FORMALIZATION AND REFINEMENT OF A FORMAL APPROACH TO ECLECTIC SOFTWARE DEVELOPMENT

by Mingu Lee

M.S., May 2001, Johns Hopkins University
B.S., March 1998, Ohio State University

A Dissertation submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

January 31, 2011

Dissertation directed by

Theresa Jefferson
Assistant Professor of Engineering Management and Systems Engineering

The School of Engineering and Applied Science of The George Washington University certifies that Min-Gu Lee has passed the Final Examination for the degree of Doctor of Philosophy as of May 18, 2010. This is the final and approved form of the dissertation.

FORMALIZATION AND REFINEMENT OF A FORMAL APPROACH TO ECLECTIC SOFTWARE DEVELOPMENT

Mingu Lee

Dissertation Research Committee:

Theresa Jefferson, Assistant Professor of Engineering Management and Systems Engineering, Dissertation Director

John Harrald, Professor Emeritus of Engineering Management, Committee Member

Julie J.C.H. Ryan, Associate Professor of Engineering Management and Systems Engineering, Committee Member

Michael A. Stankosky, Professor of Engineering Management and Systems Engineering, Committee Member

Don-In Kang, Computer Scientist, University of Southern California Information Sciences Institute, Committee Member

© Copyright 2010 by Min-Gu Lee
All rights reserved

Dedication

This thesis is dedicated to my parents who have encouraged and believed in me since the beginning of my learning. Also, this thesis is dedicated to my wife, Yuri, and my son, Jake, who have been a great source of love and happiness.

Acknowledgments

The author wishes to express his gratitude to Dr. Theresa Jefferson for her guidance during the course of research.

Abstract of Dissertation

FORMALIZATION AND REFINEMENT OF A
FORMAL APPROACH TO ECLECTIC
SOFTWARE DEVELOPMENT

Over the last 10 years the private sector has shown considerable interest in agile software development methods, the same can be said of the government sector. However, agile methods present limitations for the environment of government software development. The Eclectic Software Development (ESD) approach provides a framework that capitalizes on the benefits of agile and traditional methodologies for the government sector while minimizing the limitations of both. The theory behind ESD is the selective use of the right tools, methodologies, processes, and human resources by project leadership at the right time, within the confines and structures already defined for large-scale and contract-based government projects. This study used action research to apply ESD practices to real government projects in a cyclic manner to both validate and refine ESD.

Table of Contents

Dedication iv
Acknowledgments v
Abstract of Dissertation vi
Table of Contents vii
List of Figures x
List of Tables xiv
Chapter 1: Introduction 1
1.1 Problem Statement 1
1.2 Background 2
1.2.1 Government Software Development Challenge Overview 2
1.2.2 Current Situation – Agile Adoption 3
1.2.3 Relevant Research 4
1.3 Purpose 5
1.4 Significance 6
1.5 Scope 7
1.6 Organization of the Document 7
Chapter 2: Literature Review 8
2.1 Software Development Methodology 8
2.1.1. Overview 8
2.1.2 Plan-Driven Methodology 11
2.1.2.1 Plan-Driven Methodology Overview 11
2.1.2.2 The Waterfall Model 11
2.1.2.3 The Prototyping Model 12
2.1.2.4 Rapid Application Development Model 12
2.1.2.5 Spiral Model 13
2.1.2.6 Rational Unified Process 13
2.1.3 Agile Software Development 15
2.1.3.1 Agile Software Development Overview 15
2.1.3.2 Extreme Programming (XP) 16
2.1.3.3 Crystal 17
2.1.3.4 Adaptive Software Development (ASD) 18
2.1.3.6 Feature-Driven Development (FDD) 20
2.1.3.7 Dynamic Systems Development Method (DSDM) 21
2.1.4 Bridging Agile and Plan-Driven Development Methods 21
2.1.4.1 Applying Agility in Large Projects 21
2.1.4.2 Applying Agility in the Telecommunications Industry 23
2.1.4.3 Applying Agility in Government Projects 23
2.2 Eclectic Software Development 27
2.2.1 ESD Background 27
2.2.2 ESD Executive Summary 27
2.2.3 Philosophy of ESD 31
2.2.4 Government Sector Contract-Based Projects 34
2.2.5 Three Steps of ESD 35
2.2.5.1 Summary 35

2.2.5.2 Step 1: Recognizing Organization Standard 36
2.2.5.3 Step 2: Assessing Project Factors 36
2.2.5.3.1 Tools 39
2.2.5.3.2 Methodologies 40
2.2.5.3.3 Processes 42
2.2.5.3.4 People 42
2.2.5.3.5 Leadership 44
2.2.5.4 Step 3: Recognizing and Responding to Project Circumstances 46
2.2.5.4.1 Overview 46
2.2.5.4.2 Continuous Integrations and Milestone Demonstrations 47
2.2.5.4.3 Ask Simple Questions 48
2.2.5.4.4 Do Simple Math 49
2.2.5.4.5 Adopt Management by Walking Around (MBWA) 50
2.2.6 Informal Assessment 51
2.2.7 ESD Agile Best Practice Checklist 58
2.2.7.1 Introduction to Agile Software Programming 63
2.2.7.1.1 Motivation 63
2.2.7.1.2 Objectives 64
2.2.7.1.3 Manifesto 64
2.2.7.1.4 Principles 64
2.2.7.2 Popular Agile Software Development Methodologies 64
2.2.7.2.1 Scrum – Process framework 65
2.2.7.2.2 eXtreme Programming 66
2.2.7.3 Additional Commercial Agile Best Practices 68
2.2.7.3.1 Management 69
2.2.7.3.2 Technical 69
2.2.7.3.3 Measurement 69
2.2.7.4 Scaling Agile 73
2.2.7.4.1 Challenges of Applying Agile Method to Enterprise 74
2.2.7.4.2 Scaling Agile Best Practices 74
2.2.7.5 Government Agile Best Practices 75
2.2.7.6 Other Considerations 76
2.2.7.6.1 Capability Maturity Model Integration 76
2.2.7.6.2 Earned Value Management 77
2.2.7.6.3 Agile Coach 77
2.2.7.6.4 Agile Training 77
Chapter 3: Research Questions and Method 78
3.1 Research Questions 78
3.1.1 Research Question 1 78
3.1.2 Research Question 2 78
3.1.3 Research Question 3 78
3.2 Research Method: Action Research 78
3.3 Research Procedures 80
3.3.1 Selecting Client Organizations and Diagnosing 82
3.3.2 Action Planning 83
3.3.3 Action Taking 84

3.3.4 Evaluating by Collecting Data and Researching 85
3.3.5 Specifying Learning (and Refining ESD) 86
3.4 Project Schedule 86
3.5 Project Steps 87
3.6 Ethical Considerations 88
Chapter 4: Analysis and Findings 89
4.1 Overview 89
4.2 Refining ESD 89
4.3 Number of responses 92
4.4 Results from Project I 93
4.4.1 Project I Overview 93
4.4.2 Experience Interview (Project I) 94
4.4.3 ESD Agile Assessment and Agile BP Checklist (Project I) 99
4.4.4 Result Interview (Project I) 103
4.4.5 Observer's Note 104
4.3.5 Research Challenges 105
4.5 Results from Project II 106
4.5.1 Project II Overview 106
4.5.2 Experience Interview (Project II) 106
4.5.3 ESD Agile Assessment Form and BP Checklist 112
4.5.4 Result Interview 117
4.5.5 Observer's Note 118
4.3.5 Research Challenges 119
4.6 Testing of Research Questions 120
4.6.1 Research Question 1 120
4.6.2 Research Question 2 122
4.6.3 Research Question 3 125
Chapter 5: Conclusions and Recommendations 127
5.1 Conclusions 127
5.2 Synthesis of Research Questions 129
5.3 GAgile Objective 132
5.4 Recommendation for Future Work 132
Appendix A Interview Notes about the Participants' and Organization's Experiences and Expectations 142
Appendix B Agile Program Assessment Form and Best Practice Checklist 148
Appendix C Interview Notes about Participants' Experiences with ESD and Results of the Project 154
Appendix D Researcher's Observation Note 156

List of Figures

Figure 1 Manifesto for Agile Software Development 8

Figure 2 The Waterfall model 11

Figure 3 The Prototyping model 12

Figure 4 The RAD model 12

Figure 5 The Spiral model 13

Figure 6 The Rational Unified Process 14

Figure 7 The Rational Unified Process (Source: IBM) 14

Figure 8 Extreme Programming 16

Figure 9 XP practices [71] 17

Figure 10 Crystal framework for methodology selection [73] 18

Figure 11 ASD lifecycle [75] 19

Figure 12 Scrum process diagram [77] 20

Figure 13 The five processes of FDD [78] 20

Figure 14 DSDM process: The three pizza and a cheese [80] 21

Figure 15 Class diagram of software development methods 26

Figure 16 Three steps of ESD 29

Figure 17 ESD pentagon 31

Figure 18 Three aspects for successful software development project 35

Figure 19 Five major factors in software development 37

Figure 20 Lifecycle – Case Study A [32] 54

Figure 21 ESD Pentagon – Case Study A 57

Figure 22 ESD Agile Best Practice Checklist (version 4.0) 59

Figure 23 Scrum Process Diagram [77] 66
Figure 24 Sample Iteration Burn-down Chart 70
Figure 25 Sample Velocity Chart. 71
Figure 26 Sample Product Burn-down Chart 72
Figure 27 Sample Iteration Completion Trends Chart 73
Figure 28 Sample Iteration Task Growth Chart 73
Figure 29 Susman and Evered's AR Cycle [160] [149, 161] 81
Figure 30 Proposed schedule 86
Figure 31 Project steps 87
Figure 32 Experience Interview Results for Project I 94
Figure 33 Methodology Used for Previous Projects (Project I) 95
Figure 34 Methodology Used for Previous Government Projects (Project I) 95
Figure 35 Methodology Planning to Use for other Government Projects (Project I) 96
Figure 36 Methodology Planning to Actively Advocate (Project I) 97
Figure 37 Thematic content analysis for initial interview (Project I) 99
Figure 38 Agile Best Practices Checklist (Project I) 100
Figure 39 Agile program assessment form – rate (Project I) 101
Figure 40 Summative content analysis: Agile program assessment form - challenges/comments (Project I) 101
Figure 41 Summative content analysis: Agile program assessment form – Approach (Project I) 102
Figure 42 Summative content analysis: Agile program assessment form – Overall (Project I) 102
Figure 43 ESD Results (Project I) 103
Figure 44 Will you use ESD again? (Project I) 104

Figure 45 Will you actively advocate for ESD in the future? (Project I) 104
Figure 46 Experience Interview Results for Project II 107
Figure 47 Methodology Used for Previous Projects (Project II) 107
Figure 48 Methodology Used for Previous Government Projects (Project II) 108
Figure 49 Methodology Planning to Use for other Government Projects (Project II) 109
Figure 50 Methodology Planning to Actively Advocate (Project II) 109
Figure 51 Thematic content analysis for initial interview (Project II) 112
Figure 52 Agile Best Practices Checklist (Project II) 113
Figure 53 Agile Process based on Agile Best Practices Checklist Review (Project II) 113
Figure 54 Agile Technical Framework based on Agile Best Practices Checklist Review (Project II) 114
Figure 55 Agile program assessment form – rate (Project II) 115
Figure 56 Summative content analysis: Agile program assessment form - challenges/comments (Project II) 115
Figure 57 Summative content analysis: Agile program assessment form – Approach (Project II) 116
Figure 58 Summative content analysis: Agile program assessment form – Overall (Project II) 116
Figure 59 ESD Results (Project II) 117
Figure 60 Will you use ESD again? (Project II) 118
Figure 61 Will you actively advocate for ESD in the future? (Project II) 118
Figure 62 Experiences of ESD-like (Project I) 123
Figure 63 Experiences of ESD (Project I) 124
Figure 64 Experiences of ESD-like (Project II) 124
Figure 65 Experiences of ESD (Project II) 125

Figure 66 Agile Best Practice Checklist Template..... 149

List of Tables

Table 1 Differences between Private and Public Sector 4

Table 2 Characteristics of plan-driven and agile methods [31] 9

Table 3 Five critical factors in agile and plan-driven methods [31] 10

Table 4 Comparison between Waterfall and Agile [53] 10

Table 5 Overview of agile methods 15

Table 6 Limitations of agile methodology in government projects 24

Table 7 Leadership models in information systems/information technology 38

Table 8 Tools selection example 39

Table 9 Management and leadership [128] 44

Table 10 Summary - Case Study A [32] 52

Table 11 Agile Program Assessment Form 60

Table 12 Overview of Other Agile Methods 67

Table 14 Target projects 83

Table 15 Refining ESD 90

Table 16 Number of responses 92

Table 17 Summative content analysis for initial interview (Project I) 97

Table 18 Summative content analysis for initial interview (Project II) 110

Table 19 Testing of Research Question 1 120

Table 20 Testing of Research Questions 2 122

Table 21 Testing of Research Question 3 125

Table 22 Agile Assessment Form 149

Chapter 1: Introduction

1.1 Problem Statement

Over the last 10 years the private sector has shown considerable interest in agile software development methods, and the government sector, including the U.S Army[1], the Federal Bureau of Investigation[2], the Department of Defense[3, 4], the Central Intelligence Agency[5], the Environmental Protection Agency[6], the German government[7], the City of Calgary, Canada[8], and other government organizations[9-12] are experiencing with the new approach. However, agile methods present limitations for the environment of supporting government software development. The Eclectic Software Development (ESD) approach provides a framework including data collection tools and analysis processes that capitalizes on the benefits of agile and traditional methodologies for the government sector while minimizing the limitations of both. The theory behind ESD is the selective use of the right tools, methodologies, processes, and human resources by project leadership at the right time, within the confines and structures already defined for large-scale and contract-based government projects. This study used action research to apply ESD practices to real government projects in a cyclic manner to validate and refine ESD.

Eclectic approaches are currently being utilized in various fields [6, 13-19], but they are individual projects lacking the formal validation the government sector demands. Using action research, this study examined the following research questions. This study focused on balancing agile and plan-driven methodology in the government software development community.

- Research Question 1. Is it possible to formalize the eclectic approach so that it can be adopted by projects in the same way the traditional approach has been used?
- Research Question 2: Is ESD well accepted by new practitioners?
- Research Question3: Does this ESD pentagon represent an acceptable management tool? If not what criteria would be needed to make it one?

1.2 Background

1.2.1 Government Software Development Challenge Overview

The U.S. government's FY 2006 Information Technology (IT) portfolio was $65 billion [20], making the U.S. government the largest IT investor in the world. The George W. Bush administration monitored the performance of IT development projects using a management watch list. This watch list included 342 of 1,087 projects valued at $15 billion, representing more than 30% of the total FY 2006 portfolio. The FY 08 actual was $72 billion, the FY 09 budget was $74 billion, and the FY 10 budget was $78 billion [21]. In June, 2009, the Barack Obama administration launched the IT dashboard, http://it.usaspending.gov, which allows people to see how the federal government is spending taxpayer dollars on its IT portfolio.

Both public and private sectors experience difficulties successfully delivering software on time and within budget with the required functionalities. When a large government project fails, the impact is huge. Recent failed U.S government projects include the IRS Business Systems Modernization project, which was delayed for five years and was over budget by $2 billion; the FBI Virtual Case File system, which spent $170 million, only to be scrapped and reinitiated; and the Transportation Security Administration CAPPS II project, which was abandoned after delays and privacy concerns surfaced [22].

The Veterans Affairs (VA) Replacement Scheduling Application, which was to be deployed in January 2010, was terminated in March 2009 [23]. The Census Bureau cancelled the wireless data collection project after the contractor had already been paid $236 million [24]. Governments of other nations have faced similar problems [25-29].

1.2.2 Current Situation – Agile Adoption

The private sector software industry has explored and come to appreciate agile software development methodology. The proven history and reputation of agile methods in the private sector have drawn the interest of the government [1-12]. For example, CrossTalk, an approved Department of Defense (DoD) software engineering journal, had "Agile Software Development" as a theme for the October 2002 edition and "Agile Development" as a theme for the December 2006 edition.

Many governments have introduced various initiatives to improve the success rate of their IT projects [30]. For example, to train qualified IT project managers, the U.S government initiated the IT Exchange Program [20], which permits government IT managers to work temporarily in the private sector, exposing them to cutting-edge management and technical trends, including agile methodology.

Even though agile methodology was originally developed for small projects, project managers in large-scale projects want to inject needed flexibility and strive for continuous process improvement by balancing agile and plan-driven methodology [2, 3, 31-34]. Recently, agile methodology, in both authentic and hybrid styles, has been adopted by some government software development projects [1-6, 8-12, 32, 35-37]. However, some organizations face challenges when migrating from traditional to agile methodology because there are conceptual differences between these two methodologies [6, 38].

1.2.3 Relevant Research

In general, after a new alternative model is introduced, the software engineering industry compares the new model to classic models with respect to meeting user needs [39-41]. In the late 1980s, new alternative models included prototyping and incremental development models. In the 2000s, it is agile methodology.

The public sector operates IT projects differently from the private sector and faces unique challenges [42-44]. Specifically, applying agile methodology to a government project is more challenging. As shown in Table 1[45], the United Kingdom Computing Service and Software Association (CSSA) compares private and public sector IT projects.

Table 1 Differences between Private and Public Sector

Sector	Differences
Private Sector	measurable outcomes-driven
	competition-driven
	less visible to the public
	less regulated
	objective-driven (risks taking)
	designed to minimize impact of failed projects
Public Sector	complex success factors
	less competition with other projects

Sector	Differences

	Inter-agencies interaction
	• References
	  o Data.gov
	  o USAspending.gov
	  o Recovery.gov
	  o Regulations.gov
	  o FirstGov [46], now USA.gov
	  o Integrated Acquisition Environment (IAE) [47]
	  o Business Gateway [47]
	  o Other examples [42, 46]

	high visible to the public
	Less adaptive [43]

	regulated
	• References
	  o E-Travel [47]
	  o Others examples [42, 48]

	risk-averse culture

1.3 Purpose

This study focuses on two characteristics (shown in bold in Table 1) related to software development methodology: Inter-agencies interaction and Less adaptive.

First, government projects require both technical interfaces and political collaboration with systems from various departments, services, or agencies. The selected methodology must harmonize with methodologies from other departments. These are often legacy systems that relate to regulations and policies.

Data.gov, USAspending.gov, Recovery.gov, and Regulations.gov are great examples. FirstGov.gov project, renamed to USA.gov, linked 47 million federal government web

pages when this collaborative project was launched in 2000 [46], and the Integrated Acquisitions Environment (IAE) initiative for the U.S. government provided an interoperable acquisition gateway with other agencies [47]. Furthermore, the administration established the Federal Enterprise Architecture (FEA) Program for improving the effectiveness of an agency's IT management by maximizing government-wide service providers, which requires more collaboration than ever before.

Second, a government project does not easily adapt to change. It usually requires a well-defined, planned, controlled, auditable, and tested project. The government often has a set of predefined processes that contractors must follow and tailor the project to fit. In addition, subcontractors must comply with their prime contractor's defined process or methodology.

1.4 Significance

Eclectic Software Development was originally developed in 2004 to support one of U.S. DoD software development projects. The methodology was employed in the successful development of an e-government web application within the Software Engineering Institute's (SEI) Capability Maturity Model Integration (CMMI) Level 3 process [32]. Since then, it has been applied to additional projects with continuous refinement. ESD provides a pragmatic, instead of a dogmatic, framework for government software development projects. ESD involves the selective use of the right tools, methodologies, processes, and human resources by project leadership at the right time, within the confines and structures of the processes and procedures already defined for large-scale and contract-based government projects.

1.5 Scope

This research implemented ESD for selected, real government projects in a cyclic manner. Each cycle involves data collection through a combination of interviews, interpretation, and literature review. Lessons learned from participants in one project have been examined and applied to the next project in order to effectuate change. Action research consists of actions invoking change and research which increases understanding [49]. It was proposed because this study demands a cyclic, responsive, and participative approach. It combined theory and practice in real situations.

1.6 Organization of the Document

This chapter introduces the research project, including problem statement, background, purpose, significances, and scope. Chapter 2 presents a literature review including a summary of popular software lifecycle methodologies, and a review of ESD. In Chapter 3, the research questions are presented, and in Chapter 4 the research methodology are presented. Chapter 5 presents conclusions and recommendations followed by appendices which enclose various data collection forms that were used.

Chapter 2: Literature Review

2.1 Software Development Methodology

2.1.1. Overview

This section 2.1 describes popular software development methods of the plan-driven and agile methodologies. Plan-driven methods consist of sequential, well-defined processes such as requirements identification and design specification [31]. Plan-driven versus agile methods are compared as 'process-oriented', 'predictive', and 'heavyweight' method versus 'people-oriented', 'adaptive', and 'lightweight' method [50, 51].

Practitioners and authors of agile methods and practices produced the "Manifesto for Agile Software Development" in 2001 as illustrated in Figure 1. It is available at www.agilemanifesto.org [52].

Individuals and interactions	over	processes and tools
Working software		comprehensive documentation
Customer collaboration		contract negotiation
Responding to change		following a plan

Figure 1 Manifesto for Agile Software Development

The characteristics of plan-driven and agile methods were compared by Boehm and Turner [31] as shown in Table 2. The major difference is the plan-driven method is characterized by a culture of clear policies and procedures and the agile method is marked by a culture of a high degree of freedom of implementing change or deviation from clear-cut policies and procedures. The plan-driven method shares many common characteristics with government projects. In general, government projects demand clear policies and procedures and also provide a low degree of freedom. The government software development community, however, is increasingly demanding efficient responsiveness to

changes. Boehm and Turner recommend selecting the appropriate method based on the five critical factors shown in Table 3. Agile methodologies are recommended for small, low-critical, and highly dynamic projects, while plan-driven methodologies are recommended for large, mission critical and less dynamic projects.

Table 2 Characteristics of plan-driven and agile methods [31]

Characteristics		Plan-Driven	Agile
Application	Primary project goals	• Predictability • Stability • High quality	• Immediate value • Responsiveness to changes
	Project size	• Large	• Small
	Application environment	• Stable • Low change • Project/organization focused	• High-change (Does not mean welcomes last-minute change) • In-house • Flexible user system • Project focused
Management	Customer relations	• Contract • Build Trust: Process maturity	• Dedicated on-site customer • Reflects the needs and desires of the users • Build Trust: Working software and customer participation
	Planning and control	• Quantitative plan	• Qualitative plan
	Project communications	• Explicit, documented knowledge	• Tacit knowledge • Person-to-person and frequent communication
Technical	Requirements	• Specific, formalized requirements	• Adjustable, informal stories
	Development	• Extensive design	• Simple design
	Test	• Test to specifications	• Develop test before code
Personnel	Customer		• Collaborative
	Developer	• Smaller percentage of talented people	• Richer mix of higher-skilled people

Characteristics		Plan-Driven	Agile
	Culture	• Clear policies and procedures	• Many degrees of freedom • Craftsman

Table 3 Five critical factors in plan-driven and agile methods [31]

Critical Factor		Plan-Driven	Agile
Size	Small	Hard to tailor down	Well matched
	Large – Government Project	Evolved to handle	Limit scalability
Criticality	Low	Hard to tailor down	
	High – Government Project	Evolved to handle	Untested. Potential difficulties
Dynamism	Low – Government Project	Excellent	Potentially expensive rework
	High	Potentially expensive rework	Excellent
Personnel	High-skilled people	Need during definition phase, but need fewer later	Continuously need
Culture	Freedom		Well matched
	Clear policies and procedures – Government Project	Well matched	

Jim Highsmith presented the following table [53] during his presentation at the Agile 2009 conference to compare Waterfall and agile.

Table 4 Comparison between Waterfall and Agile [53]

Area	Waterfall	Agile
Performance Philosophy	Conformance to Plan	Adapt to Change
Performance Measure	Scope, Schedule, Cost	Value, Quality, Constraints
Product/Project Focus	Project	Product
Delivery	Project End	Continuous/Incremental
Organization	Functional Teams	Feature Teams (Cross-functional)
Planning	Task-based, Detailed Network Diagrams	Feature-based, timeboxed iterations

Area	Waterfall	Agile
Management Culture	Command-and-Control	Leadership-and-Collaboration
Team Culture	Manager Controls	Self-organizing
Architecture/Requirements	Big up-front	Evolutionary

2.1.2 Plan-Driven Methodology

2.1.2.1 Plan-Driven Methodology Overview

The popular plan-driven method includes Waterfall, Spiral, Rational Unified Process (RUP), Prototyping, Rapid Application Development (RAD) models. Some of the contents in section 2.1.2 Plan-Driven Methodology and 2.1.3 Agile Software Development are excerpts from the author's previous paper [32]. Figures 2, 3, 4, 5, and 6 compare development time and efforts among these plan-driven methodologies.

