEE Software* (15:5), Sep/Oct 1998 1998, pp 104-110.

Prowell, S.J., Trammell, C.J., Linger, R.C., and Poore, J.H. *Cleanroom Software Engineering: Technology and Process* Addison-Wesley, Reading, MA, 1999.

Purgathofer, P. "Is Informatics a design discipline?," *Poiesis & Praxis* (4:6), December 2006, pp 303-314.

Rajlich, V. "Incremental redocumentation using the web," *IEEE Software* (17:5), September/October 2000, pp 102-106.

Rajlich, V. "Changing the Paradigm of Software Engineering," *Communications of the ACM* (49:8), August 2006, pp 67-70.

Raymond, J., and Yee, C. "The Collaborative Process and Professional Ethics," *IEEE Transactions on Professional Communication* (33:2), June 1990, pp 77-81.

Reddy, R., Wood, R.T., and Cleetus, K.J. "Concurrent engineering: The DARPA initiative: encouraging new industrial practices," *IEEE Spectrum* (27:8), July 1991, pp 26-27,30.

Reifer, D.J. "Web development: estimating quick-to-market software," *IEEE Software* (17:6), November/December 2000, pp 57-64.

Rettig, M., and Simons, G. "A Project Planning and Development Process for Small Teams," *Communications of the ACM* (36:10), October 1993, pp 45-55.

Rising, L., and Janoff, N.S. "The Scrum software development process for small teams," *IEEE Software* (17:4), July/August 2000, pp 26-32.

Ritzer, G. "Professionalization, Bureaucratization and Rationalization: The Views of Max Weber," *Social Forces* (53:4), June 1975, pp 627-634.

Ritzer, G., and Walczak, D. "Rationalization and the Deprofessionalization of Physicians," *Social Forces* (67:1), September 1988, pp 1-22.

Romme, A.G.L., and Witteloostuijin, A.v. "Circular organizing and triple loop learning," *Journal of Organizational Change Management* (12:5) 1999.

Rosier, G. "Using reflective reports to improve the case method," *The Journal of Management Development* (7:8) 2002, pp 589-597.

Rothenberg, A. "The Process of Janusian Thinking in Creativity," *Archives of General Psychiatry* (24), March 1971, pp 195-205.

Russ, M.L., and McGregor, J.D. "A Software Development Process for Small Projects," *IEEE Software* (17:5), September/October 2000, pp 96-101.

Schaefer, R. "A Critical Programmer Searches for Professionalism," *ACM SIGSOFT Software Engineering Notes* (31:1), July 2006, pp 83-100.

Schön, D.A. *Technology and Change: The New Heraclitus* Pergamon Press, Oxford, 1967, p. 248.

Schön, D.A. *Beyond the Stable State* W W Norton and Compnay, Inc, New York, 1973, p. 254.

Schön, D.A. "Generative Metaphor: A Perspective on Problem-Setting in Social Policy," in: *Metaphor and Thought,* A. Ortony (ed.), Cambridge University Press, Cambridge, 1979, pp. 254-283.

Schön, D.A. *The Reflective Practitioner: How Professionals Think in Action* Basic Books, New York, 1983, pp. x, 374 p.

Schön, D.A. *Educating the reflective practitioner*, (1st ed.) Jossey-Bass, San Francisco, 1987, pp. xvii, 355 p .

Schön, D.A., and Bennett, J. "Reflective Conversation With Materials," in: *Bringing Design to Software*, T. Winograd (ed.), ACM Press, New York, 1996, pp. 171-184.

Schön, D.A., and Rein, M. *Frame Reflection* Basic Books, New York, 1994.

Schultze, U. "A Confessional Account of an Ethnography about Knowledge Work," *MIS Quarterly* (24:1), March 2000, pp 3-41.

Schutz, A. *Collected Papers I. The Problem of Social Reality* Springer, The Hague, 1962.

Schutz, A. *The Phenomenology of the Social World* Northwestern University Press, Evanston, Illinois, 1967, p. 255.

Schwaber, K., and Beedle, M. *Agile Software Development with Scrum* Prentice-Hall, Upper Saddle River, NJ, 2002.

Scott, W.R. *Organizations: Rational, Natural and Open systems*, (5th ed.) Harper-Collins, New York, 2003.

Searle, J.R. "Social ontology: Some basic principles," *Anthropological Theory* (6:1) 2006, pp 12-29.

Senge, P.M. *The Fifth Discipline: The Art and Practice of the Learning Organization* Currency Doubleday, New York, 1994.

Sharp, H., and Robinson, H. "An Ethnographic Study of XP Practice," *Empirical Software Engineering* (9:4), December 2004, pp 353-375.

Shaw, M. "Prospects for an engineering discipline of software," *IEEE Software* (7:6), November 1990, pp 15-24.

Simon, H.A. "Bounded Rationality and Organizational Learning," *Organization Science* (2:1), February 1991, pp 125-134.

Simon, H.A. *The sciences of the artificial*, (3rd ed.) M.I.T. Press, Cambridge, Massachusetts, 1996, p. 231.

Smith, M.K. *Local Education* Open University Press, Buckingham, 1994.

Smith, M.K. "Donald Schön: learning, reflection and change," in: *the encyclopedia of informal education*, 2001.

Socha, D., and Walter, S. "Is Designing Software Different From Designing Other Things?," *International Journal of Engineering Education* (22:3) 2006, pp 540-550.