2.1.2.2 The Waterfall Model

This model consists of serialized development phases [54]. Royce is credited by software engineering textbooks as the author of this oldest and most widely used model [55, 56]. However, Cantor argues that the traditional Waterfall approach is "one of three common, but inadequate, approaches" due to uncertainty in the progress [57].

One of the classic weaknesses of the Waterfall model is that it requires a complete list of requirements at the beginning [40] and is less effective for environments that require quick responses to changes. Figure 2 illustrates serialization of software development activities throughout the development life cycle.

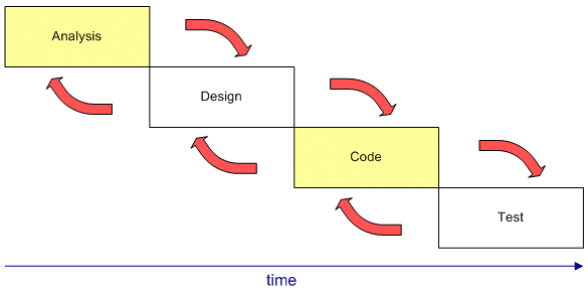


Figure 2 The Waterfall model

2.1.2.3 The Prototyping Model

This model is based on developing throwaway or evolutionary prototypes[55]. This model is widely adopted for the development of user interface intensive applications. Figure 3 shows a series of prototypes throughout the development life cycle.

Figure 3 The Prototyping model

Schrage suggests the prototyping partnership between developers and clients [58]. According to Schrage, requirement changes are inevitable. The developers and clients learn from each other and working together to accomplish change. His idea of the prototyping partnership is in accordance with some agile principles.

2.1.2.4 Rapid Application Development Model

In the Rapid Application Development (RAD) model, a SWAT (Skilled With Advanced Tools) team develops an application within a short defined development time, marked by user involvement. The team consolidates the design by Joint Application Design (JAD) sessions, evaluates using prototypes, and builds using reusable components such as CASE tools [59]. Figure 4 compares the length of development period in RAD and other typical development.

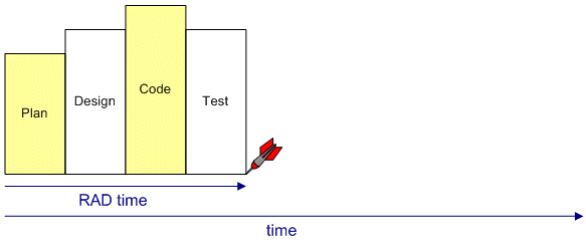


Figure 4 The RAD model

Experiences with RAD led to initial versions of frameworks supporting other agile methods such as Adaptive Software Development (ASD) and Dynamic Systems Development Method (DSDM)[60, 61].

2.1.2.5 Spiral Model

The Spiral model involves incremental cycles. This model can be applied throughout the life of the software. It works better with flexible environment such as internal software development [62]. One of the classic weaknesses of Spiral model is that it can face the death spiral [63]. This is when the project repeats the spiral without knowing when the project can declare a successful end. A check mark indicates an end of each cycle in Figure 5.

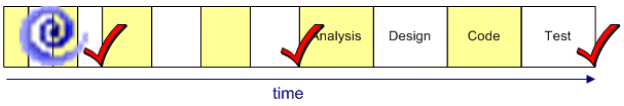


Figure 5 The Spiral model

An extension of the Spiral model is the WinWin Spiral model which adds the process of negotiation with stakeholders (which applies Theory W) at the start of every cycle [64]. A software manager is a 'negotiator' in Theory W," while other management theories such as Theory X, Y, and Z consider a manager as different characters. [65].

2.1.2.6 Rational Unified Process

A product from Rational Software, Rational Unified Process (RUP), captures the software development best practices, including controlled iterative development, requirement management, component-based development, modeling with the Unified Modeling Language (UML), quality assurance, and change management [66]. This process has four phases, and each phase consists of different weights of iterative development

activities as illustrated in Figure 7. These development activities overlap at given times, as Figure 6 illustrates.

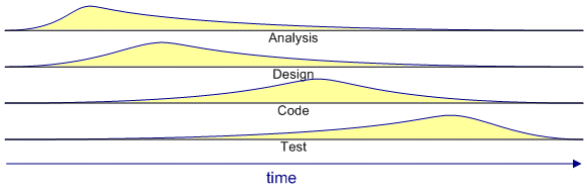


Figure 6 The Rational Unified Process

The phases and activities of RUP are illustrated in Figure 7. The four phases are

- Inception: Bring forth the idea or RFP into the Elaboration phase.
- Elaboration: Define the product vision and its architecture.
- Construction: Produce a product from an executable architectural baseline.
- Transition: Deliver the product to the hands of the user's community.

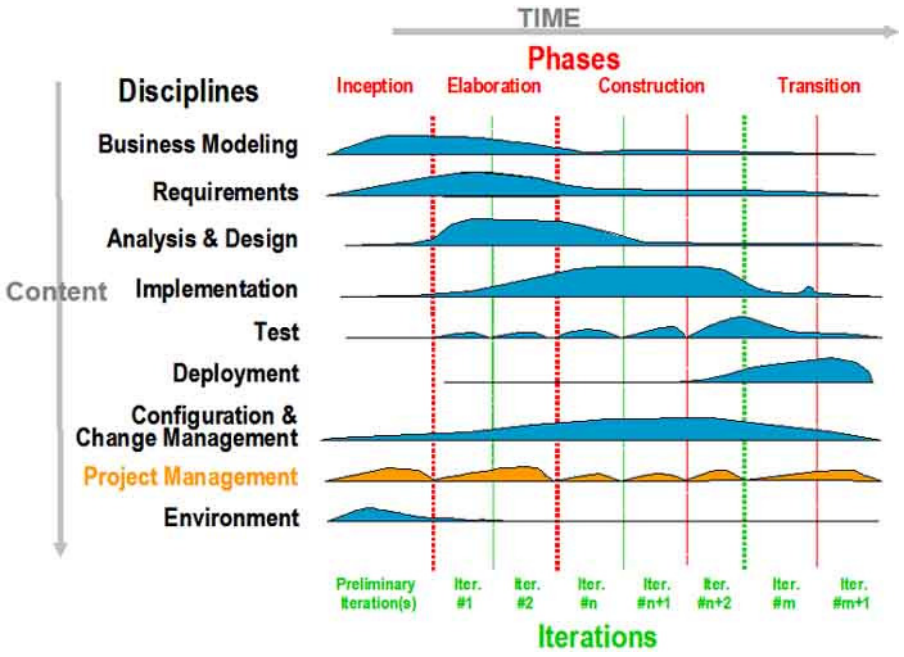


Figure 7 The Rational Unified Process (Source: IBM)

2.1.3 Agile Software Development

2.1.3.1 Agile Software Development Overview

Various methodologies and practices have been developed to adopt continuous changes in software development seamlessly. These include XP, Crystal, ADP, and SCRUM in the US; FDD in Australia; and DSDM in Europe [67]. Table 5 provides an overview of the methodologies.

Table 5 Overview of agile methods

Agile Methods	Overview
XP	Plan, design, develop, test, and release in short development cycles throughout the development lifecycle by a small team with customer involvement.
Crystal	A framework for software development methodology selection based on staff size, system criticality, and project priority
ADP	Accepts continuous changes by continuous adaptations
Scrum	Like a team of eight players in rugby, a small development team, with no more than 10 team members, acts together with a well-defined role on a single goal for each increment
FDD	Features object-oriented based development of components with collaboration between domain experts and programmers.
DSDM	Project delivery framework with iterative and incremental steps, which takes less time and discovers and corrects problems earlier than the waterfall method.

These agile methodologies have been widely explored and utilized by the software industry. The Agile Alliance site (http://www.agilealliance.org/) is a prime source of information concerning this methodology. This section provides an overview of popular agile methods.

Twelve principles [68] are available at www.agilemanifesto.org. Agile methodologies are change-driven, customer-oriented, people-oriented, and result-oriented. They are less complex and rigorous than plan-driven, contract-based, process-oriented, and design-oriented methodologies.

2.1.3.2 Extreme Programming (XP)

Kent Beck is the leader of Extreme Programming (XP). XP programmers continually plan, design, develop, test, and release in short development cycles throughout the development lifecycle. Figure 8 demonstrates the small and continuous development activities using small squares.

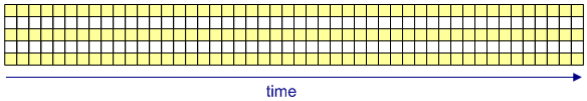


Figure 8 Extreme Programming

When the C3 project team invented the "eXtreme Programming" methodology at Chrysler [69], they broke away from the methodology of the day. Figure 9 provides the following key practices of XP [70-72].

- Planning game: Predicting accomplishment for the given schedule rather than predicting schedule for the given scope.
- Small releases: Releasing tested software every iteration and on a frequent basis
- Metaphor: Developing a concept of understanding
- Simple design: Building software with simple designs throughout the lifecycle
- Test-driven development: Building test cases and then coding instead of coding and then building test cases

- Refactoring: Refactoring to improve the design of existing code
- Pair programming: Pairing two programmers to collaborate
- Continuous integration: Fully integrating code changes at all times
- Collective code ownership: Improving any code at any time by any pair of programmers
- No overtime
- On-site customer: Having a customer available whenever they are needed to answer questions and to provide direction

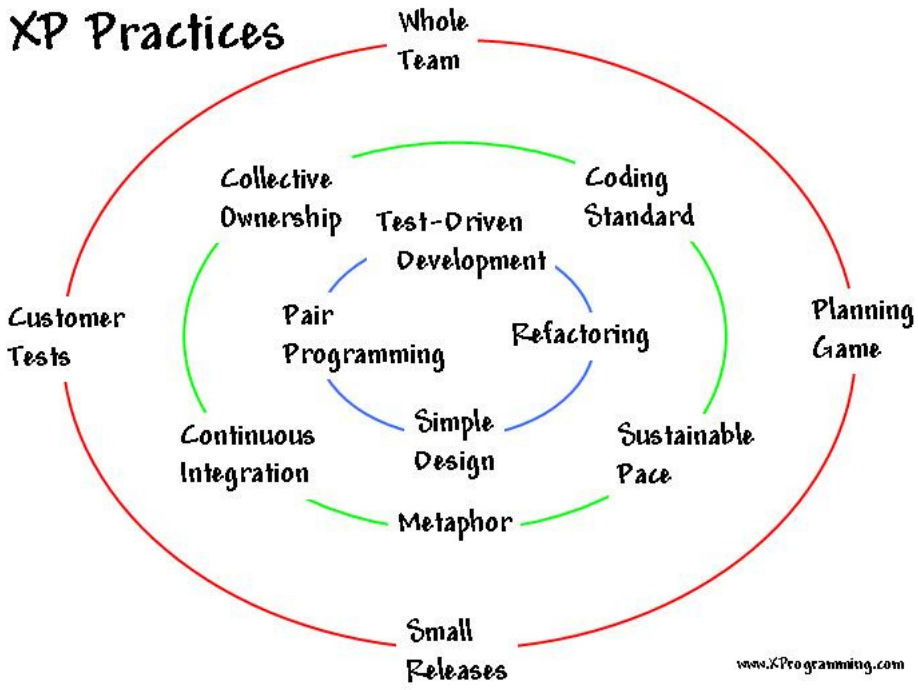


Figure 9 XP practices [71]

2.1.3.3 Crystal

Alistair Cockburn, the leader of Crystal, proposes a framework for software development methodology selection based on project priority, system criticality, and staff

size [51]. His idea is that no one methodology fits all projects. Figure 10 is a framework of Crystal [73]. The horizontal axis indicates the staff size. The figure shows that a project needs more communication coordination as the staff size increases. The vertical axis indicates system criticality based on the potential damage from poor quality. The project needs more validation practices as the system criticality increases. The different planes indicate different project priority, such as productivity and legal liability. For example, E40 means a project with 20 to 40 people with potential loss of essential money. When this project is changed to L100, it requires a different methodology to support more communication coordination and more validation practices.

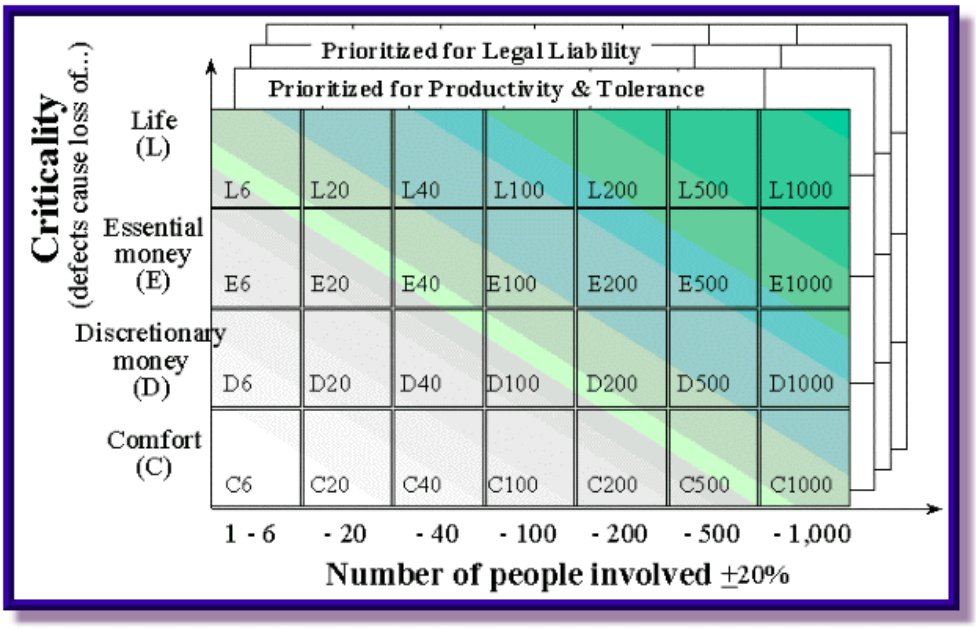


Figure 10 Crystal framework for methodology selection [73]

2.1.3.4 Adaptive Software Development (ASD)

Jim Highsmith is a founder of ASD, which was evolved from Radical Software Development [60], a iterative and short RAD for large organizations. ASD involves

accepting continuous changes by continuous adaptations with a “Speculate -Collaborate-Learn” lifecycle, which replaces the “Plan-Design-Build” lifecycle [74].

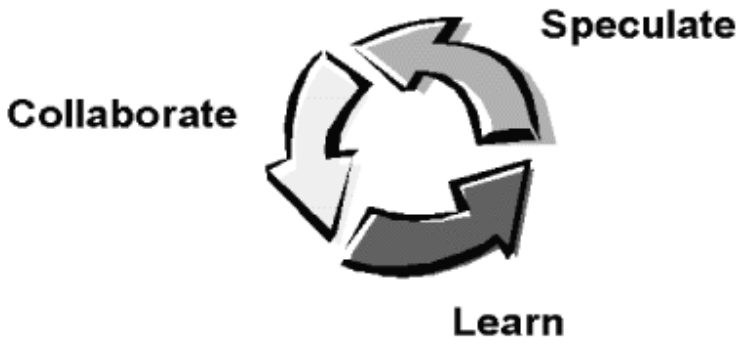


Figure 11 ASD lifecycle [75]

Ken Schwaber, Mike Beedle, and Jeff Sutherland are key contributors to Scrum. Like a team of eight players in rugby, a small development team with no more than 10 team members in Scrum acts together with a well-defined role on a single goal for each increment [76]. A small team works on a series of “Sprint” (iteration) based on “Backlog” (a list of tasks) with “Scrum” meetings (short daily meetings). During a 15-to-30-minute Scrum meeting, the team discusses its accomplishments since the last meeting, obstacles, and plans from that time until the next meeting. Figure 12 illustrates the Scrum process flow.

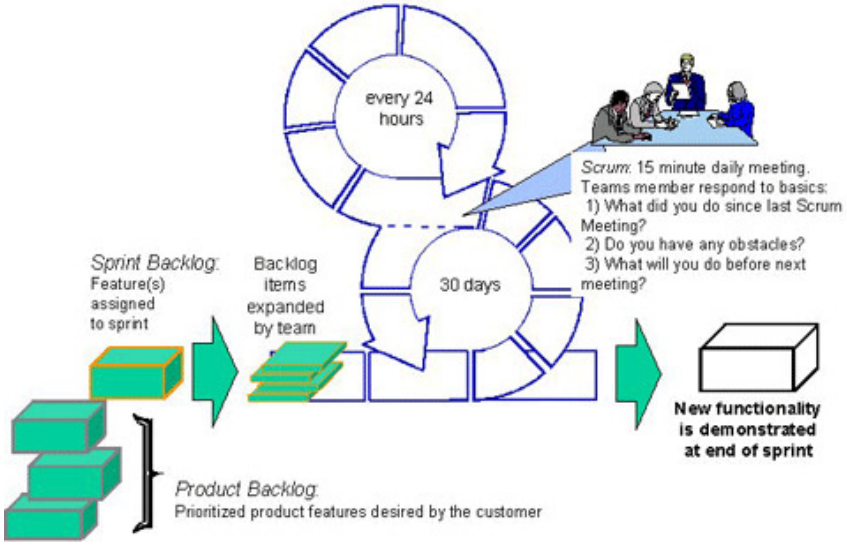


Figure 12 Scrum process diagram [77]

2.1.3.6 Feature-Driven Development (FDD)

Jeff De Luca and Peter Coad founded this method. Figure 13 illustrates its five processes [78]. These five processes are located between initial requirements and system test because this method considers the core problem area in software development to be between those two activities. However, other activities can work with the core FDD process.

Each iteration of design and build is no longer than two weeks. Unlike the plan-driven method with long design and build phases, FDD supports projects with rapid business requirement changes. FDD also requires collaboration between domain experts and programmers.

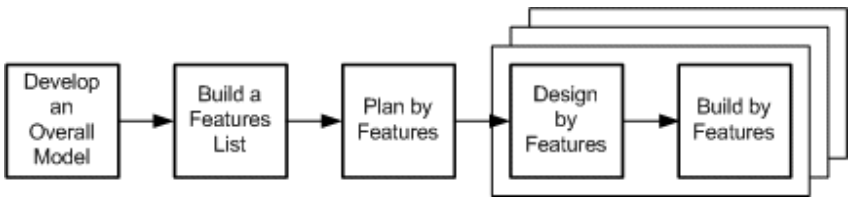


Figure 13 The five processes of FDD [78]

2.1.3.7 Dynamic Systems Development Method (DSDM)

The DSDM Consortium in the UK publishes this project delivery framework as shown in Figure 14. This incremental and iterative framework consists of seven phases [79]. Compared to Waterfall, it takes less time and recovers from problems early.

Figure 14 DSDM process: The three pizza and a cheese [80]

2.1.4 Bridging Agile and Plan-Driven Development Methods

2.1.4.1 Applying Agility in Large Projects

According to Boehm, an author of the spiral model, and Turner, plan-driven methodology has a culture of clear policies and procedures and agile methodology has a culture of a high degree of freedom [31]. Fowler suggests traditional methodologies for projects with large team (over hundred) or fixed price projects [50].

Highsmith states that "Chaos is easy – just do it. Stability is easy – just follow steps. Balancing is hard – it requires enormous managerial and leadership talent" and emphasizes leadership in ASD [74].

The growing deployment of agile methodologies has resulted in many articles about applying agile methods to large projects. Some large organizations conducted pilot projects to evaluate agile methods for timely delivery of product and flexibility but without compromising the organization's high quality standards [17].

More than other agile methods, XP initially had been successfully proposed and applied to large projects by choosing selected practices in the existing process [17, 18], modifying the rules [81-83], evolving the practices [84], adding new practices [85-87], or reorganizing companies [88]. XP was originally designed for projects involving fewer than 10 people. Because many successful development projects have given credit to the effectiveness of XP, more large and complex projects employ XP-influenced approaches knowing that for them using XP directly is not recommended. However, to take advantage of XP in large projects, the participants must rigorously apply the method [81].

Enterprise Agile and Scaling Agile are now widely accepted concepts [89-91]. At the Agile 2009 Conference, many presenters from large corporations including Borland [92, 93], HP [94], Qualcomm [94], BMC [94], MySpace [95], Microsoft [96, 97], Amazon.com [98], Marriott [99], SIEMENS [100], IBM [101], Lockheed Martin [6], Yahoo [102], Google [102], and Disney [103] shared their agile stories.

Industry leaders agree that the agile method can work with the existing methods for large projects. Cockburn, an author of the Crystal method, proposes that plan-driven projects can benefit from applying agile values [73]. Paulk, at the Software Engineering Institute, says that XP is compatible with the Capability Maturity Model (CMM) [104]. He states that XP and CMM can create synergy when XP focuses on engineering aspects and CMM focuses on management aspects.

At the same time, many companies have enhanced their corporate-wide standard software development process methodology to gain agility. Increasingly, each individual project uses the standard process with some level of flexibility [105].

2.1.4.2 Applying Agility in the Telecommunications Industry

According to the experiences of pilot projects at Motorola and Nokia, extensive tailoring was required to introduce XP in their organization. Motorola received consistent and positive results from piloting tailored XP for four complex mission critical systems [83], which consisted of 29 engineers operating over an 18-month period. Compared to the "on-site customer" practice of XP, these pilot projects hired an experienced coach and their technical domain expert acted as a customer because the real customer was not available. Compared to the "small design" practice of XP, initial architectures were defined before the first iteration because these projects were part of a large system release. These pilot projects used design reviews and requirement verification reviews not only to ensure maintainability but also to meet XP's minimal document rule. Overall, the results from the four pilot projects included positive morale, reduced learning curve, high productivity, more test coverage, and comparable quality and maintainability.

Nokia's several in-house methods are considered to be agile methods for large organizations [86]. These methods utilize facilitated cross-team workshops with a Community of Practice (CoP) theory to overcome the limitation of the agile method's team concept. This team concept is deemed insufficient for large or multi-team organizations.

2.1.4.3 Applying Agility in Government Projects

In the 2.2 Eclectic Software Development section, this proposal presents a framework for balancing between plan-driven and agile methodologies for government software development projects. Agile methodology is introduced into the process and management

of government contract projects [4, 35]. For example, the German government has accepted agile strategy as one way to run projects with V-Model XT, a new official process model [7], and the US government has published Request For Proposals (RFP) with requirements to use Agile method. On the other hand, some studies discuss the challenges of migrating from traditional to agile methodologies [38].

McMahon [106] identified the four conflicts and five recommendations from observations of a prime contractor using a traditional method and a subcontractor using an agile methodology in a government project.

Tuck, France, and Rumpe [107] state that agile methodologies have limited support for large team, distributed team, subcontracting, large and complex software, and safety-critical software. Most of those limitations are characteristics of government projects, as shown in Table 6.

Table 6 Limitations of agile methodology in government projects

	Characteristics of Government Project (See Table 1, Extracted from [45]	Home Ground of Agile (See Table 2 and 3, Extracted from [31]	Limitations of Agile (Exacted from [107])
Team	Interactions with other departments	High degree of freedom	Distributed development environments
			Subcontracting
			Large team
Product	Difficulties to adapt to change because of scale and complexity	Potential difficulties for high criticality projects	Developing safety-critical software

	Characteristics of Government Project (See Table 1, Extracted from [45]	Home Ground of Agile (See Table 2 and 3, Extracted from [31]	Limitations of Agile (Exacted from [107])
		Limit scalability for large project	Developing large, complex software

This section 2.1 Software Development Methodology describes and compares characteristics of popular plan-driven and agile software development methodologies. It also shows that large projects, including those in the telecommunications and government fields, increasingly consider agile methodology to take advantage of the effectiveness of the time of delivery. However, some limitations of agile methodology are the nature of government projects. Overall, the software development industry is depicted in Figure 15.

The next chapter, 2.2 Eclectic Software Development, will present an approach to utilizing the effectiveness and reducing the limitations of agile methodologies for the government software development environment.

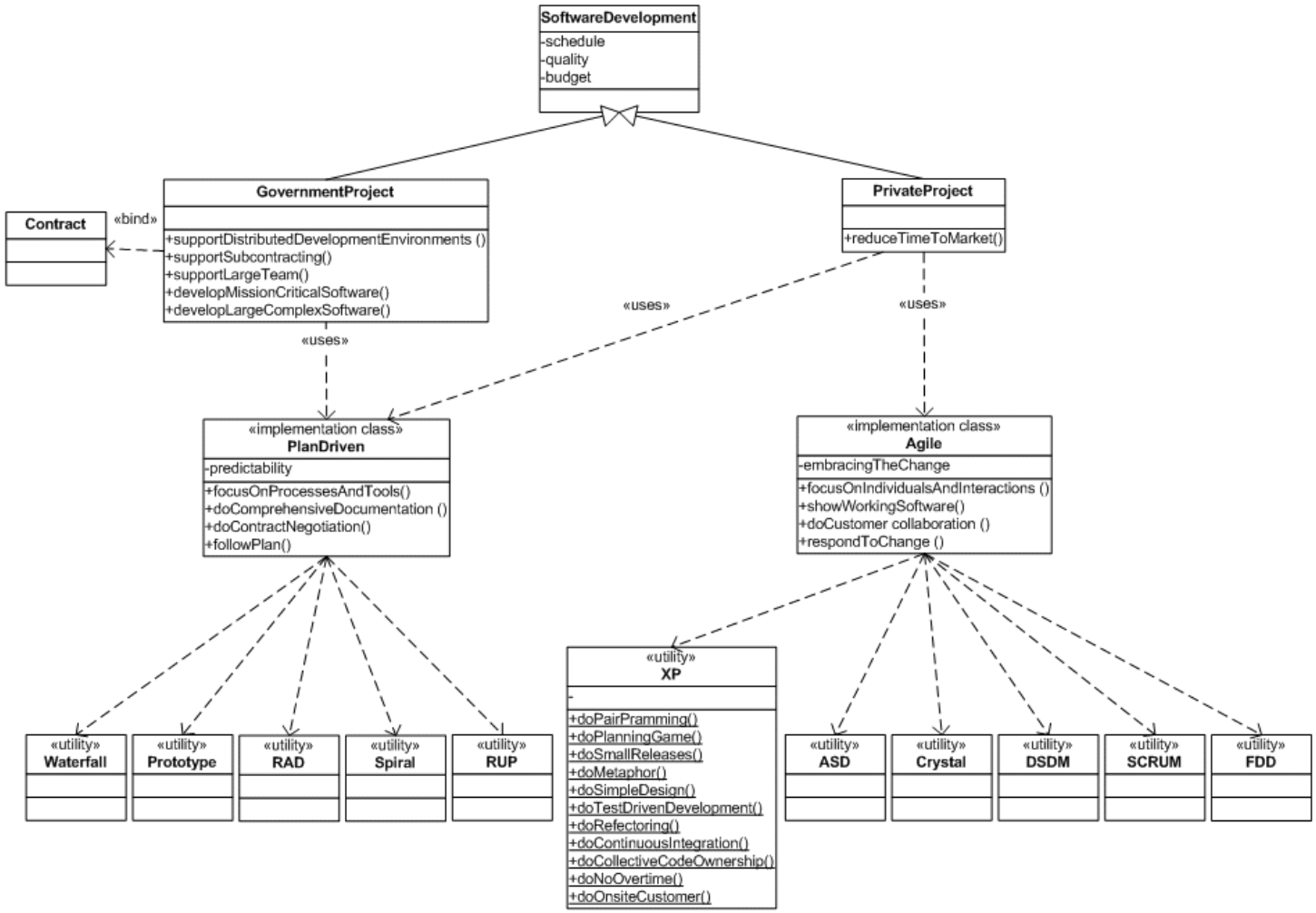


Figure 15 Class diagram of software development methods

2.2 Eclectic Software Development

2.2.1 ESD Background

When project leaders of large-scale government software development projects select, modify, or utilize any agile methodology or its practices, the ESD framework can serve as a guide to balance between the agile and traditional methodologies within the confines and structures of the tools, processes, methodologies, people, and leadership already defined for the government projects.

ESD was initially developed in 2004 by a software development team at Northrop Grumman Mission Systems for U.S. DoD projects. It is not a proprietary methodology. It has been peer-reviewed [32] and well received at an international conference and at Northrop Grumman [108]. Also, preliminary results from additional government projects are promising. ESD was designed to support government projects even though it has been applied in commercial projects. Additional studies are required to validate its effectiveness in types of large-scale projects other than government projects.

This section provides an overview of ESD. ESD includes some general principles and philosophies. It makes a valuable contribution as a useful guideline and reference throughout the software development lifecycle. This section has been validated and refined during this study.

2.2.2 ESD Executive Summary

The theory behind ESD is the selective use of the right tools, methodologies, processes, and human resources by project leadership at the right time within the confines and structures already defined for large-scale and contract-based government projects. Characteristics of government projects are as follows:

Government software development projects require both technical interfaces and political collaboration with systems from different departments, services, or agencies [45]. The selected methodology must harmonize with methodologies from the other departments. The systems are often legacy systems and involve regulations and policies. This harmonization characteristic is getting more attention because of current trends in technology, such as net-centric, service-oriented, web services, and federal enterprise architectures.

Government software development projects do not easily adapt to change. They usually require a well-defined, planned, controlled, auditable, and tested project. The government often has a set of predefined processes that the contractors must follow and tailor. Change is often not embraced, especially, when each module (i.e., web services provider or requester module) or task (i.e., requirement, development, or test) is delivered from different contractors.

ESD has been enhanced to support the characteristics of government projects noted above. When government projects consider any agile methodology or its practices to inject needed flexibility, they do so to balance agile and plan-driven methodology and to perform continuous process improvement. In such cases, ESD provides a pragmatic, not dogmatic, framework. The following diagram provides a high level overview involving three steps:

Figure 16 Three steps of ESD

Step 1: Recognizing organization standard

Step 2: Assessing project factors

- Tools
- Methodologies
- Process
- People
- Leadership: Visionary, Technological, Functional, and Managerial

Step 3: Recognizing and responding to project circumstances

- Continuous integrations and milestone demonstrations: Gain a true picture of the status, and open communication
- Ask simple questions
- Do simple mathematics: Often miss the forest for the trees

- Adopt management by walking around

Figure 17 zooms in on the center pentagon shape in Figure 16 and serves as a template to evaluate the project. Using the ESD Pentagon, each agency, department, or contractor assesses the project factors from its point of view. Within the same team, project leader, technical team leader, and developers also can assess the project factors. Boehm and Turner provide a framework to understand a project's characteristics in order to select either plan-driven or agile Software Development Life Cycle (SDLC) methodologies. The objective of ESD is to provide a better framework for understanding a project's characteristics and thereby balance between plan-driven and agile SDLC methodologies. During the assessing of the project factors, the ESD's goal is to make the pentagon balanced. Utilizing agile methodology and its best practices is not recommended unless the project is balanced for the five factors among the different stakeholder groups. The sub-factors can be added or removed by the practitioner. Also, critical success factors and proposed practices can be collected at the same time.

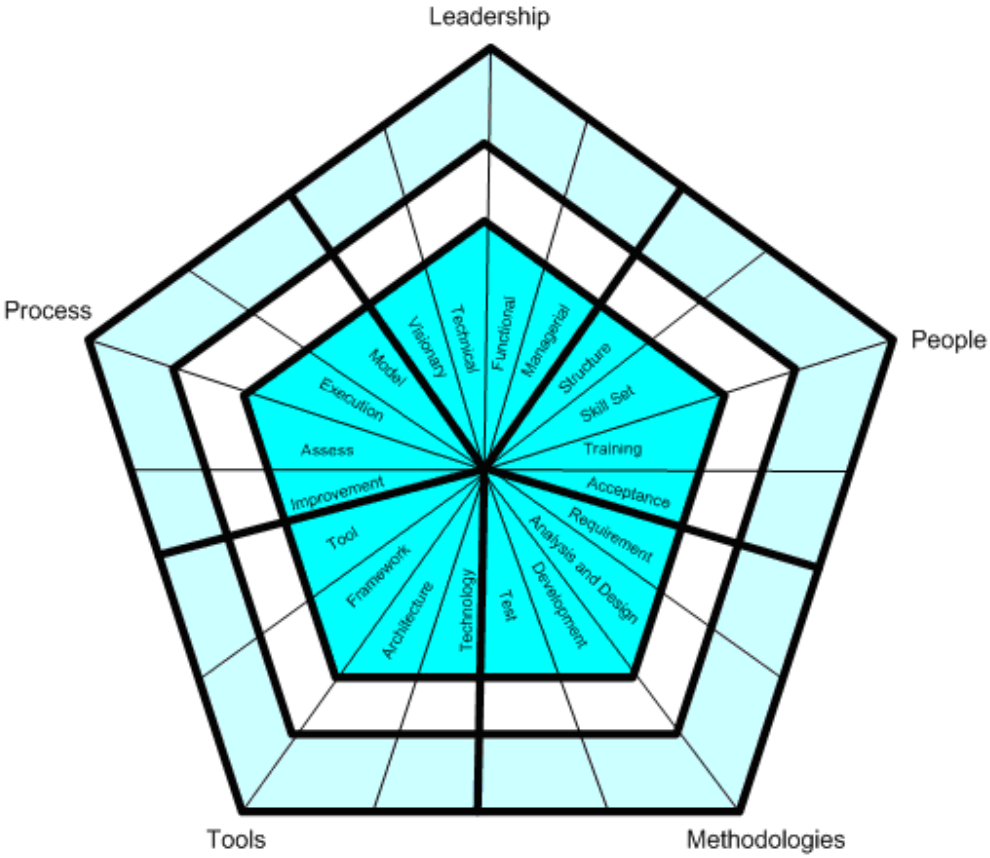


Figure 17 ESD pentagon

Based on the preliminary results before this study, ESD

- Helps to inject needed flexibility
- Provides continuous process improvement
- Ensures success throughout the lifecycle
- Is easy to learn and implement
- Works well with existing processes and methodologies

2.2.3 Philosophy of ESD

In her article "How to Read a Business Book," [109], J. Reingold states that managers view business how-to books as a generalized approach rather than a specific solution for their company. This suggests that a business manager needs to understand the key ideas

from various business books and add them to their own toolbox like a good auto mechanic. Software development managers can apply this idea to treat software processes and methodologies in a similar fashion.

The ESD approach introduces a concept of using the right processes, methodologies, and tools promptly [32]. In order for ESD to be effective, the team requires a project leader, who can provide an overall vision of the project's direction as well as technical, functional, and managerial leadership. The leader and team members need to recognize the problem, and apply the proper methodology. They should avoid dogmatically adopting the popular methodologies, hoping to find the silver bullet that will reward them with a productivity improvement of magnitude.

The concept behind ESD is not new. Many projects might be already utilizing an unstructured ESD approach without being aware of it. By choosing the combination of methodologies that work best for projects, the team is applying a set of general principles or philosophies of ESD. Software development engineers and managers often select and use different portions of existing methodologies. This way of thinking about software development methodology can be found in the previous studies reviewed below. Each one presents slightly different approaches but shares the same philosophy and vision of ESD. The most interesting fact is that agile methods, especially XP, foster the ESD idea because they provide a set of best practices from which to select.

- *Situational Method Engineering (SME)*: assembles *method fragments* into situational methods to the project at hand utilizing a rich repository [14, 19]
    - *Dual-Agility method*: uses *method engineering* to construct an agile method by selecting method fragments with quick plug and unplug of the fragments; this offers a high degree of flexibility [15]

- *Living Software Development Process*: provides a vision to support the selection of process fragments at the starting of the project (static tailoring), the reconstruction of process fragments during the project to support frequent changes of the project's environment and requirements (dynamic tailoring), and a continuous improvement of an organization's standard (evolutionary process improvement) [13]
- *Other Agile-influenced Hybrid Approaches*
    - *Tailored Agile*: applies agile practices into the existing process [17]
    - *Agile Process Tailoring and probLem analYsis (APTLY)*: combines techniques and ideas from a process knowledge base of best practices and local experience [16]
    - *Situated process and quality framework*: implements agile values and principles into an organization's standardized process of RUP and CMM [18]

The Harvard Business School's Case Method provides students the experience of solving complex problems by analyzing and resolving various business cases in a controlled and compressed environment. In other words, students are refining their personal problem-solving algorithms. Students are encouraged to be creative and draw from their personal knowledge base to solve these problems. Future business leaders cannot expect to be able to solve real-world situations working in the business environment by dogmatically applying a specific financial theory. The program encourages students to create their own understanding and strategies to solve future problems. This technique also can be applied to software engineering programs. These programs need to emphasize that to be a successful project manager, one must use judgment and eclectically select or apply the body of knowledge of software engineering in an appropriate fashion and not blindly apply the most popular methodology du jour.

2.2.4 Government Sector Contract-Based Projects

Software service providers in the public sector working on government projects are contractually required to apply rigorous project management processes. The government utilizes these processes to formally review and accept the progress of the project throughout its lifecycle. Government contracting officers and their technical representatives do not tolerate chaos. For example, government projects frequently cite several military standards, such as MIL-STD-1521, MIL-STD-498, DOD-STD-2167, DOD-STD-2167A, , MIL-STD-1679, and CMM for well-defined, planned, controlled, auditable, and tested projects [31, 48].

Agile methodology is beginning to be accepted for many government projects. As noted above, for example, the German government accepts agile strategy as one way to run projects with V-Model XT, a new official process model [7]. However, software development contractors for large projects often find it difficult to conscientiously use authentic agile methodology, which requires fundamental process changes in the public-sector contract-based, mission-critical project. ESD supports a strategy for balancing between plan-driven and agile methodology, which Boehm and Turner provide in their book [31].

In addition to process controls within the project, the typical government system has many interfaces with systems from different departments, services, or agencies, which also requires orderly and controlled interface definition processes. Another level of complexity is that these projects are not brand new products. They are retrofit projects, which are by definition heterogeneous in nature. They may be built with several different programming languages and technologies. The typical government contractor will be at some level of

CMMI, which means that it already has a set of predefined processes that the project team must follow and adhere to.

Extending advantages of agile methodology into the realm of large complex projects in the public sector is one of the basic objectives of ESD.

2.2.5 Three Steps of ESD

2.2.5.1 Summary

Wallin and Crnkovic state that 'successful project execution', 'successful technical solution', and 'promising business case' are three important aspects for a software development project [110]. Figure 18 shows the three aspects as a triangle. In addition, it has to be on time and on budget, and it must provide the required functionalities.

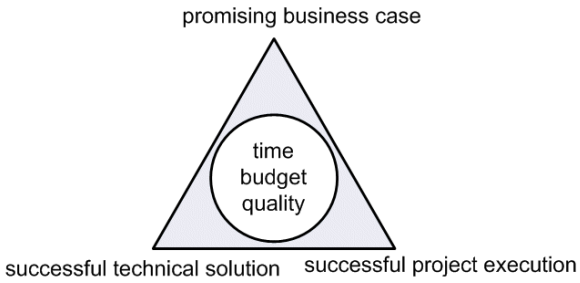


Figure 18 Three aspects for successful software development project

Many software development methodologies tend to focus primarily on the project execution aspect. ESD includes technical solutions and business cases to refocus its perspective on the overall success of the project in line with the business goal, instead of just on the project execution aspect.

The three steps of ESD application were shown earlier in Figure 16. The first step is to recognize the best practices and constraints placed on the project by the standard processes of the governing organization, e.g. corporation, program, or government. The second step is to assess the project relative to the five factors of tools, methodology, processes, people,

and leadership. The third step is to recognize in a timely manner and respond properly to the project by applying simple techniques, such as milestone reviews, simple questions, simple mathematics, and Management by Walking Around (MBWA).

2.2.5.2 Step 1: Recognizing Organization Standard

In order for a methodology to work on a public sector software development project, it must work within the confines and structures of the processes and procedures already defined for the contractor organization and the host program. For example, SEI-CMMI is widely adopted in the public sector contracting community. By definition, the company has to have a set of corporate procedures, processes and guidelines. The programs (contracts) of these companies adopt and tailor the corporate guidelines to the program. The project team is expected to adhere to the program level guidelines and tailor them to its specific project as necessary.

The ESD approach to integrating the CMMI processes and procedures is to accept them with open arms and integrate the processes into the everyday work environment.

2.2.5.3 Step 2: Assessing Project Factors

After the organization standards are recognized, the team needs to define critical success factors and risks by assessing the project. ESD recognizes that software development is a harmonized activity of tools, methodologies, processes, people, and leadership. Figure 19 shows these five factors in a pentagon configuration.

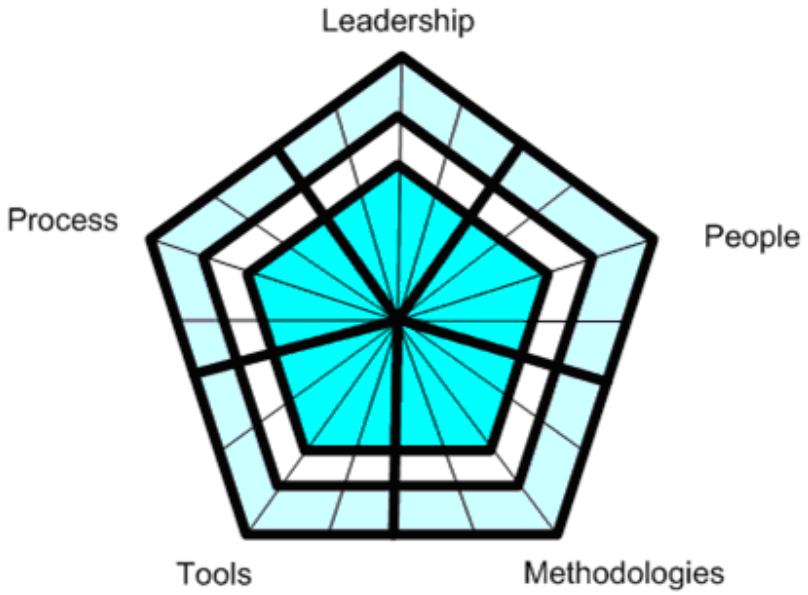


Figure 19 Five major factors in software development

Various tools are available to complete a software development project. Tools are often referred to as technologies, program languages, COTS products, architecture, or development frameworks. Mechanics and carpenters understand and practice the saying "Use the right tool for the job" every day. Whereas the saying "Give a boy a hammer, the world becomes a nail" shows that naive or dogmatic practitioners tend to misapply tools in a given situation.

The software engineering field provides various methodologies including Analysis and Design, Software Development Lifecycle, Process, Quality, Requirements, Configuration Management, Testing, and others to capture and share the best practices.

Processes and procedures help in repeating a series of steps. They do not guarantee a successful outcome. If the procedures were flawed in the first place, and there is no corrective activity, one will repeatedly produce the wrong product. Therefore, continuous process improvement is as critical as the well-defined process in the first place.

While traditional software development solutions tend to focus on methodologies and processes, agile methodology focuses on the other two factors of people and leadership. Agile methodology involves incorporating people as a key factor for better results. Qualified people are an essential key for good products.

ESD depends on leadership for software development projects. A success of the project depends on not only the good team but also leadership. By providing technical, functional, managerial, and visionary direction, effective leaders recognize and encourage qualified people to perform at their best [111]. Effective leaders pay attention to their team members and ensure that they are trained, guided, and appreciated for their efforts. An effective leader, not necessarily a technical guru, leads a team to success even with the imperfection of real-life projects [112]. The success of a software project depends on the qualifications of the project leaders, who must have theoretical knowledge and practical experience [113]. Table 7 compares the six leadership roles from Computer Sciences Corporation [114] and four leadership types from ESD.

Table 7 Leadership models in information systems/information technology

CSC	ESD
Chief Architect	Technical
Product Developer	Functional
Change Leader	
Chief Operating Strategist	Visionary
Technology Provocateur	
Coach	Managerial

In the real world, good leaders are hard to find. They are even harder to train. Effective leadership can be performed or contributed by more than one manager within a team. It can

come from project managers, program managers, software development managers, team leader, or senior developers.

2.2.5.3.1 Tools

Selecting the right tools is imperative to increase productivity. It includes technologies, architecture, frameworks, and tools. For example, one case study says that 80 percent error reduction and 40 to 80 percent of code reduction was achieved by its application framework [115]. However, "right" is a very subjective word. The team must understand that the recommended technical solution must assist to certify the success of the current job. To perform an objective evaluation, evaluation criteria [116] or checklists [117] can be used. Also, prototyping, a trade study, or Decision Analysis and Resolution (DAR) can help in the selection process.

The list of tools in Table 8 is an example of tool usages from one of the Java web-based application development projects in the U.S. DoD. Literally countless tools are available in the market, waiting to be selected. Therefore, software managers and team members should be capable of understanding what kind of tools the projects needs and equipping the project with a collection of harmonized tools, not just with a lengthy list of non-integrated tools.

Table 8 Tools selection example

Category	Tools
Java IDE	JetBrains IntelliJ IDEA, Eclipse
Database	Oracle
UML Modeling	Borland Together, IBM Rational Rose*
Database Modeling	Oracle Designer
Database Access	Quest TOAD
OR Mapping	Hibernate
XML Development	Altova XMLSpy
Framework	Apache Struts, Enhydra, Jcorporate Expresso**, Apple WebObjects**
Application Server	Caucho Resin, Tomcat**, BEA WebLogic*, IBM WebSphere*,

Category	Tools
	Jboss*, Microsoft IIS
Web Server	Apache
Business Intelligence	Cognos, Business Objects Crystal Report
Build Management	Jakarta Ant
Logging	Log4J
Unit Test	JUnit
Performance	Quest Jprobe
Object/Xml binding	JAXB
Web Services	Apache Axis
Template Engine	FreeMarker, Velocity
Wireless	J2ME*, RIM BlackBerry**
Source Control	PVCS Version Manager, CVS*
Defect Tracking	PVCS Tracker
Requirement Mgt	Borland Caliber
Regression Test	Mercury WinRunner
Test Management	Mercury TestDirector/QualityCenter
Help Desk Mgt	BMC/Remedy Magic Service Desk
On-line Help	RoboHelp
PKI Digital Signature	DBSign
CAC Middleware	ActiveCard, NetSign

** Evaluated, ** Prototyped*

After selecting the tools, frameworks, and other sub-systems, the team establishes a central repository. The repository stores experiences for reuse to prevent duplicated efforts or repeating of common problems and eventually to assist experienced-based process improvement [117, 118]. The team needs to create knowledge bases within the repository, outlining how to set up the development environment and utilize it with coding standards and standard operating procedures (SOP). This is crucial for diminishing the learning curve of new developers who may be subsequently added to the project and also for assisting the maintenance team that follows. Recently, low-budget Web 2.0 collaboration tools such as "Wikis" are becoming popular [119].

2.2.5.3.2 Methodologies

As software engineering encompasses many areas, various types of methodologies are used. To efficiently balance agile and plan-driven methodologies, other methodologies

from various software engineering fields have to be evaluated. For example, Software Development Life Cycle (SDLC) methodologies include Waterfall, Spiral, Prototype, RAD, RUP, and Agile. Analysis and design methodologies include Structural [120], Declarative, Object Oriented [121], and Service Oriented Architecture (SOA). Process methodologies include ISO 9000, and SEI-CMM. Requirement, configuration management, testing, and other areas in software engineering have their own methodologies.

The purpose of methodology is to capture the best practices and to share with others those best practices. The intent is to communicate complex ideas between customers and developers. When choosing a methodology, it is imperative to consider both sides of the coin. On one side is technical understanding and on the other is helping the customer visualize and understand the end product of the project. If the developers employ methodologies only to assist themselves in understanding the problem without helping the customer understand the resulting system, then the project will inevitably end up in a finger-pointing exercise, where the contractor and customer argue over their different understanding of requirements or designs. Architects have been solving this problem by progressing from blueprints to artist renderings of the building façade to 3-D scale models to the use of virtual reality walkthroughs. As Figure 16 illustrates, one of three aspects of a successful software development project is a successful business case. The project team does not dictate success in this area. The customers or business owners determine success or failure. If a project team blindly and dogmatically follows methodology without considering its customer's ability to understand the output from the chosen methodology, then surely there will be disconnections at the end of the project.