Society, I.C. "Guide to the Softare Engineering Body of Knowledge," IEEE Computer Society Professional Practices Committee, Los Alamitos, California, USA, p. 203.

Sorensen, R. "A Comparison of Software Development Methodologies," Software Technology Support Center, Hill Air Force Base, Utah, pp. 12-18.

Speed, J.R. "What do you mean I can't call myself a software engineer?," *IEEE Software* (16:6), November/December 1999, pp 45-50.

Stapleton, J. *DSDM, Dynamic Systems Development Method: The Method in Practice* Addison-Wesley, Reading, MA, 1997.

Stempfle, J., and Badke-Schaub, P. "Thinking in design teams - an analysis of team communication," *Design Studies* (23:5), September 2002, pp 473-496.

Straub, D.W., and Welke, R.J. "Coping With Systems Risk: Security Planning Models for Management Decision Making," *MIS Quarterly* (22:4), December 1998, pp 441-469.

Strauss, A., and Corbin, J. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, (2nd ed.) SAGE Publications, Thousand Oaks, 1998, p. 312.

Subramaniam, V., and Hunt, A. *Practices of an Agile Developer - Working in the Real World* The Pragmatic Bookshelf, Raleigh, North Carolina, 2006, p. 203.

Sullivan, D. "comScore Media Matrix Search Engine Ratings," in: *SearchEngineWatch.com*, 2006.

Sutton, S.M. "The Role of Process in a Software Start-up," *IEEE Software* (17:4), July/August 2000, pp 33-39.

Tomayko, J.E., and Hazzan, O. *Human Aspects of Software Engineering* Charles River Media, Inc., Hingham, MA, 2004, p. 338.

Trauth, E.M. "The Professional Responsibility of the TechKnowledgable," *Computers and Society* (13:1), Winter 1982, pp 17-21.

Trauth, E.M. "The Choice of Qualitative Methods in IS Research," in: *Qualitative Research in IS: Issues and Trends,* E.M. Trauth (ed.), Idea Group Publishing, Hershey, PA, 2001, pp. 1-19.

Truex, D., Baskerville, R., and Travis, J. "Amethodical systems development: the deferred meaning of systems development methods," *Accounting Management and Information Technologies* (10) 2000, pp 53-79.

Turk, D., France, R., and Rumpe, B. "Limitations of Agile Software Processes," Third International Conference on eXtreme Programming and Agile Processes in Software Engineering, 2002, p. 4.

Ulrich, W. "Reflective Practice in the Civil Society: The Contribution of Critically Systemic Thinking," *Reflective Practice* (1:2) 2000, pp 247-268.

Vessey, T.M.S.a.I. "The Role of Cognitive Fit in the Relationship between Software Comprehension and Modification," *MIS Quarterly* (30:1) 2006.

Vince, R. "Organizing Reflection," *Management Learning* (33:1), March 2002, pp 63-78.

Visser, M. "Deutero-learning in organizations: a review and a reformulation," *Academy of Management review* (32:2), April 2007, pp 659-667.

Wake, W.C. *Extreme Programming Explored* Addison-Wesley, Boston, MA, 2002.

Wakkary, R. "Framing complexity, design and experience: a reflective analysis," *Digital Creativity* (16:2) 2005, pp 65-78.

Walsham, G. "Interpretive Case Studies in IS Research: Nature and Method," in: *Qualitative Research in Information Systems: A Reader,* M.D. Myers and D. Avison (eds.), Sage Publications, London, 2002, pp. 101-114.

Wasserman, A.I. "Toward a discipline of software engineering," *IEEE Software* (13:6), November 1996, pp 23-31.

Watson, J., and Everett, J.E. "Do Small Businesses Have High Failure Rates?," *Journal of Small Business Management* (34:4), October 1996, pp 45-62.

Weick, K.E. "What Theory Is Not, Theorizing Is," *Administrative Science Quarterly* (40:3), September 1995, pp 385-390.

Wells, J.D. "Extreme Programming: A Gentle Introduction," 2000.

Whinston, A.B., and Geng, X. "Operationalizing the Essential Role of the Information Technology Artifact in Information Systems Research: Gray Area, Pitfalls, and the Importance of Strategic Ambiguity " *MIS Quarterly* (28:2) 2004.

Whyte, G. "Groupthink Reconsidered," *Academy of Management Review* (14:1) 1989, pp 40-56.

Williams, L., and Cockburn, A. "Agile software development: it's about feedback and change," *IEEE Computer* (36:6), June 2003, pp 39-43.

Williams, L., and Kessler, R.R. "All I really need to know about pair programming I learned in kindergarten," *Communications of the ACM* (43:5) 2000a, pp 108-114.

Williams, L., Kessler, R.R., Cunningham, W., and Jefferies, R. "Stengthening the case for pair programming," *IEEE Software* (17:4) 2000b, pp 19-25.

Wolcott, H.E. *Writing Up Qualitative Research*, (3rd ed.) Sage, Los Angeles, 2009, p. 191.

Womack, J., Jones, D., and Roos, D. *The Machine that Changed the World: The Story of Lean Production* Harper-Collins, New York, 1991.

Yin, R.K. *Case study research : design and methods*, (2nd ed.) Sage Publications, Thousand Oaks, 1994, pp. xvii, 171 p.