2.2.5.3.3 Processes

Software development is a complex process, and process maturity fosters performance improvement associated with quality, time, and productivity [122]. The software process is the set of methodologies and technologies to model (process modeling), support (process execution), assess (metrics and empirical studies), and improve (process improvement) software development activities. While the software development lifecycle methodology delineates the process, the software process defines a precise course of action [123]. The team should establish project specific processes that are within the structure of the overall corporate and program processes and the selected methodologies. These processes should be designed to reduce conflict, allow for repeatability, enhance management control, and allow the team to work smart and as a cohesive unit. A technically oriented project leader can look at this as designing and programming systems for human-based computing devices instead of silicon-based devices. In other words, the project leader plans people's activities under the name of processes.

The popular process quality model includes SEI-CMM and the ISO 9001 standard, and the popular process improvement method includes SPICE and IDEAL [123, 124].

2.2.5.3.4 People

People are the most important asset across industries for a company's success [125] and the software development industry cannot be an exception. ESD uses an extended definition of people, which includes all stakeholders in a project.

First, often the team structure in the public sector project is just given to the project. For example, agencies and departments join the project based on the authorization of a decision maker at a higher level. Also, the project generally does not get to hand-select the

team members who work on the project. It is important for the project leader to observe, train, and test the capabilities of each agency and contractor and member of them.

Second, each SDLC methodology requires different skill sets of people. As Table 2 provides, the agile method demands a richer mix of higher-skilled people. The information technology industry has experienced a high turnover rate of employees, and the higher-skilled people drive the rate upward. In fact, recognition and encouragement of middle-level-skilled people are essential to have stable supports in the software development industry [126].

Third, it is essential to train the stakeholders, especially customers and development team members, in their use of the tools, process, and methodologies. All team members must be drilled on the use of development tools, the configuration management procedures and tools, and any other procedures and methodologies that they must be adhere to. As with a set of golf clubs in a golf bag, the project leader switches the training styles in different situations. That is the same philosophy used in ESD's eclectic approach. During training, different leadership styles can be applied [127]. Sometimes it requires the project leader to sit with the other team members individually to observe them carrying out the tasks and procedures. This is analogous to drill sergeants watching recruits cleaning their weapons in the military.

Fourth, it is important to note how the stakeholders accept the changes. It is critical that the stakeholders and development team members are well informed and support the changes. Good people management skills will help to focus on the goals. People management practices of successful organizations include sustaining employment security, hiring the right people in the right place, organizing work into teams, providing

comparative compensation, and more [125]. Those management practices can be applied to software development organizations.

2.2.5.3.5 Leadership

Leadership assessment, the most difficult for any team, requires the team to look in the mirror and determine if the team has the proper skills, capabilities, and training to lead the team into this construction project. It is very difficult to perform this self-evaluation and be honest with oneself.

In order to simplify the self-assessment, ESD has identified four sub-factors of leadership in software development projects: visionary, technical, functional, and managerial leadership. Those four sub-factors of leadership need to be performed by the team leader or other staff members who can provide any absent factors.

First, visionary leadership is critical for long-term government projects. Software development projects need not only managers but also leaders who have vision. Managers produce plans for stability and leaders provide vision for changes, and both are complementary [128]. Table 9 describes management and leadership. A vision must serve the interest of stakeholders and must be easily translated into a realistic competitive strategy. According to one of the conceptual frameworks about vision [129], the core ideology includes core values (what we stand for, i.e., Imagination: Walt Disney) and core purpose (why we exist, i.e., To make people happy: Walt Disney), and the envisioned future includes long-term goals (i.e., Become the Harvard of the West: Stanford University, 1940s) and vivid descriptions.

Table 9 Management and leadership [128]

Management	Leadership
Planning and budgeting	Setting a direction (creating vision and strategies)
Organizing and staffing	Aligning people
Controlling and problem-solving	Motivating people

Second, technical leadership is critical because software development utilizes various technologies. Since the industry introduces newer technologies so rapidly, it is not unusual that many project leaders have not been exposed to leading-edge technologies after the project leader was promoted to the leadership position. In this case, unless the leader can learn the technology in a timely manner, the leader must have staff members who can support the team to avoid difficulties arising from a lack of technical leadership.

Third, functional leadership is about domain knowledge of the business process of the government agency. When a team develops software for a government agency, it is critical to have a sufficient understanding of how the agency operates its business and what policies and regulations relate to this new software.

Fourth, managerial leadership is about management for project execution. When a software engineer is promoted to the project leadership position on the basis of his technical abilities, the organization must train him or her well for managerial skills, such as project management, people management, schedule, and budget.

Great programmers, like great craftsmen, will create great software. A great programmer with great tools will create great software faster. However, complex projects are not single-developer projects and require a team of programmers and a manager. In a team environment, a bad manager will trump great programmers and cause bad software to be produced later. Leadership determines if the human system works or not. Leadership

creates the environment in which developers operate. In combat, failures in leadership get people killed. In software development, failures in leadership result in poor quality software delivered late or never. In both situations, that which is lost by lack of leadership is irretrievable.

2.2.5.4 Step 3: Recognizing and Responding to Project Circumstances

2.2.5.4.1 Overview

The team leader must be able to lay out a project plan that is realistic in terms of organizing the work of the development team and the planned duration of tasks. The project plan must be meaningful to the external program control team and to the project manager and developers. Project Portfolio Management (PPM) software has been developed to offer a reflection of the reality of the project. Some project management techniques help leaders to stay with the reality and avoid being sucked into the looking glass and living in the reflection.

For an enterprise to adapt to a rapidly changing business environment, Haeckel [130] suggests "sense-and-respond" (SaR) over "command-and-control." SaR organization performs "sense-interpret-decide-act" processes. SaR enterprise monitors trends and acts in a timely manner by using effective decision-support tools [131]. The third step of ESD, recognizing and responding to project circumstances, is utilizing this SaR approach, applying it at the software development team level instead of the enterprise level.

Once the project starts, the project leaderships must stay in touch with what is actually happening on the project, regardless of the development methodology, and lead the project team to create the right solution within the constraints of the project. In order to accomplish this, the project team must recognize risks, issues, and otherwise unproductive practices and formulate timely responses to any of these negative events. The project leader must

support the team by recognizing the problems and taking the proper actions. Although there are many traditional ways of identifying risks and issues (e.g., weekly reviews and formal risk assessments), ESD suggests four simple yet very effective ways for the identification and recognition of problems for effective project management in addition to the formal governance process. The first is early and continuous integration of software sub-systems. The second is by asking simple and direct questions. The third is by doing simple mathematics. The fourth is management by walking around. These four management principles and practices, which are selected from many others, work efficiently and effectively in contemporary software development projects, especially in the government sector.

2.2.5.4.2 Continuous Integrations and Milestone Demonstrations

This practice combines a milestone concept from plan-driven methods [132] and a continuous integration and reflection of the product concept from agile methods. The best way to determine the health of any systems development project, either software or hardware, is to see the pieces of the system work together as early as possible. When the project leader defines the project schedule, he or she should plan for periodic integrated milestone demonstrations for the whole development team. The frequency of these demonstration or reviews must be appropriate for the overall length of the project and the development pace of the team. Ideally, the whole team would jointly review the demonstration, and the project leader should invite the customer representatives to observe. Although there are risks associated with having failures in front of the customer, the benefits of gaining a true picture of the status of the project and opening the communication channels between team members far outweigh the risks. According to previous studies from OECD, the United Kingdom, and Norway [133], user involvement is

considered an important key, and the lack of end-user involvement in government IT projects provides more challenges than in private IT projects.

Instead of depending on second hand status reports as a diagnostic tool to determine the project health, a team can observe the accomplishment of their project. The team can compare between what they have reported and what they have actually accomplished. The team can see the true accomplishment of the whole team. Every team members can understand who is ahead of schedule and who needs help. The value to the project leader of stakeholders actually seeing live demonstrations of the sub-components working together on a periodic basis cannot be overstated.

The added benefit of having the whole development team attend the review is that during a milestone demo, the team discusses not only what it has accomplished but also what it needs to do for the next milestone. The whole team takes ownership of the process and takes pride in the outcome of the project. The whole team knows the objective, risks, issues, and other obstacles that must be overcome to achieve the objective.

2.2.5.4.3 Ask Simple Questions

"Asking the right question at the right time," a critical technique in education and medicine [134], has been applied in the software engineering field, for example, during design reviews [135], software measurement [136], or system acquisition [137].

A good way to determine the quality and depth of understanding is for the project leader to ask team members simple but insightful questions. When simple answers do not come back from the team members, it usually indicates the team does not fully understand the situation or the problem. When there is no good answer, the project leader must coach or guide the team member in a professional manner.

Although these questions are simple in nature, they should be insightful and relevant to the problem at hand. The team leader must be wise enough to identify the weakness in the design and process to know which simple questions are appropriate.

Simple questions help the team to prevent any last-minute surprises. Often the development leader does not have the same technical proficiency of knowledge as the technical team. However, if management is able to ask good simple questions, then management may be able to eliminate the communication gaps. However, this technique has to be used properly. Some team members can be defensive, resulting in less than positive outcomes.

2.2.5.4.4 Do Simple Math

The forest is often missed for the trees. When planning or assessing project health, the team should use simple mathematical equations to look at the forest. All too often, managers will be drawn into the forest and focus on the complexities of the trees and miss the whole point of navigating through the forest. This simple calculation technique can help to avoid prospective disasters in software projects [138].

Basic ratios are very good simple mathematical equations to start with. For example, assume one is working on a four-week development project. The task has 100 programs to be developed. The development team has two programmers on it. The simple math at the planning stage would be that each developer would have to develop and unit test 50 programs in the four weeks or 12.5 programs per week, or 2.5 programs per day. Without asking about level of complexity of the individual programs, the project manager already can ask whether the pace is realistic for the developers to sustain. If not, then this is a good time to ask for additional time or resources to reach a sustainable pace. The manager can quickly determine the status of the project during execution by using this simple math.

One simple equation to remember is that at a given point in time on the project, each day in the future becomes an increasingly significant percentage of the remainder of the project. In other words, if you have two weeks left in the project, then the first day of the two weeks is 10% of the remaining project. On the second day, it becomes 11.11% or 1/9, and the third becomes 1/8, and so on. Once the project reaches this point, it is very important for the project leader to assess the current situation and wisely utilize the remaining time.

2.2.5.4.5 Adopt Management by Walking Around (MBWA)

MBWA means being in touch with customers, suppliers, partners, and one's own people for face-to-face listening, coaching, and facilitating [139]. This method is to help managers obtain the real-time status of the nature of the project. By walking around and interacting with the team members who actually do the work, a manager will be able to hear first-hand the real problems and successes of the project. In addition, by walking around and interacting by two-way communication with the customers being served, the manager will receive real feedback [140].

By walking around, a leader can lead from the front. The team members can see the leadership among the troops, taking interest and truly understanding the situation that everyone is facing. MBWA also creates opportunities for management to have positive interactions with the team, whereas a manager who hides in his office will tend to have only negative interactions with the team. In such a case, the only time he will want to see the team is when something is wrong. This type of behavior does not build trust or promote bonding between management and the development team.

Although weekly status reports are essential documents of project status, MBWA along with the other techniques enables project management to obtain the true gauge of the

status of the project and to improve the relationship between management and the team. A supplemental activity of MBWA is breaking bread with the development team. ESD has found that having team lunches after the scheduled milestone reviews is a good way to improve bonding between team members and the leader.

MBWA is not limited to one's own people, but includes customers, suppliers, and partners. It supports the large government software development project, which often involves team members and customers from other departments as well as partners and suppliers in remote locations. However, when traditional MBWA may be not feasible for distributed teams or virtual teams [141], the team can utilize other techniques, such as continuous integration and milestone demonstrations.

2.2.6 Informal Assessment

This section provides a summary of an informal assessment that applies ESD to an e-government web application project for the U.S. Department of Defense [32]. Figures included in this section illustrate how ESD can be applied throughout the lifecycle. Table 10 presents a summary, Figure 20 shows a lifecycle diagram, and Figure 21 illustrates the changes in the ESD Pentagon throughout the lifecycle.

Table 10 shows five project factors in the first column, each assessed by four selected sub-factors in the second column. These sub-factors are rated from 0 to 3, indicating levels of preparedness: None (0), Plan-Driven (1), Hybrid (2), or Agile (3). Along with the rating, descriptions of the events that render such ratings follow. Each event includes an indicator: positive (+), neutral (=), or negative (-). This case study has three assessment checkpoints, one at the beginning of project, one after the design review, and one before the system testing. Table 9 presents them in the third, fourth, and fifth columns. The table also charts recognitions and responses from the project leaders in the categories of Continuous

Integration (CI), Simple Questions (SQ), Simple Mathematics (SM), and Walking Around (WA).

Table 10 Summary - Case Study A [32]

Project Factors	Sub-Factors	Recognizing organization standard	Assessing project factors (System Requirement & Design Review)	Assessing project factors (Test Readiness Review)	Recognizing and responding to project circumstances
Leadership	Visionary	2 + A new project manager and a technical team leader are highly motivated. + The new manager (highly-paid contractor) has solid industry experiences.	No change N/A	No change N/A	N/A
	Technical	1 = The technical team leader is familiar with existing Java framework.	No change N/A	2 + Three team members, who became experts for the new framework, are promoted.	N/A
	Functional	2 + The technical team leader, product manager, and requirement engineer are very familiar with the functional area.	2 + Prototyping is used to solve the difficulty in visualizing the end product from the requirements	No change No change	WA: The leaders found there are strong barriers between technical team and functional knowledge team
	Managerial	0 - The new manager and new team members are not trained for the organizational standard.	1 + The new team is provided training for organizational standards.	No change No change	N/A
People	Structure	0 - There is poor collaboration among teams prior to their own phase (i.e., requirement, development, or system testing). - 70% of team members are newly hired.	1 + Integrated Product Team is used to solve the difficulty of team communication across organizational teams.	2 + Theory W is applied to make everyone happy.	WA: The development team and system test team complain to each other.
	Skill Set	1 = The team has Java skill set with existing technology.	No change No change	2 + Pair Programming (from XP) is applied to perform timely domain and technology knowledge transfer timely.	CI: The team found that some developers are too slow to learn the new framework.

Project Factors	Sub-Factors	Recognizing organization standard	Assessing project factors (System Requirement & Design Review)	Assessing project factors (Test Readiness Review)	Recognizing and responding to project circumstances
	Training	0 - There is no training plan due to time limitation	1 + Vendor training and OJT are provided.	No change No change	N/A
	Acceptance	1 = The organization only worked with Waterfall. The team members are not familiar with non Plan-Driven Method.	2 + The team appreciates new practices because it fosters the winning spirits	No change No change	N/A
Methodologies	Requirement	1 = The full lifecycle is Waterfall, and the current schedule goal is 5 months, which is very intensive.	No change = Joint Application Design sessions (from RAD) are hosted by representatives of real users, not just the requirement analyst. It is used to resolve communication issues.	No change No change	N/A
	Analysis & Design	1 = Waterfall	2 + Prototyping	No change No change	N/A
	Development	1 = Waterfall	2 + Prototyping + Non-Serialized activities (from RUP) are used to resolve schedule delays due to serialization of activities.	No change + Continuous Integration and milestone demo with stakeholders are used.	CI: Monitor real progress
	Test	1 = Waterfall	No change No change	2 + Pilot Site Testing is used to prevent any last-minute surprises.	SQ: The Customer wants to avoid any last-minute surprise.
Tools	Tool	1 = Existing tool	No change No change	No change No change	N/A
	Framework	1 = Existing framework	2 + New framework increases response time for changes	No change No change	N/A
	Architecture	1 = Existing architecture	No change No change	No change No change	N/A
	Technology	1 = Java web-based application	No change No change	No change No change	N/A
Process	Model	1 = CMM Level 3	No change No change	No change No change	N/A
	Execution	1 = CMM Level 3	No change No change	No change No change	N/A
	Assess	1 = CMM Level 3	No change No change	No change No change	N/A
	Improvement	1	No change	2	WA: The project

Project Factors	Sub-Factors	Recognizing organization standard	Assessing project factors (System Requirement & Design Review)	Assessing project factors (Test Readiness Review)	Recognizing and responding to project circumstances
		= CMM Level 3	= Collective Code Ownership	+ Continuous Integration (from XP)	leader hears that it takes so long to change a line of code WA: The project leader hears that it is hard to measure the progress. WA: The project heard that "the beginning of an integration test phase is usually stressful"
Critical Success Factors	N/A	The existing process and framework can't meet the deadline. The project wants new frameworks, process or methodology to accomplish the schedule.	The project has to resolve communication gaps. Each stakeholder has different interpretation and expectation of the product.	The project applies continuous Integration and pilot test techniques to optimize the schedule.	SM: The existing process and framework can't meet the deadline.

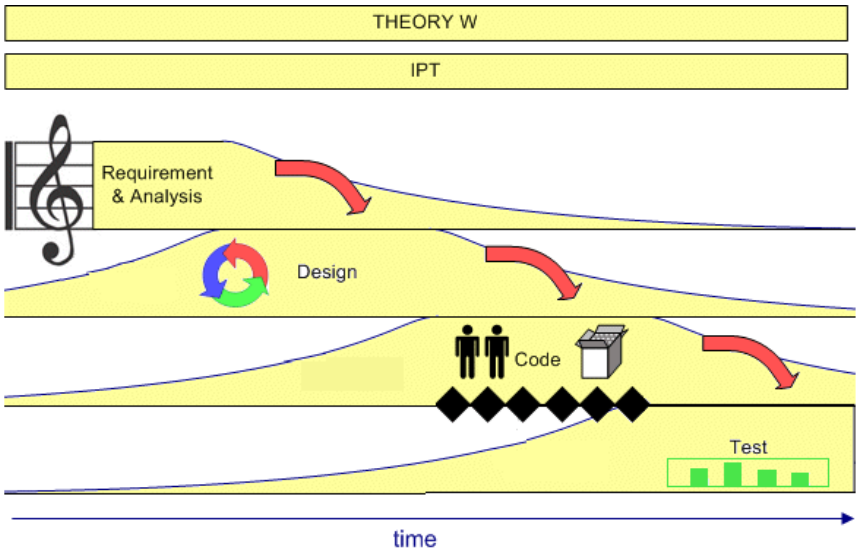


Figure 20 Lifecycle – Case Study A [32]

Figure 20 illustrates the inter-relationship of the selected practices used on this project over time. The organization has performed CMMI level 3, and follows a conventional

Waterfall methodology. This application uses ESD utilizing mock-up screens from Prototyping; Joint Application Design (JAD) sessions technique from Rapid Application Development (RAD); non-sequential activities from Rational Unified Process; and Theory W from the WinWin Spiral model. Also, XP practices such as continuous integration, collective code ownership, and pair programming are applied.

In Figure 20, JAD activity is iconized as a musical note (JAD sounds like JAZZ); Mock-Up Screens are iconized as a circular arrow; Theory W, "make everyone a winner," is shown as a flat-wide box on the top, covers the entire lifecycle, and shows a concept of non-sequential activities; pair programming is shown as a two-person icon, continuous integration is shown as a series of diamond milestone icons; shared code ownership is shown as an open box; IPT is shown as a flat-wide box on the top; and pilot test is shown as a bar chart icon.

Figure 21 shows the results of project factor assessment at specified checkpoints. The initial assessment diagram demonstrates that the project has three missing factors in a plan-driven layer. It shows that the project requires work on the fundamental areas before implementing any agile principles. The interim diagram, assessed after the design phase, shows that the missing factors are enhanced. It indicates that the project now has a good foundation to accommodate agile practices and principles. The final diagram, assessed before the system test phase, indicates that the project promotes itself by balancing plan-driven and agile methodologies. This Pentagon chart is visually accessible and provides significant benefit when the project leader keeps monitoring the chart until the project is fulfilled.

For example, when the pentagon form displays negative or no positive movements while the team is trying to improve the sub-factors, it indicates that the project manager needs help. The project may be running into problems.

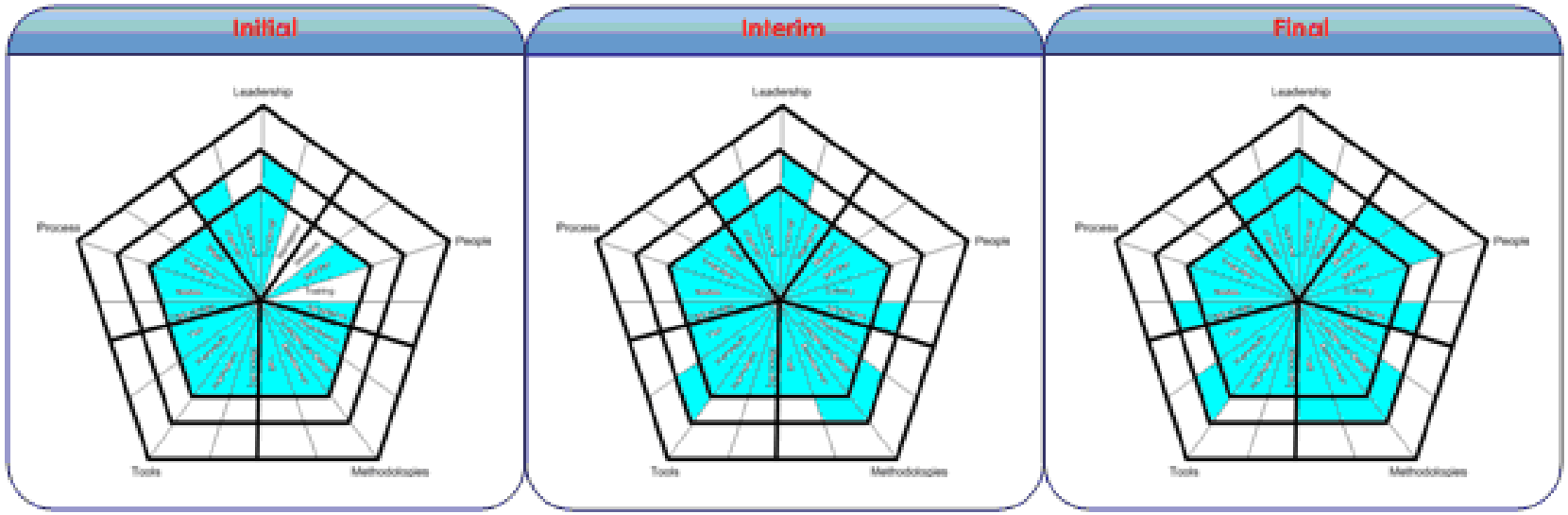


Figure 21 ESD Pentagon – Case Study A

2.2.7 ESD Agile Best Practice Checklist

ESD provides the Agile Best Practice Checklist in Figure 22 and Agile Program Assessment Form in Table 11 to select agile best practices, thereby allowing a balance between Agile and traditional methodologies. The Agile Best Practice Checklist helps to evaluate agile best practices for government software development projects. This checklist provides a list of best practices from popular agile methods in addition to commercial, scaling, and government agile best practices. This checklist can be customized to a specific project.

The checklist and form can be used to review multiple best practices based on various factors. This enables the developer or manager to drill down and select the right set of best practices for the program.

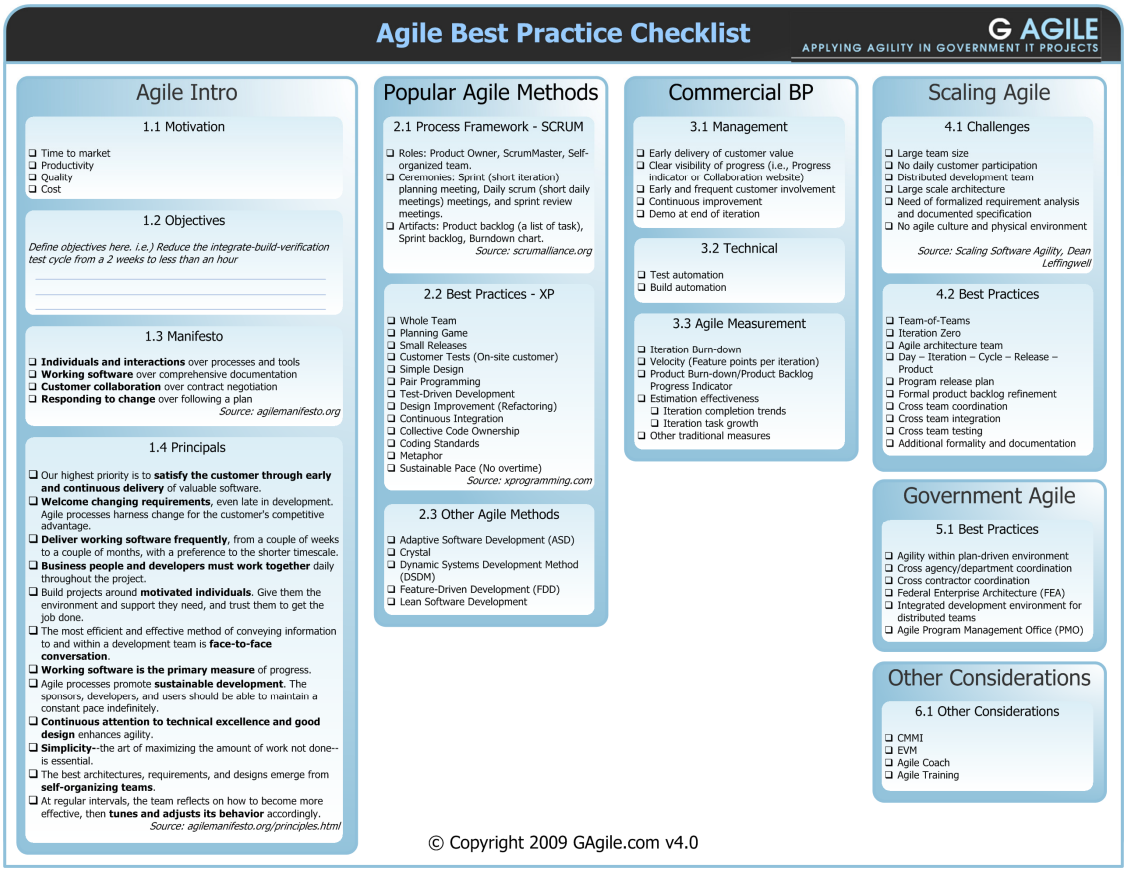


Figure 22 ESD Agile Best Practice Checklist (version 4.0)

The first column categorizes five project factors (People, Methodologies, Tools, Process, and Leadership), and each project factor will be assessed by customizable sub-factors in the second column.

In the third column, select one of (Initial (1), Managed (2), Defined (3), Quantitatively Managed (4), or Optimizing (5)) to rate these sub-factors, assigning a rating for each sub-factor in your project.

In the fourth and fifth column, enter a description of challenges your team faces and the corresponding actions your team will take. Input will be continually updated and revised. Multiple challenges and actions can be entered.

Table 11 Agile Program Assessment Form

Project Factors	Sub-Factors	Rate	Challenge / Comments	Approach
People	**Structure**	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Skill Set	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Training	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Acceptance	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
Methodologies	**Requirement**	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		

Project Factors	Sub-Factors	Rate	Challenge / Comments	Approach
	Analysis and Design	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Development	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Test	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
Tools	**Tool**	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Framework	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Architecture	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		

Project Factors	Sub-Factors	Rate	Challenge / Comments	Approach
	Technology	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5))		
Process	**Model**	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Execution	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Assess	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Improvement	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
Leadership	**Visionary**	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		

Project Factors	Sub-Factors	Rate	Challenge / Comments	Approach
	Technical	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Functional	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Managerial	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		

2.2.7.1 Introduction to Agile Software Programming

Agile software development methodologies are based on common motivation, manifesto, and principles. They target high quality products with lower cost, which also aligns with the customer's needs and goals. The program understands the common motivation, manifesto, and principles, and develops the program-specific measurable objectives.

2.2.7.1.1 Motivation

The general motivation in adopting Agile methodology to the organization includes delivering a working product continuously in the shortest time possible, working in a highly productive team, ensuring quality and controlling cost.

The software programming agility provides the following business benefits:

- Continuous delivery of a working product
- High productivity
- High quality

- Cost-saving opportunity.

2.2.7.1.2 Objectives

The following are the sample objectives of utilizing the Agile software development best practice:

- Improve customer satisfaction by being more flexible with respect to incorporating requirement changes while still understanding their impact
- Reduce cost by shortening the integrate-build-verification test cycle
- Improve quality by reducing the response time to resolve the defects
- Integrate agility with program specific CMMI best practices
- Deliver working solutions that meet the needs and expectations of stakeholders through continuous collaboration

2.2.7.1.3 Manifesto

In February 2001, 17 leaders of the Agile methodologies met and developed the "Manifesto for Agile Software Development (http://www.agilemanifesto.org):

2.2.7.1.4 Principles

The principles of the Agile software development (http://www.agilemanifesto.org/principles.html) support the philosophy of the manifesto.

2.2.7.2 Popular Agile Software Development Methodologies

As Agile software development gains popularity, process frameworks such as Scrum, best practices methods like XP, and other Agile software development methodologies are implemented to help deliver business value. The next section describes each method and its benefits. The program will adopt the selected best practices from Scrum and XP.

2.2.7.2.1 Scrum – Process framework

Scrum derives from the rugby word scrummage, which can be described as a team of developers who have a well-defined incremental role with a specific task to complete. In software development, this lightweight process can be utilized to use iterative, incremental best practices. Tasks are structured in cycles called sprints and a list of tasks for each sprint is called backlog. At the end of each sprint, project milestones can be met. This is discussed daily during short scrum meetings. The team discusses its accomplishments and obstacles for current tasks. The following is a high-level framework of Scrum. More information on Scrum can be found at http://www.scrumalliance.org/pages/what_is_scrum

- Planned Meetings
    1. Sprint planning
    2. Daily Scrum
    3. Sprint review
- Roles
    1. Product Owner
    2. Scrum Master]
    3. Self-organized team
- Artifacts:
    1. Product backlog (a list of tasks)
    2. Sprint backlog
    3. Burn-down chart

Figure 23 illustrates the Scrum process flow.

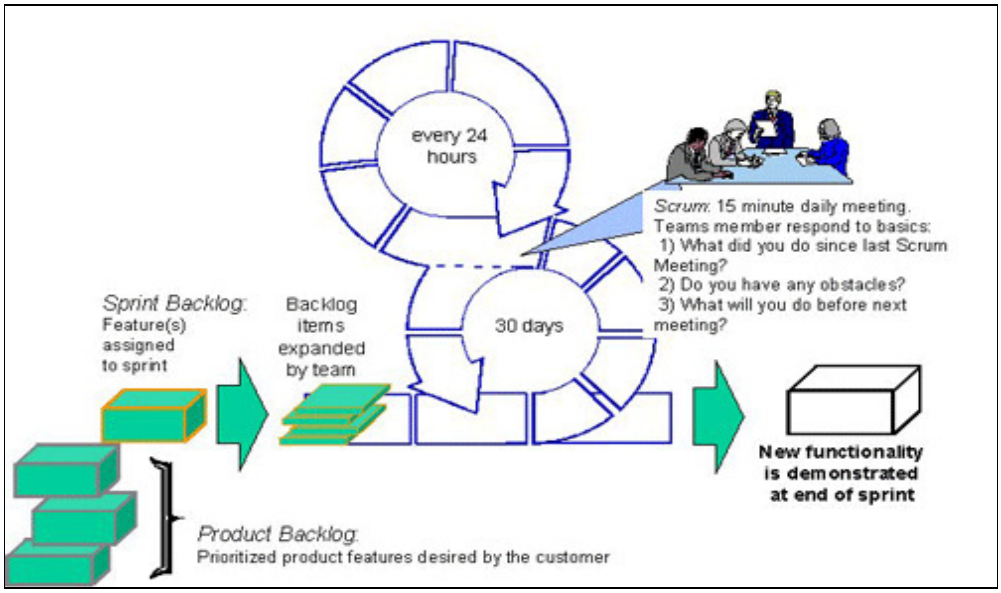


Figure 23 Scrum Process Diagram [77]

2.2.7.2.2 eXtreme Programming

XP programmers plan, design, develop, test, and release in short development cycles throughout the development lifecycle by a small team with customer collaborative involvement. A quick overview of XP is available at http://xprogramming.com/xpmag/whatisxp.htm. Key practices of XP include:

- Planning game: Predicting accomplishment for the given schedule rather than predicting schedule for the given scope.
- Small releases: Releasing tested software every iteration and on a frequent basis
- Metaphor: Developing a concept of understanding
- Simple design: Building software with simple designs throughout the lifecycle
- Test-driven development: Building test cases and then coding instead of coding and then building test cases

- Refactoring: Refactoring to improve the design of existing code
- Pair programming: Pairing two programmers to collaborate
- Continuous integration: Fully integrating code changes at all times
- Collective code ownership: Improving any code at any time by any pair of programmers
- No overtime
- On-site customer: Having a customer available whenever they are needed to answer questions and to provide direction

2.2.7.2.3 Other Agile Methods

Various methodologies and practices have been developed to adopt continuous changes in software development seamlessly. Table 12 provides an overview of other Agile methodologies.

Table 12 Overview of Other Agile Methods

Agile Methods	Overview	Key Practices and Process
Crystal	A framework for software development methodology selection based on staff size, system criticality, and project priority	
ASD	Accepts continuous changes by continuous adaptations	"Speculate-Collaborate-Learn" lifecycle, which replaces the "Plan-Design-Build" lifecycle
FDD	Features object-oriented based development of components with	Five steps focused on design and development:

Agile Methods	Overview	Key Practices and Process
	collaboration between domain experts and programmers.	Refer Figure 13
DSDM	Project delivery framework with iterative and incremental steps, which takes less time and discovers and corrects problems earlier than the waterfall method.	Seven steps: Refer Figure 14
Lean Software Development	Improving the delivery of software while minimizing the impact on processes.	The principles of Lean are: • Add value to the Customer • Create Knowledge • Respect People • Build integrity • Deliver Fast

2.2.7.3 Additional Commercial Agile Best Practices

This section provides an overview of the additional commercial best practices.

Besides the best practices from the popular Agile methods, the program also adopts some additional best practices, which have been introduced as lessons learned from the adapters and implementers of Agile methods on many projects. These additional commercial Agile best practices are categorized as project management, technical, and measurement areas, and work well with best practices from the popular Agile methodologies listed in Section 2.2.7.2 Popular Agile Software Development Methodologies.

Management plays the important role of ensuring that customer values have been addressed throughout the duration of the life cycle by assessing progress indicators and encouraging the project sponsor's involvement as early and as frequently as possible. Technical best practices align themselves with test and build iterations, which can foster continuous improvement in the process to deliver on time. By using the measurement, the team can understand impact of any requirement change requests.

2.2.7.3.1 Management

The following are additional commercial best practices for the project management area.

- Early delivery of customer values
- Clear visibility of progress (that is, progress indicator or Collaboration Web site)
- Early and frequent customer involvement
- Continuous improvement
- Demo at end of iteration.

2.2.7.3.2 Technical

The following are additional commercial best practices for the technical area.

- Test automation
- Build automation

2.2.7.3.3 Measurement

The following are additional commercial best practices for the measurement area.

- Iteration Burn-down: This is a working-level progress measurement, for use by individual teams and team leads. Figure 24 is an example of an Iteration Burn-Down chart showing the work remaining (hours).

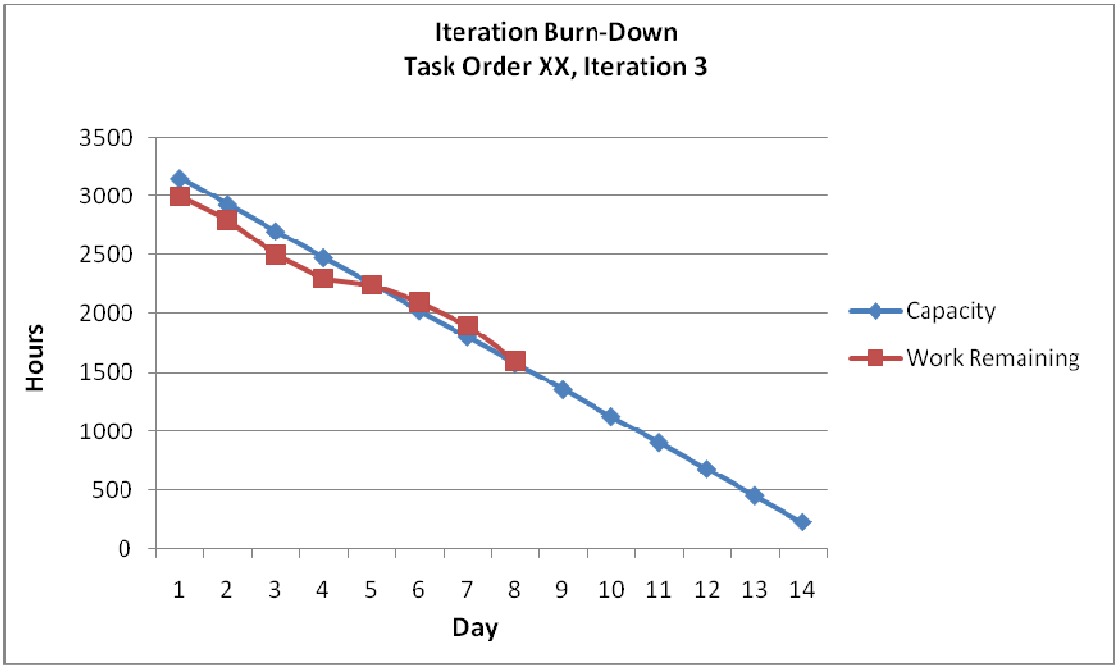


Figure 24 Sample Iteration Burn-down Chart

- Velocity (Requirement per iteration): A simplified chart showing velocity for one team in terms of requirement per iteration. Figure 25 is an example of a Velocity chart showing the number of successfully delivered requirements per iteration.

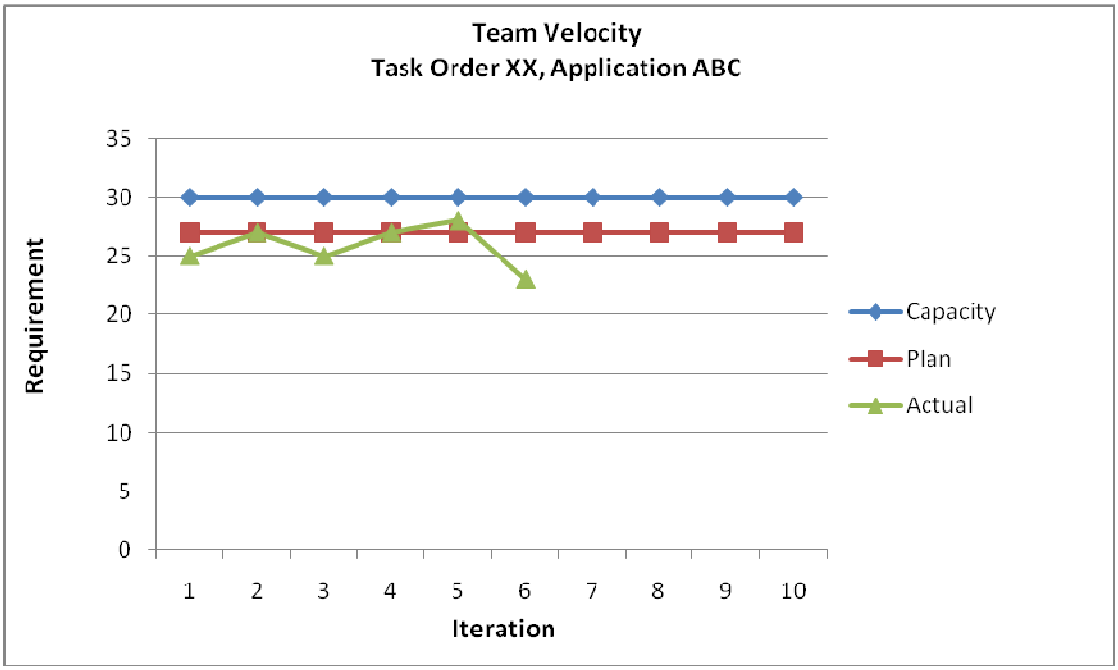


Figure 25 Sample Velocity Chart.

- Product Burn-down / Product Backlog Progress Indicator: As the example in Figure 26 illustrates, the Product Burn down chart shows progress towards completing the requirement in the product backlog.

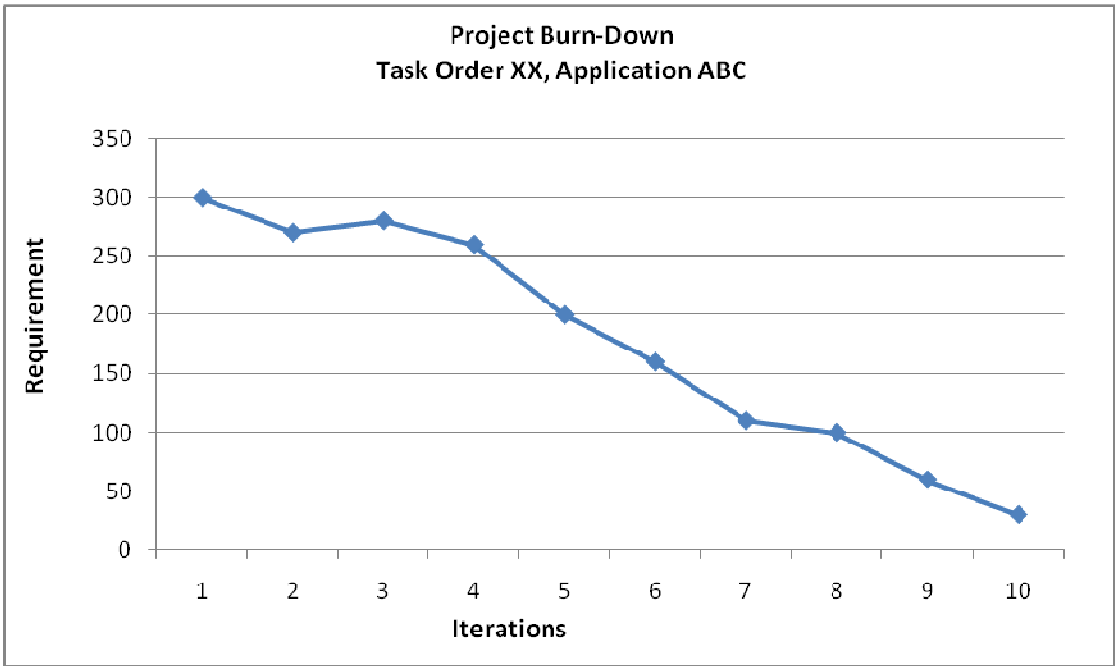


Figure 26 Sample Product Burn-down Chart

- Estimation effectiveness
  - Iteration completion trends: Figure 27 is an example of an Iteration Completion Trends chart showing number of requirements differed per iteration.

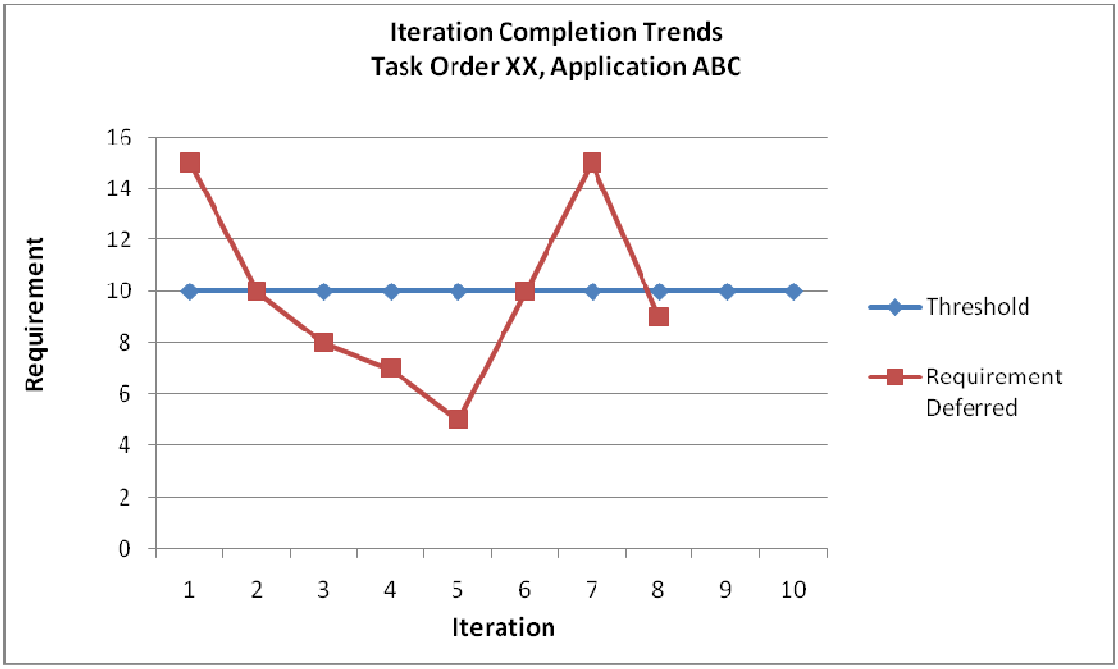


Figure 27 Sample Iteration Completion Trends Chart

- Iteration task growth: Figure 28 is an example of an Iteration Task Growth chart showing number of requirements added per iteration.

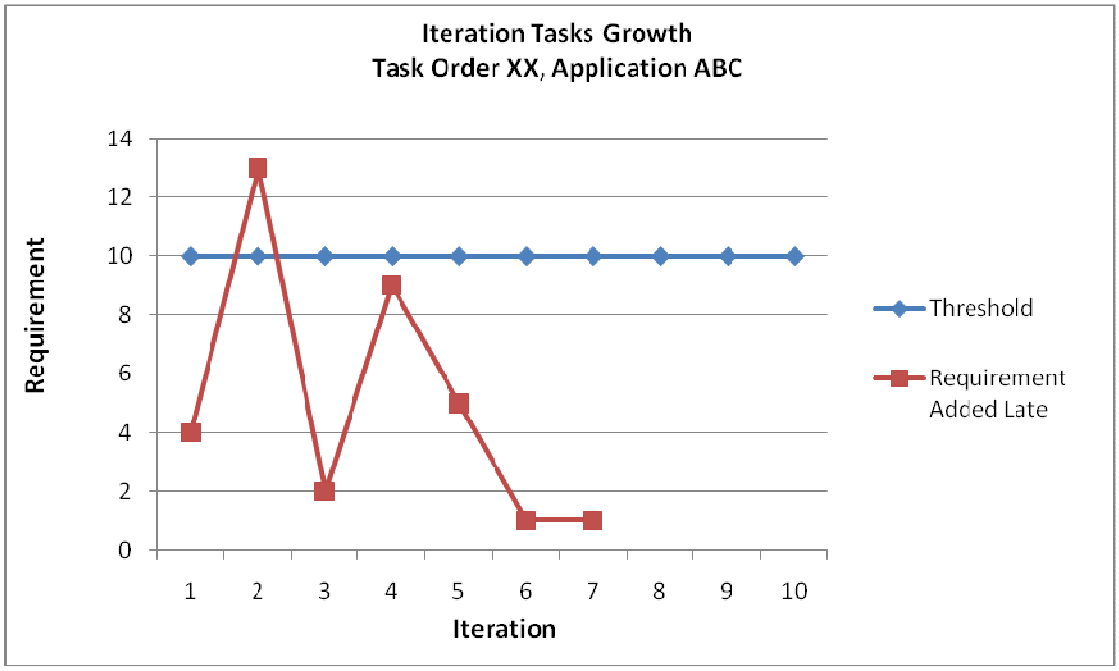


Figure 28 Sample Iteration Task Growth Chart

- Other traditional measures.

2.2.7.4 Scaling Agile

This section provides an overview of the Scaling Agile, also known as Enterprise Agile, best practices.

Best practices in the section 2.2.7.2 Popular Agile Software Development and additional best practices in the section 2.2.7.3 Additional Commercial Agile Best Practices were initially developed for small-scale software development projects. However, many best practices have been added based on lessons learned from large-scale agile software development projects. The team will benefit from the proven scaling Agile best practices, described in section 2.2.7.4.2 Scaling Agile Best Practices.

2.2.7.4.1 Challenges of Applying Agile Method to Enterprise

The general Agile methods, described in Section 2.2.7.2 Popular Agile Software Development Methodologies, are not designed for enterprise organizations which have the following characteristics [89].

- Large team size
- No daily customer participation
- Distributed development team
- Large scale architecture
- Need of formalized requirement analysis and documented specification
- No Agile culture and physical environment.

2.2.7.4.2 Scaling Agile Best Practices

To assist large enterprise programs in taking advantage of the Agile method, the following best practices are recommended.

- Team-of-Teams: Leads from each of the Task Orders (TO) to manage dependencies and coordinate the efforts among the teams (Scrum approach)
- Iteration Zero: Some architecture development up front
- Agile Architecture team: Architecture is a critical part of scaling agile best practices to meet real world demands. Document a minimalist architecture, develop a working architecture foundation, define and maintain system qualities and architectural vision
- Day – Iteration – Cycle – Release – Product: Cycles are collections of several iterations, and provide an interim management and technical coordination point.

- Program release plan
- Formal product backlog refinement
- Cross-team coordination
- Cross-team integration
- Cross-team testing
- Additional formality and documentation.

2.2.7.5 Government Agile Best Practices

Most of the government applications interact with other programs or lines of business, and do not easily adapt to change due to its complexity and scale. In fact, the Agile Manifesto includes some conflicts with the characteristics of government projects.

- The project has to follow predefined processes and use enterprise tools.
- Formal deliverables include documents.
- It is not common to have on-site collaborative customer.
- There is no culture of welcoming requirement changes.

Agile methodologies have limited support for large or distributed team, subcontracting, and safety-critical, large or complex software. Most of those limitations are characteristics of government projects.

The following are Agile best practices for government programs.

- Agility within a plan-driven environment using the Agile Program Assessment Form and the Best Practice Checklist
- Cross-agency / department coordination
- Cross-contractor coordination

- Federal Enterprise Architecture (FEA)
- Integrated development environment for distributed teams
- Agile Program Management Office (PMO)

2.2.7.6 Other Considerations

This section provides the description of other considerations while applying Agile best practices in a government program.

2.2.7.6.1 Capability Maturity Model Integration

While often thought of as two incompatible models and methodologies for developing complex software products, benefits can be derived by utilizing both Agile and CMMI best practices in the same project. Already many government programs can further improve business performance by exploiting the strengths of each approach.

Agile projects are best typified by adaptive planning. With releases delivered on a more frequent basis, they return value with each iteration and allow for customer inspection of the product at regular intervals. The method is software-centric or product-centric with the focus on development, as opposed to the detailed plan and documentation-centric approach of CMMI. CMMI focuses on process improvement.

The Software Engineering Institute (SEI) publication, CMMI or Agile: Why Not Embrace Both! [142], provides a multi-dimensional comparison of the two paradigms.

At the project level, CMMI focuses on process maturity, and Agile methods focus on ‘adaptive’ and ‘lightweight’ method. When implemented together, each provides key strengths.

2.2.7.6.2 Earned Value Management

Many government programs implement Earned Value Management (EVM) to measure project progress by combining cost, schedule, and scope measurements. As the program is defining Agile best practices for the program, the program needs to coordinate with the EVM process.

2.2.7.6.3 Agile Coach

As a project adapts to Agile software development methods, many organizations have found it useful to have an Agile coach. This is a person that has extensive Agile project experience and is an essential mentor of the development and project management team. The Agile coach is a facilitator that can quickly assess the current status of a project and report gaps in order to successfully move a project from a traditional methodology to an Agile state of mind. The Agile coach is also actively involved in keeping up on updates to best practices and can help a project in organizational and professional development.

2.2.7.6.4 Agile Training

When a government program decides to use the Agile method on any given task order, the program would ensure the team is adequately trained.

Chapter 3: Research Questions and Method

This section outlines research methods. Section 3.1 presents the research questions. Section 3.2 describes the action research paradigm and the selected methodology. Section 3.3 presents the research procedures that are used.

Previous action research studies provided guidelines during construction of this research process, especially one conducted by Ned Kock at a defense contractor in the United States [143] and another one by him about lessons learned from his doctoral research [144]. Those research questions have been determined by using observation of complex social interactions in the software development process [145, 146].

3.1 Research Questions

The following three research questions were addressed.

3.1.1 Research Question 1

Is it possible to formalize the eclectic approach so that it can be adopted by projects in the same way the traditional approach has been used?

3.1.2 Research Question 2

Is ESD well accepted by new practitioners?

3.1.3 Research Question 3

Does this pentagon represent an acceptable management tool? If not what criteria would be needed to make it one?

3.2 Research Method: Action Research

Action research produces changes in organization (*action)*, and improves understanding for researchers and the organization (*research)* [49]. This dual-goal, cyclic, responsive, and participative approach allows action and research together while removing the gap between researcher and practitioners [147].

It was important for this study to have responsiveness [147] for timely balancing between the plan-driven and agile methodologies. Action Research can be compared with other major research approaches, as shown in Table 13.

Table 13 Comparison with other major research approaches [144, 148]

Approach	Root	Key Data Collection Approaches	Uniqueness
Experimental research	Scientific practices of biologists and physicians	Numeric data collection with standardized statistical analysis procedures	Tests models with manipulation of variables over time
Survey research	Work of economists and sociologists	Questionnaires with closed-ended questions in a considerable sample organization	Permits quantitative evaluation
Case research	Business studies	Textual data collection with interviews in a few samples of organization intensively	Refers to Harvard Method
Action research	Studies of social and work-related issues	Observation and interviews in a few sample of organizations intensively	Provides dual goal of both improving the situation and relevant knowledge. Compared to Case Research, the action researcher is directly involved in change.

Another reason for choosing action research for this study is that the theoretical model for ESD currently is being developed. Action research is useful for this type of development [144]. Moreover, Baskerville and Wood-Harper [149] stated that AR is one of selective approaches currently available for acceptably researching for alternations in system development methodologies.

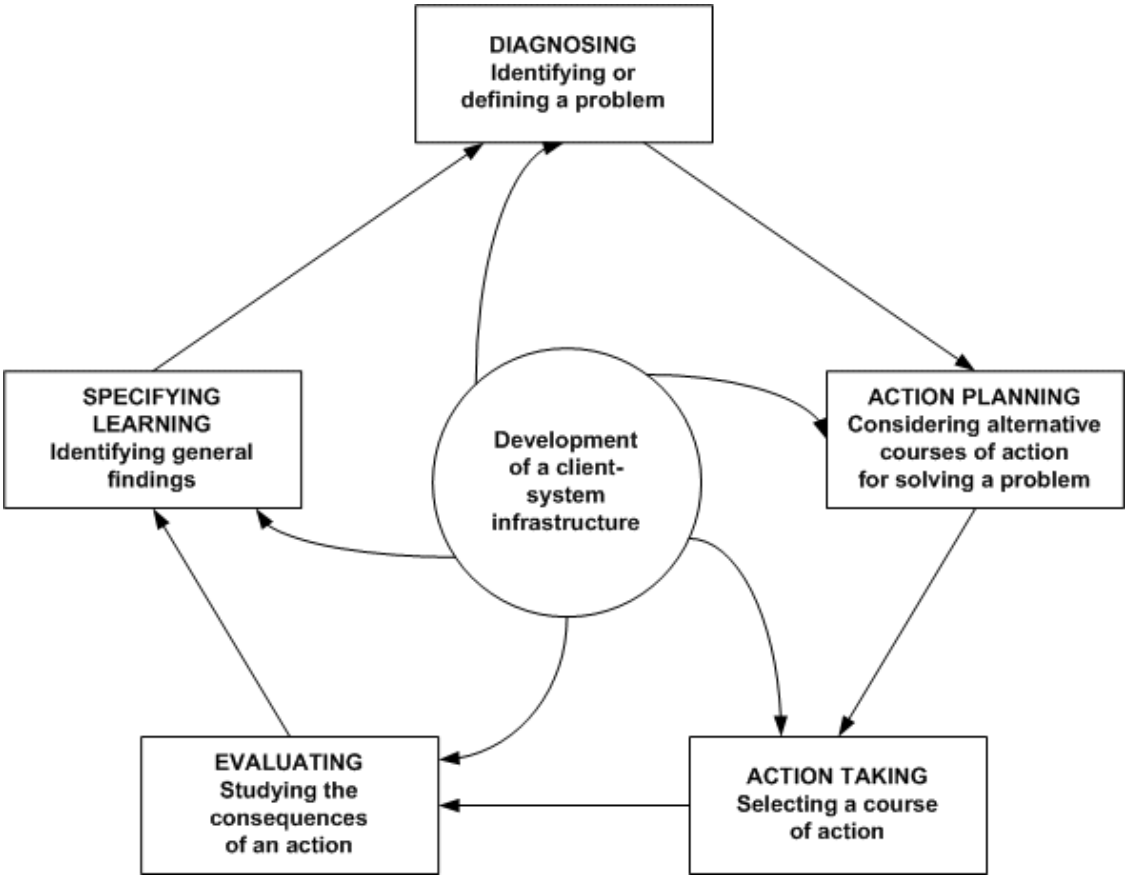
For this study, ESD was implemented for selected, real-industry projects in a cyclic manner. The research design was refined as the researcher and participants learned more.

Each cycle involves data collection, interpretation, and literature review. Learning from each project was applied to the next project to bring about change.

Action research was selected for previous studies relevant to this proposal for balancing between theory and practice. For example, success factors were researched by using AR in large-scale e-gov projects within an organizational, non-laboratory setting [150]. Other government projects selected action research to improve both the situation and the theory [143, 151]. Also, action research was selected recently to improve the software process by utilizing agile methodology [15, 16, 33, 34, 85, 152-157]. New approaches and models in information systems were introduced, refined, or validated using action research [15, 158, 159].

3.3 Research Procedures

Action research requires partnerships among the researchers, participants, and organizations. The researcher negotiated with the client organizations for participating projects. After a project was selected, it was necessary to educate the participants for action research and this study. Experience Interview, Program Assessment Form (A.K.A ESD Pentagon form), Best Practice Checklist, Result Interview, and Observation Notes were collected with other necessary data.



Diagnosing	Identify the primary problems that are underlying causes of the organization's desire of change
Action Planning	Specify organizational actions that should relieve or improve these primary problems. Indicate desired future state for the organization and the changes that would achieve such a state
Action Taking	Implement the planned action
Evaluating	Determine whether the theoretical effects of the action were realized and whether these effects relieved the problems
Specifying Learning	Direct the knowledge gained in the action research to audiences

Figure 29 Susman and Evered's AR Cycle [160] [149, 161]

This study followed Susman and Evered's AR Cycle, shown in Figure 29 [160]. Baskerville and Wood-Harper [149, 161] provide a summary of Susman and Evered's AR Cycle as shown in Figure 29.

In addition, for quality control purposes, each research cycle was planned, conducted, and evaluated by utilizing a framework for AR in IS studies proposed by Lau [162].

The client organization was informed that this study is performed under the guidance of Professor Theresa Jefferson of the Engineering Management and Systems Engineering Department, School of Engineering and Applied Science, George Washington University.

3.3.1 Selecting Client Organizations and Diagnosing

The first step of this study was to have the decision maker of the client organization understand that it is mutually beneficial to conduct this study. ESD has been applied and appreciated in Northrop Grumman Corporation projects [32], and another top 5 government IT contractor in the U.S.

However, it was open to other projects from other organizations or companies, depending on negotiation and criteria. Working with more than one organization is strategically recommended to avoid dependency on a single group, for example to avoid uncontrolled or unexpected delay due to the participating organization's various situations [144].

Three criteria for selecting client organizations and projects are: interest in applying agility by the government software development project, commitment from the team members and manager, and approval of the researcher's planned observation and interviews about the process and impact. All criteria were requested and evaluated before the researcher presented the research plan for the project to the decision maker in a client organization.

Table 14 Target projects

	Project Name	Application Type	Period	Resources	Status
1	Prior to this study	Java Web-based Application	6 months / 2003	9 Dev + 1 Mgr	Paper Published
2		Java/Progress/Oracle/XML Entreprise Solution	12 months / 2004	16 Dev + 1 Mgr	Completed.
3		Major Release of Enterprise Solution	12 months / 2006	29 Dev + 3 Tech Mgr + 1 Mgr	Completed
4	Pilot	ColdFusion Application	12 months /2008	5 team members	Part of this study
5		Java Web-based application	Multi years	5 team members	
6		Enterprise Portal	Multi years	20 Team members	
7	Project I	Multi agency, indefinite delivery/indefinite quantity (IDIQ) contract	9 years /2004 – 2013	200 program members, 35 sub projects	
8	Project II	Enterprise system integration	5 years / 2008 - 2014	20 team members	

The number of cycles drives the length of the research timeline and the level of research efforts. Previous doctoral studies have used between two and four action research cycles. Teasdale [163] conducted two cycles of action research for his doctoral study on design and implementation of an enterprise learning system. Kock [144] conducted four cycles of action research.

3.3.2 Action Planning

The next step is action planning for the ESD target project, which may be based on research results from a previous project. The following fundamental questions were asked during this step:

- What outcomes does the client organization wish to achieve?

- What actions does the researcher think will achieve the outcomes?

At this stage, the researcher will collect the participants' previous work experiences with SDLC and their expectations from this study. The researcher will provide SDLC and ESD training sessions to the members of the project and facilitate how to use the ESD Pentagon Form. To avoid any biases, no SDLC suggestions will be offered.

3.3.3 Action Taking

This step implements ESD for a project. Section 2.2 Eclectic Software Development describes the current implementation of ESD, noting that it has evolved as this study has progressed.

The researcher met the client organization to continue facilitating and observing based on its schedule. The researcher balanced the research pace with the understanding that the "in the middle of the action" involvement of the researcher may be uncomfortable for some team members [144].

Action research recommends brief cycles. In fact, this study customized this action research procedure based on the client project schedule. There were overlaps between projects because some of projects took longer to be completed than scheduled for various reasons.

Because the organizational changes, including internal changes and environmental adaptation, occurred during the study, each cycle of the system development activity needed to incorporate the continuous changes [164, 165]. Most action research is participative and interventionist. Instead of just training the participant and watching the result, the participant had the opportunity to be interactive with the methodology. They modified it during use for their best result, allowing the chance for our understanding of the

theoretical underpinnings of the methodology to also be adjusted. It provided insight into how the participant thinks critically. For example, the initial project factors do not include security, but it was recommended that it be added by the participants during the study.

3.3.4 Evaluating by Collecting Data and Researching

The study collected four main types of data from multiple information sources. :

- interview notes about the participants' and organization's experiences and expectations in the beginning stage (see Appendix A Interview Notes about the Participants' and Organization's Experiences and Expectations)
- ESD Agile Program Assessment Form (also known as the Pentagon Form) and Best Practice Checklist (see Appendix B Agile Program Assessment Form and Best Practice Checklist)
- interview notes about participants' experiences of ESD and the results of the project (see Appendix C Interview Notes about Participants' Experiences with ESD and Results of the Project), and
- researcher's observation notes (see Appendix D Researcher's Observation Note).

Those appendixes were developed utilizing previous studies [166, 167].

Both interviews used semi-structured interviewing, which allows asking more information to increase researcher's understanding [166]. Using semi-structured interviews, the researcher collects data on a set of predefined variables and identifies other undefined emerging variables.

3.3.5 Specifying Learning (and Refining ESD)

Based on the collected data, the research questions were analyzed. At the end of each cycle, ESD was refined before being applied to the next project.

3.4 Project Schedule

Figure 30 presents the schedule for this study. The proposal was approved to conduct action research for 2 projects with two cycles each. The Action Research Project I was performed in 2009 and the Action Research Project II was performed in 2010. In addition to the approved study, the researcher conducted 3 pilot projects in 2008. The research progress and findings have been presented at international conferences.

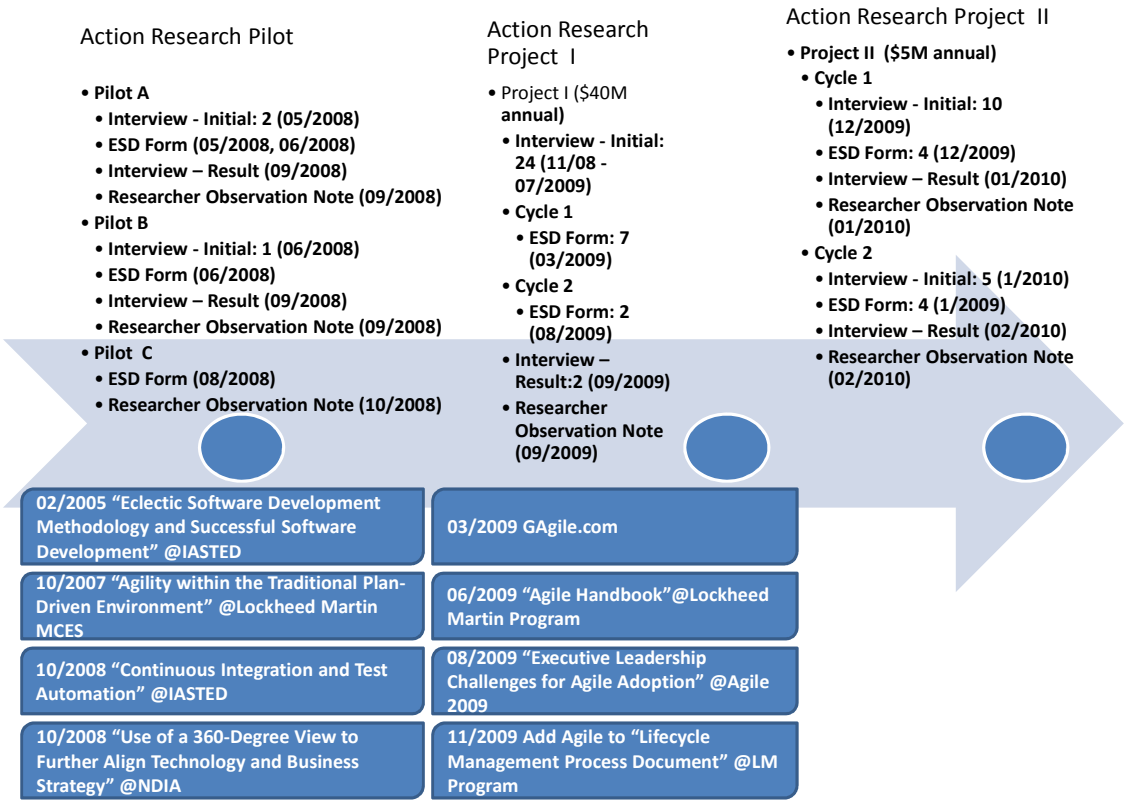


Figure 30 Proposed schedule

3.5 Project Steps

AR researcher's interest and action with confounding variables may bias the results [144]. The Figure 31 Project steps illustrates project steps that the project needs to follow so that it does not end up biasing the results. This is a protocol to conduct the research and guidelines of behavior for the project.

The project step explicitly illustrates the entire process using problem solving and research interest aspects. To maintain credibility, McKay and Marshall [168] suggest to design the AR study explicitly consisting of two interlinked cycles: a problem solving and a research interest cycles. This will facilitate the researcher in attaining the dual goals of AR throughout the research cycle.

In addition, the cycles within cycles utilizes the principle of reflective critique [169], also called regular critical reflection [49], to correct errors. Figure 31 illustrates the research process including opportunity for the participants to refine ESD during the Action Taking phase.

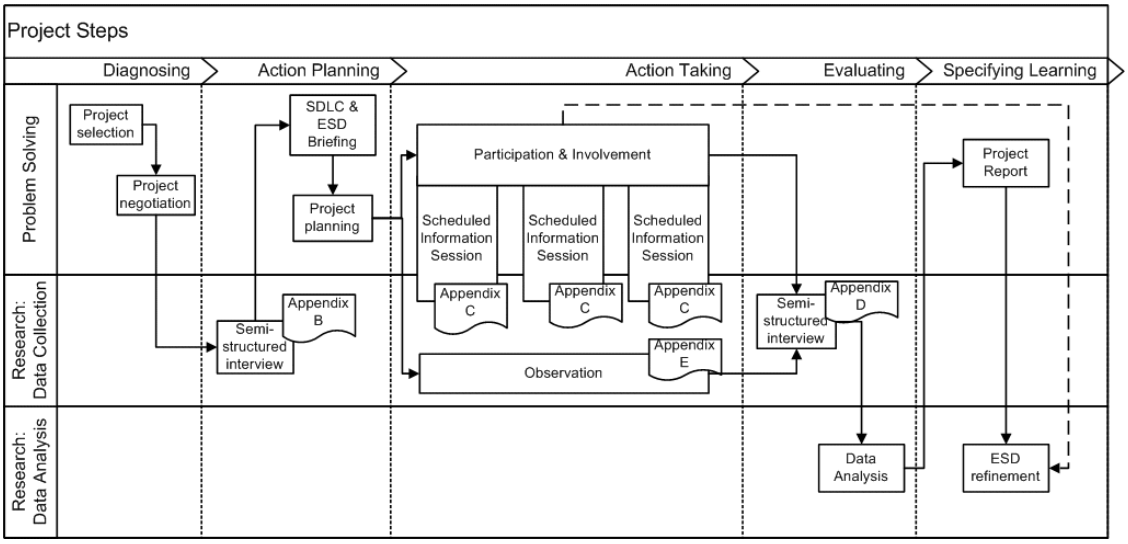


Figure 31 Project steps

3.6 Ethical Considerations

Due to the nature of AR about interacting in a non-laboratory environment, the researcher had to consider the ethical principles. These are listed by other AR practitioners [169, 170].

Chapter 4: Analysis and Findings

4.1 Overview

This study consists of 3 pilots and two comprehensive action research projects. The researcher enhanced and revised the ESD forms based on the responses from contributors during the first pilot (Pilot A). The researcher launched GAgile.com after the three pilots (Pilot A, B, and C) to provide commercial level accessibility and credibility and he introduced the Agile Best Practice Checklist based on the knowledge gained in the action research.

Two projects (I and II) are end-to-end action research projects, each consisting of two cycles. After Project I, the knowledge gained in the action research was presented at the Agile 2009 Conference, Chicago, USA.

This chapter provides the refinement progress of ESD during the study, the number of study participants, and the study results from project I and II followed by answers to the research questions.

4.2 Refining ESD

Table 15 depicts milestones for refining ESD during the course of this study. After 3 pilots, the Agile Best Practice Checklist was added and GAgile.com was launched. After Project I, the experiences were presented at the Agile 2009 conference, and the Agile Best Practice Checklist was improved based on the feedback. Also, the duration of Action Taking phase was significantly decreased from four months to one month. The researcher concludes that the refinement of ESD impacts the acceptance of ESD by new practitioners.

Table 15 Refining ESD

	Experience Interview (Appendix A)	**ESD Agile Program Assessment Form (Appendix B)**	**Agile Best Practice Checklist (Appendix B)**	**Result Interview (Appendix C)**	**Observation Note (Appendix D)**
Pilot Study A **2008-05**	Version 1 & Version 2 (Add 'Neutral' to the answers)	Version 1	No checklist available. Recognized a need for Agile BP Checklist.	Version 1 & Version 2 (Add 'Neutral' to the answers)	Version 1 & Version 2 (Removed fields regarding company information)
Pilot Study B **2008-06**	No changes	No changes	No checklist available	No changes	No changes
Pilot Study C **2008-08**	No changes	Version 2 (Remove Comments section. Add "Challenge-Approach-Action" sections)	No checklist available	No changes	No changes
2009 - 03	Launched GAgile.com				
Project I Cycle	No changes	Version 3 (Update	Version 3	No changes	No changes

	Experience Interview (Appendix A)	ESD Agile Program Assessment Form (Appendix B)	Agile Best Practice Checklist (Appendix B)	Result Interview (Appendix C)	Observation Note (Appendix D)
1 2009-03		instructions and introduce the agile BP checklist v.3)			
Project I Cycle 2 2009-08	No changes	No changes	No changes	No changes	No changes
2009-08	Presented at the Agile 2009 Conference, Chicago, USA				
Project II Cycle 1 2009-12	No changes	Version 4 (Use the agile BP checklist v.4.0. Move leadership from the first to last factor to evaluate. Add Agile Software Development Process. Delete Result column. Reformat.)	Version 4 (Reformat, Add Agile PMO)	No changes	No changes
Project II	No changes	No changes	No changes	No changes	No changes

	Experience Interview (Appendix A)	ESD Agile Program Assessment Form (Appendix B)	Agile Best Practice Checklist (Appendix B)	Result Interview (Appendix C)	Observation Note (Appendix D)
Cycle 2 2010-02					

4.3 Number of responses

Table 16 provides the number of responses for each research material.

Table 16 Number of responses

	Experience Interview	ESD Agile Program Assessment Form (also known as the Pentagon Form, Table 11)	Result Interview	Observation Note
Pilot Study A	2 Responses	2 Responses (including Researcher)	1 Response (Researcher)	1 Response (Researcher)
Pilot Study B	1 Response	1 Response (Researcher)	1 Response (Researcher)	1 Response (Researcher)
Pilot Study C	0 Response	1 Response (Researcher)	0 Response	1 Response (Researcher)

	Experience Interview	ESD Agile Program Assessment Form (also known as the Pentagon Form, Table 11)	Result Interview	Observation Note
Project I Cycle 1	24 Responses (including Researcher)	7 Responses (including Researcher)	2 Responses	1 Response (Researcher)
Project I Cycle 2		2 Responses		
Project II Cycle 1	9 Responses	4 Responses (From the Agile PMO including Researcher)	0 Response	1 Response (Researcher)
Project II Cycle 2	0 Responses	7 Responses	8 Response	1 Response (Researcher)

4.4 Results from Project I

4.4.1 Project I Overview

The participating program is an Indefinite Delivery Indefinite Quantity (IDIQ) contract for the U.S. Federal Government. The program was awarded on January, 2004 with $700 million ceiling budget for nine years. Their systems engineering services include high-performance computing, document management, business intelligence, geospatial solutions, application development, application security, and IT architectural support.

4.4.2 Experience Interview (Project I)

A total of 24 program leaders participated in the research. Figure 32 provides responses of their experiences and expectations.

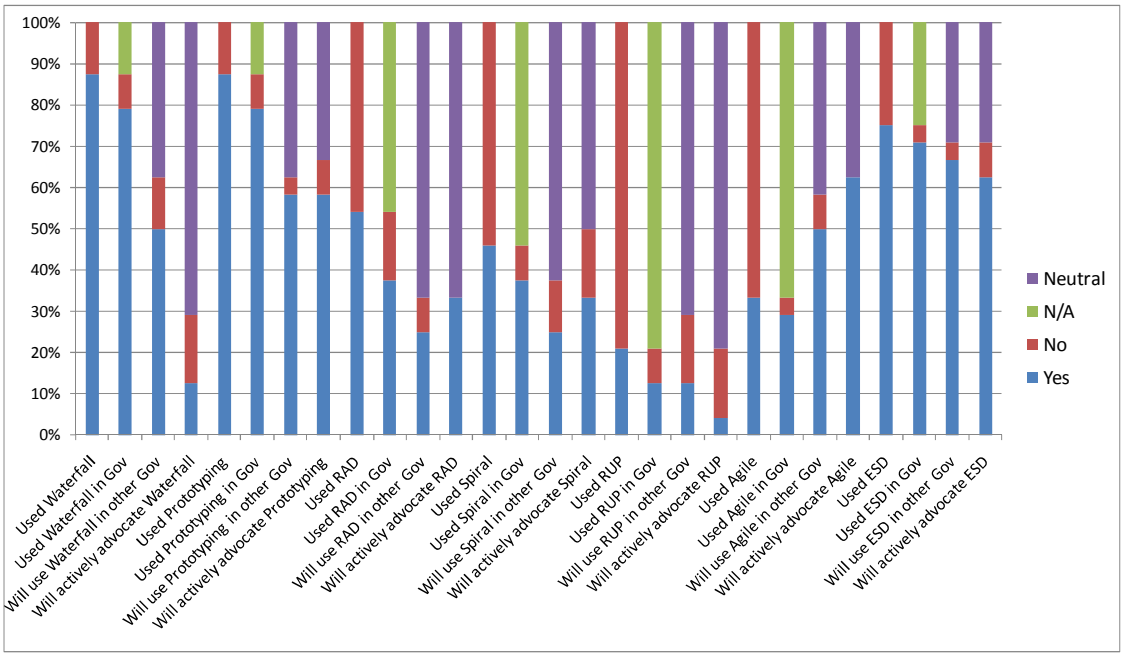


Figure 32 Experience Interview Results for Project I

Figure 33 illustrates responses about methodology used for previous projects.

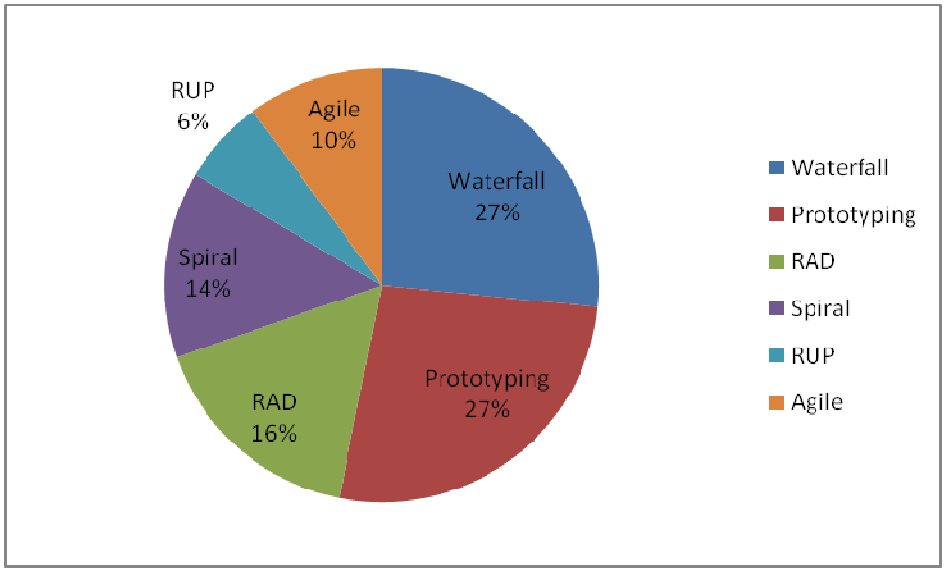


Figure 33 Methodology Used for Previous Projects (Project I)

Figure 34 illustrates responses about methodology used for previous Government projects.

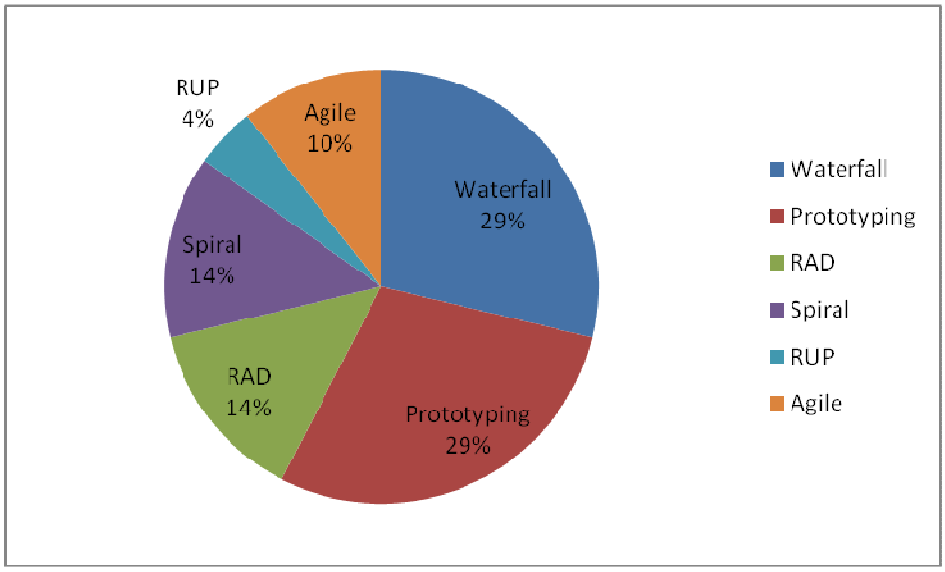


Figure 34 Methodology Used for Previous Government Projects (Project I)

Figure 35 illustrates responses about methodology planning used for other Government projects.

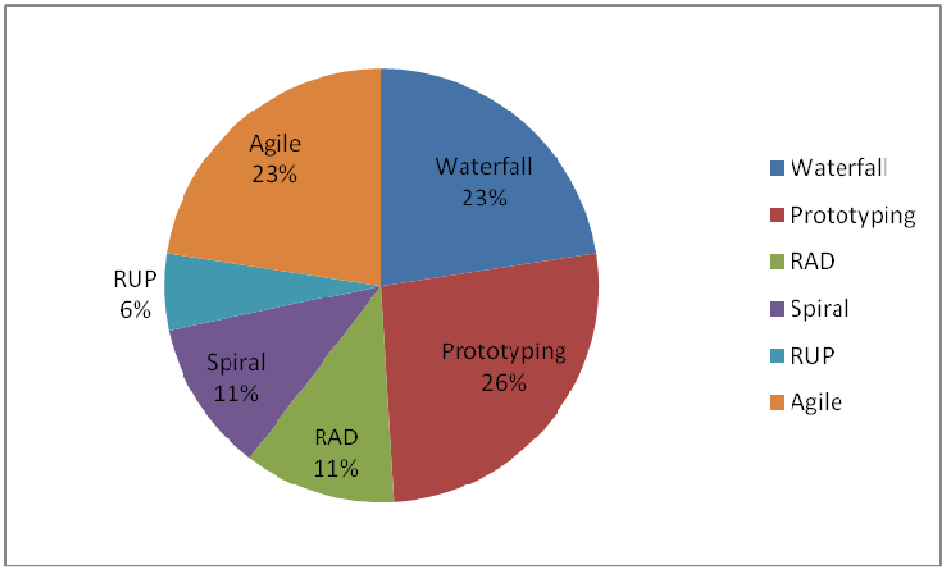


Figure 35 Methodology Planning to Use for other Government Projects (Project I)

Figure 36 illustrates responses about methodology planning used to actively advocate for other projects.

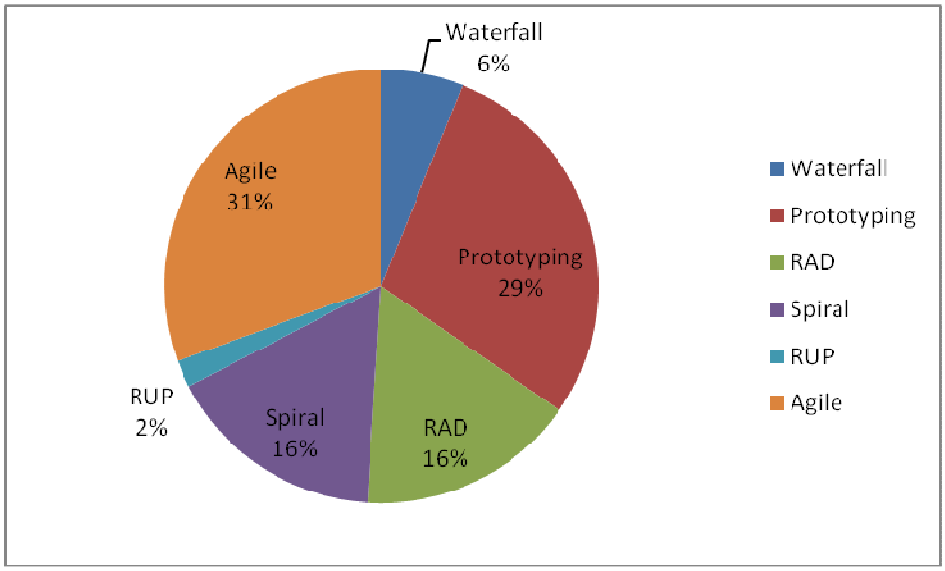


Figure 36 Methodology Planning to Actively Advocate (Project I)

The content analysis is a typically quantitative approach to the study of texts [171-175]. In addition to the above quantitative analysis, the researcher performed the following quantitative (summative) and qualitative (thematic) content analysis based on answers from the semi-structured interview.

- Summative content analysis [171] (Key word counts, Corpus analysis [172]): Compute frequencies of the occurrences of individual words and concordances (Table 17)

Table 17 Summative content analysis for initial interview (Project I)

Keywords	Frequency	ESD Project Factor	ESD Project Sub factor
agile	19	N/A	
government	18	Project	
method	13	Methodologies	
development	10	Methodologies	
process	10	Processes	
project	10	Project	
methodology	9	Methodologies	

customer	8	People	
think	7	N/A	
changes	6	Project	
reservation	6	N/A	
software	6	Tools	
believe	5	N/A	
team	5	People	
apply	4	Process	Model
aspect	4	N/A	
documented	4	Methodologies	All
experience	4	People	Skill Set
requirements	4	Methodologies	Requirement
agencies	3	People	Structure
contractor	3	People	Structure
design	3	Methodologies	Analysis and Design
early	3	Process	Model
following	3	N/A	
implementation	3	Process	Execution
involved	3	People	Acceptance
major	3	Project	Scope
people	3	People	All
plan	3	Process	Model
result	3	Project	
schedule	3	Project	Schedule
success	3	N/A	
type	3	N/A	N/A

- Thematic content analysis [172]– Directed approach analysis[171]: Capture dominant themes in a text using categories (coding schema) based on project factors of the Agile program assessment form.

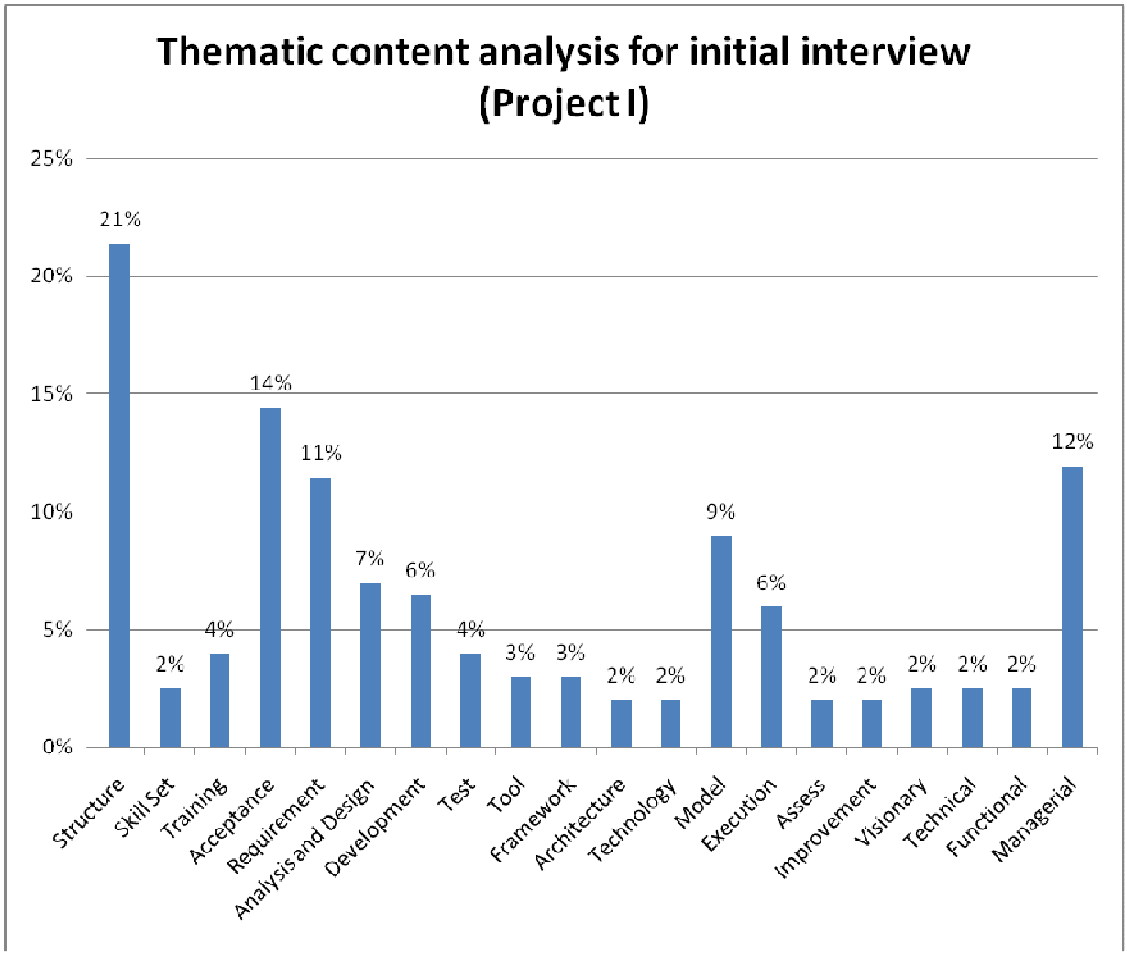


Figure 37 Thematic content analysis for initial interview (Project I)

4.4.3 ESD Agile Assessment and Agile BP Checklist (Project I)

The program selected Agile best practices to balance between Agile and traditional methodologies. Agile Program Assessment Form (version 3.0) and Best Practice Checklist were used to support the selection. These two assets have been utilized to review multiple best practices based on various factors.

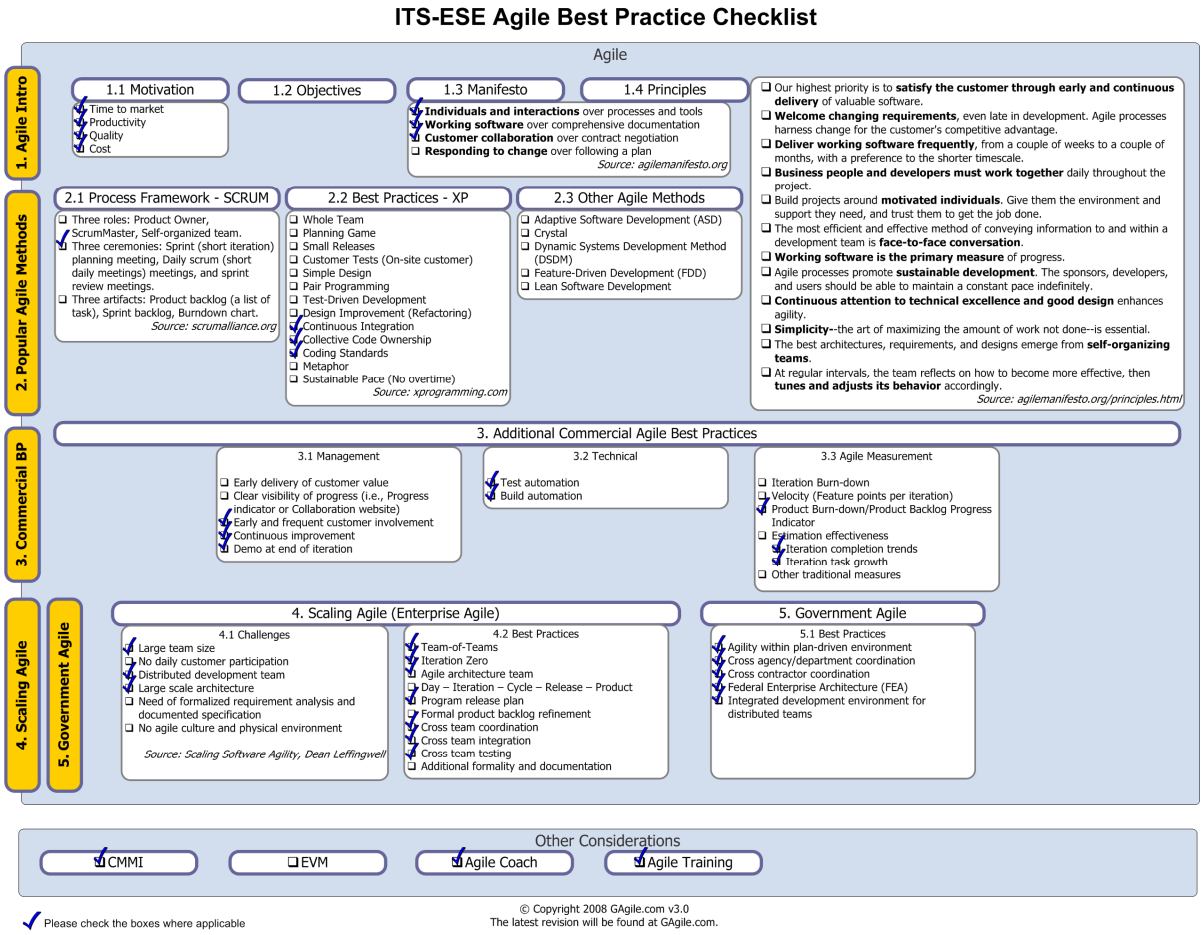


Figure 38 Agile Best Practices Checklist (Project I)

Figure 39 provides the rating by the assessment, and Figure 40, Figure 41, and Figure 42 provide summative analysis result of the assessment.

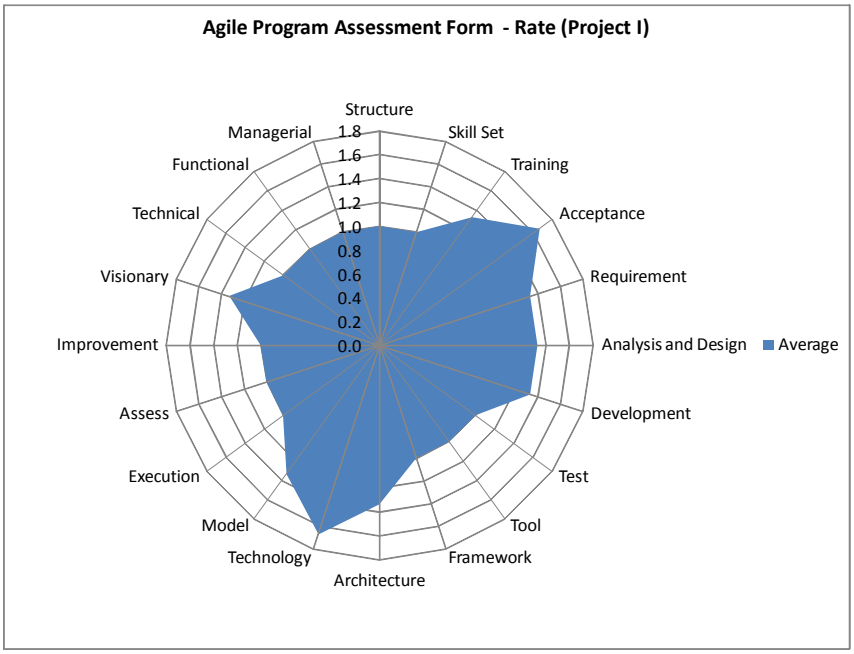


Figure 39 Agile program assessment form – rate (Project I)

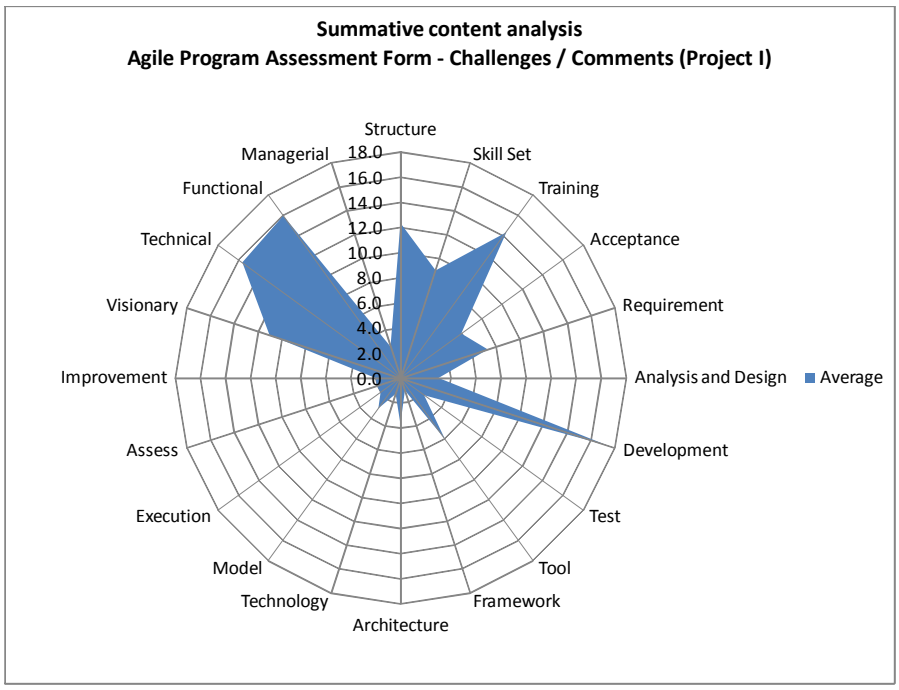


Figure 40 Summative content analysis: Agile program assessment form - challenges/comments (Project I)

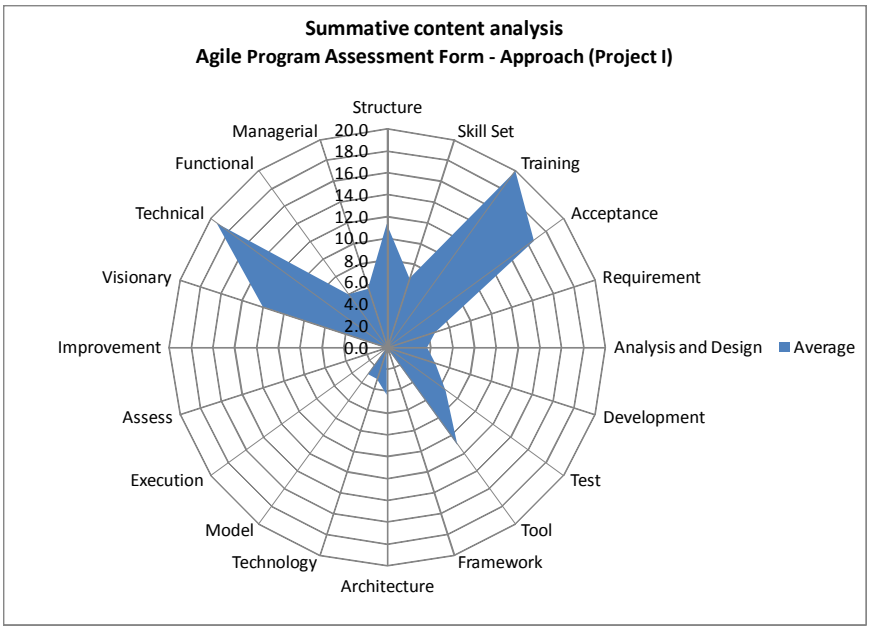


Figure 41 Summative content analysis: Agile program assessment form – Approach (Project I)

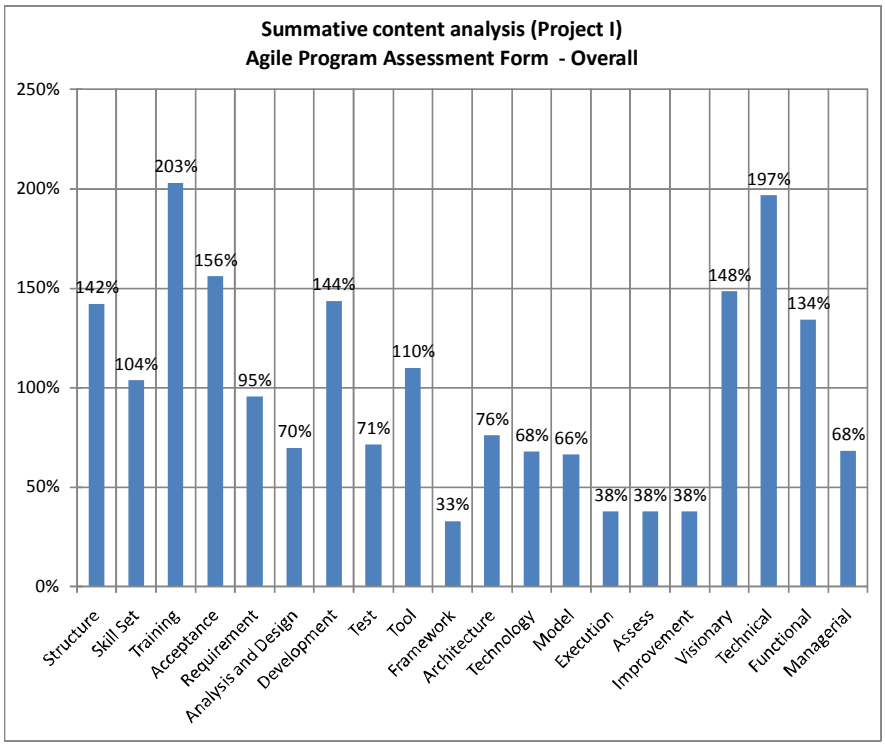


Figure 42 Summative content analysis: Agile program assessment form – Overall (Project I)

4.4.4 Result Interview (Project I)

Two key individuals, who participated in the project, were interviewed.

ESD has been implemented to promote efficient responsiveness to changes (a limitation of the plan-driven method and a benefit of the agile method) and predictability (a benefit of the plan-driven method) for the government project (a limitation of agile). As Figure 43 indicates, they responded that the team partially or fully achieves these goals using ESD.

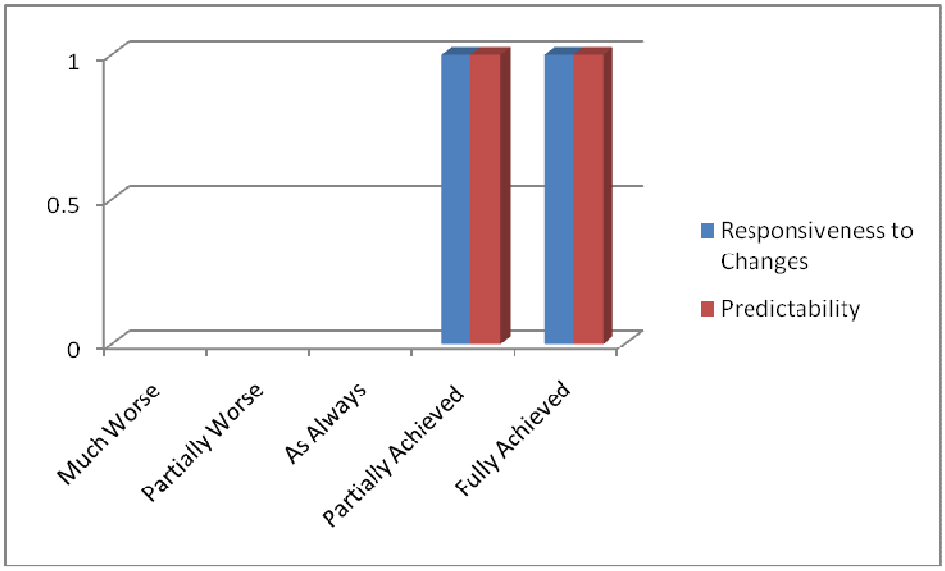


Figure 43 ESD Results (Project I)

As Figure 44 and Figure 45 indicate, both respondents said that they will use ESD again and will actively advocate for ESD in the future.

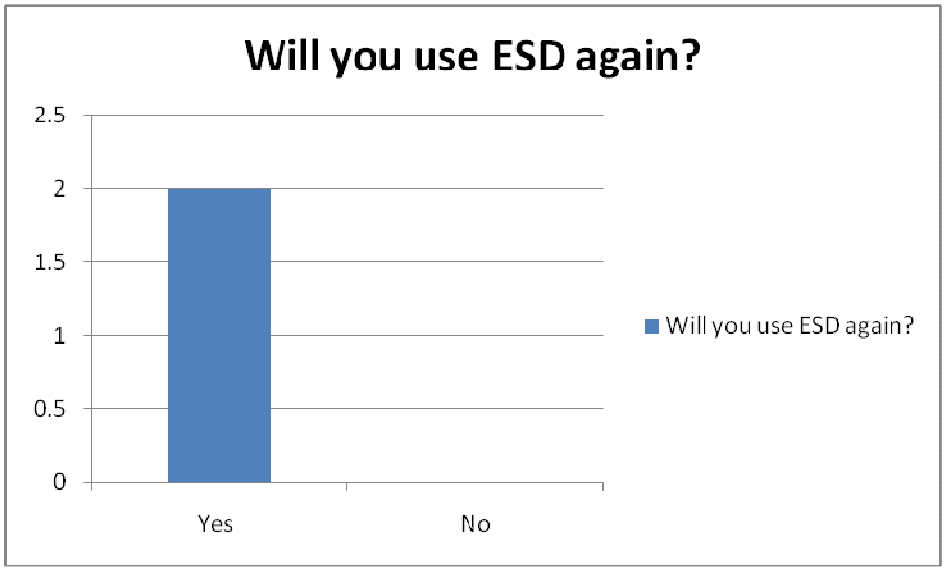


Figure 44 Will you use ESD again? (Project I)

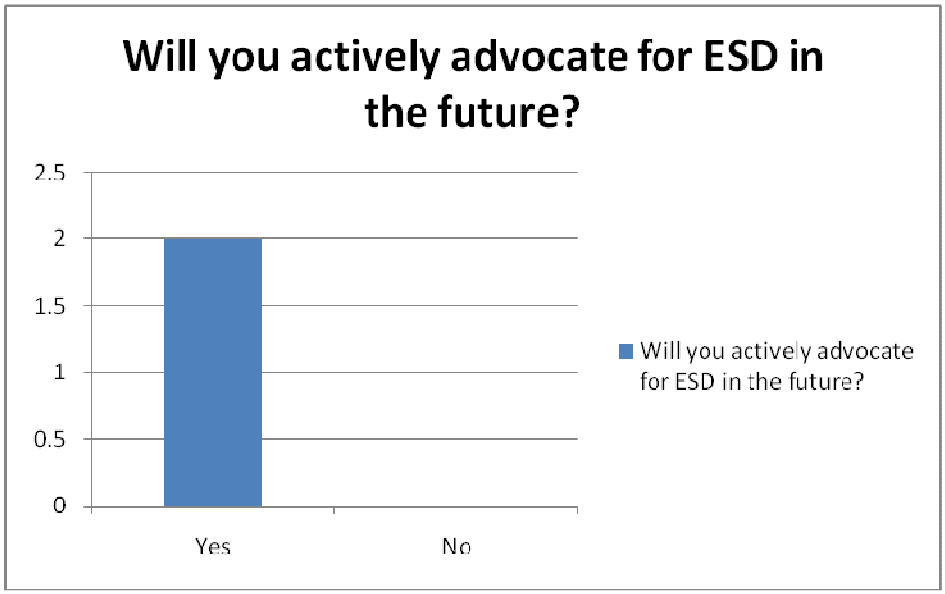


Figure 45 Will you actively advocate for ESD in the future? (Project I)

4.4.5 Observer's Note

Based on ESD, this program developed the Agile Software Development Handbook to provide information about the Agile Software Development Best Practices selected by the

program. The handbook describes how the program development teams can leverage the selected Agile Software Development Best Practices. This handbook also introduces the development team to the Agile Best Practices and tailoring process and how it can be used effectively with the program System Development Life Cycle (SDLC).

Some key activities are presented below.

- December, 2008: Presented ESD to CTO, Chief Software Engineer, and Tech Leads
- January, 2009: Developed Agile Handbook
- February, 2009: A Federal customer requested an Agile Process document.
- March, 2009: Presented ESD to Program Director, Deputy Program Director, and Portfolio Managers.
- May 2009: Conducted Agile Lunchtime Learning Session
- September, 2009: Developed Software Development Lifecycle Process Document including Agile as one of approved SDLC methodologies.

4.3.5 Research Challenges

The researcher faced the following challenges:

- Large number of stakeholders: In initiating agile adoption for the program, it took time to build a program-level partnership with many stakeholders.
- Existing process documents: The traditional program management office mandated the development of a formal process document for Agile, which would result in updating over 200 existing organizational process documents. While necessary, it delayed the adoption process, causing the team to lose momentum.

- Priority: The agile implementation was dependent upon program priorities, especially short term urgent issues.
- Leadership style: The agile implementation was impacted by the leadership styles of senior program leaders.
- Agile champion: The program agile champion did not have enough knowledge and experience with Agile.
- Budget: Program level implementation required an investment in training which was not budgeted.

4.5 Results from Project II

4.5.1 Project II Overview

This program, $5M for FY 2010, is an integration of four primary U.S. Environmental Protection Agency Superfund data collection, reporting and tracking systems. It will serve as the single official source of primary site activity data, records, and support documentation.

4.5.2 Experience Interview (Project II)

A total of 10 agile team members participated in the research. Figure 46 provides responses of their experiences and expectations.

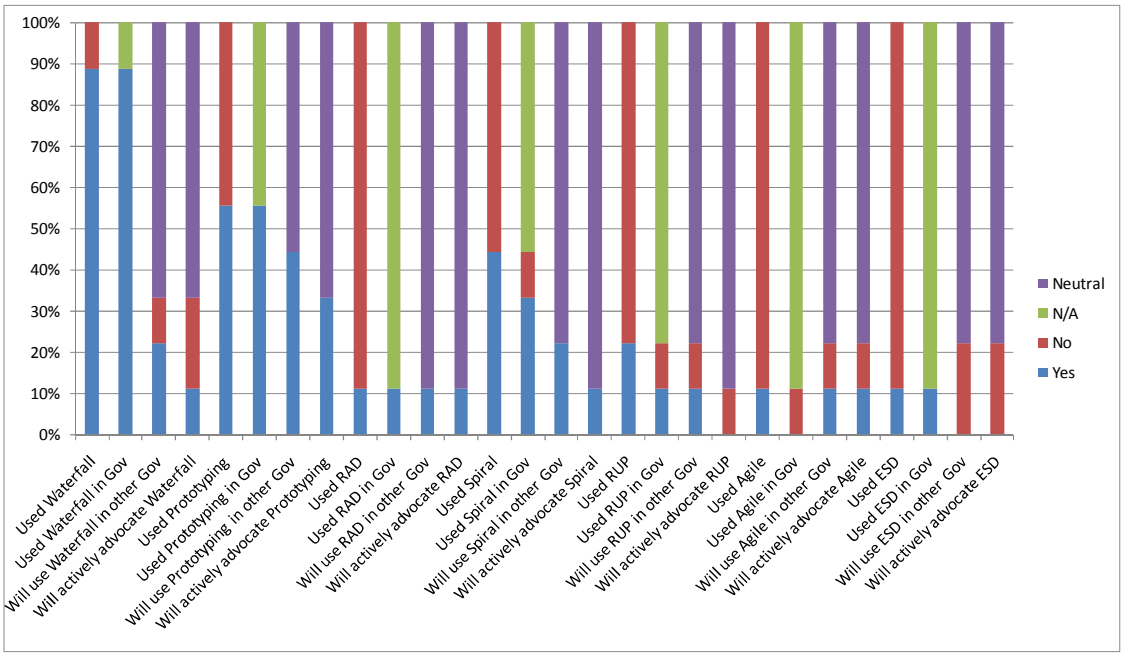


Figure 46 Experience Interview Results for Project II

Figure 47 illustrates responses about methodology used for previous projects.

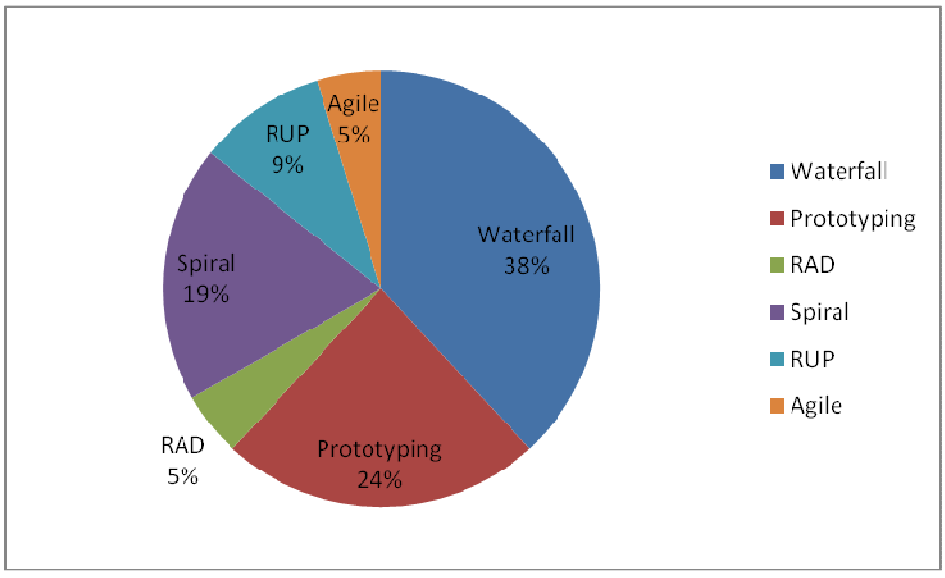


Figure 47 Methodology Used for Previous Projects (Project II)

Figure 48 illustrates responses about methodology used for previous Government projects.

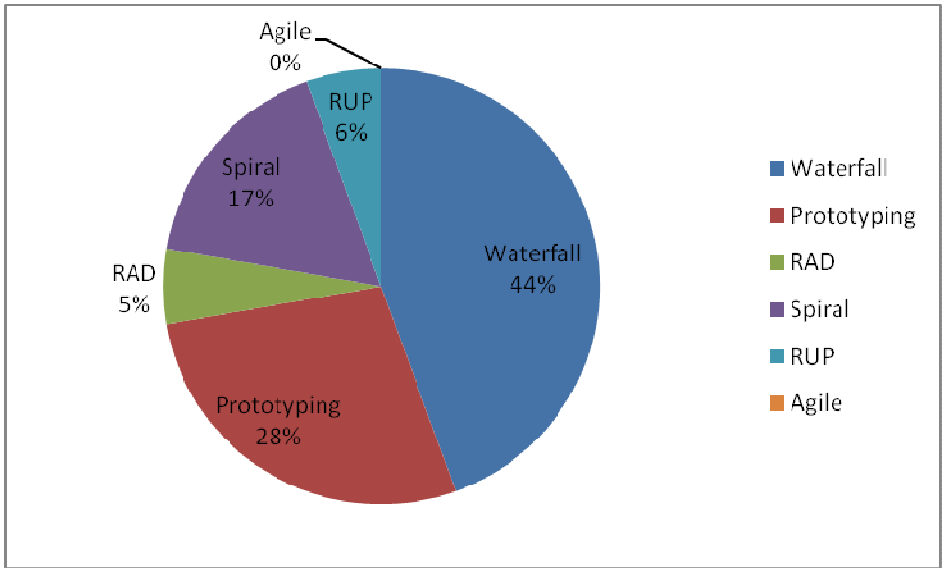


Figure 48 Methodology Used for Previous Government Projects (Project II)

Figure 49 illustrates responses about methodology planning to use for other Government projects.

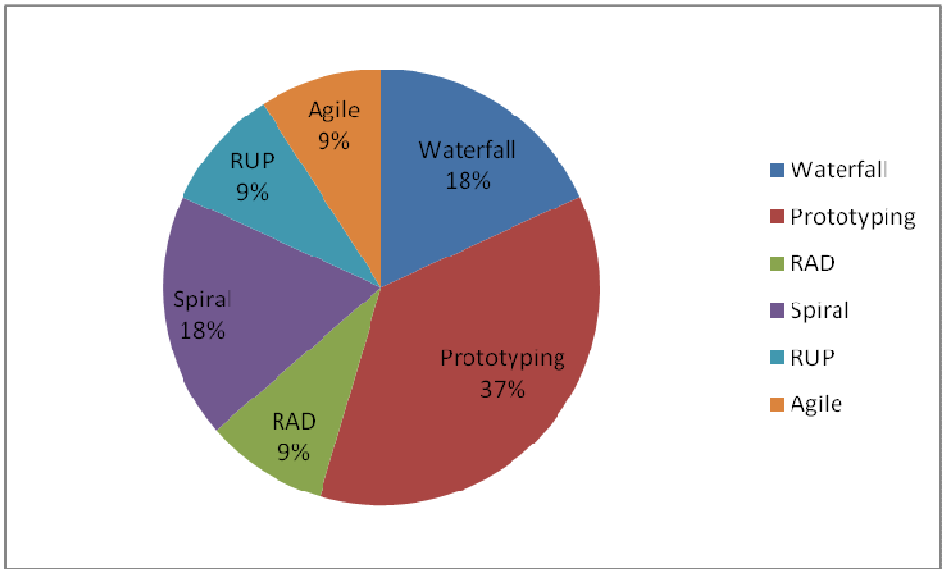


Figure 49 Methodology Planning to Use for other Government Projects (Project II)

Figure 50 illustrates responses about methodology planning to actively advocate for other projects.

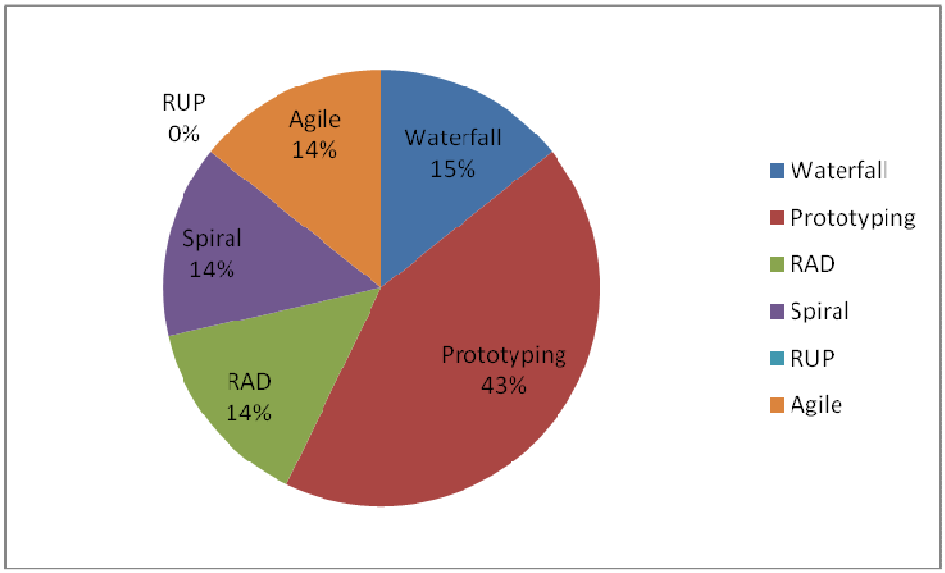


Figure 50 Methodology Planning to Actively Advocate (Project II)

As in Project I, a content analysis was conducted for Project II. Results are presented in Table 18 Summative content analysis for initial interview (Project II) and Figure 51 Thematic content analysis for initial interview (Project II).

Table 18 Summative content analysis for initial interview (Project II)

Keywords	Frequency	ESD Project Factor	ESD Project Sub factor
agile	7	N/A	
method / methodology	6	Methodologies	
code / coding	3	Tools	
development / developing	3	Methodologies	
difficult	3	People	Acceptance
government	3	Project	
apply	2	Process	Model
client	2	People	
customer	2	People	
docs / documenting	2	Methodologies	
prototype / prototyping	2	Methodologies	
reliance	2	Tools	
ability	1	People	Skill Set
adopt	1	People	Acceptance
artifacts	1	Process	
assumptions	1	People	
automatable	1	Process	
beginning	1	Process	
changing	1	Project	
cmmi	1	Process	
commercial	1	N/A	
commitments	1	People	Acceptance
contracts	1	Project	
cycles	1	Process	
db	1	Tools	
demo	1	Methodologies	
design	1	Methodologies	
developers	1	People	
directive	1	Leadership	

enhance	1	Process	Improvement
experience	1	People	Skill Set
feedback	1	Process	Improvement
flexibility	1	Methodologies	
involved	1	People	
loe	1	Leadership	Managerial
meeting	1	People	Structure
model	1	Process	Model
overhead	1	People	Structure
process	1	Process	
produces	1	Methodologies	
progress	1	Leadership	Managerial
qualify	1	Process	
requirement	1	Methodologies	Requirement
responsive	1	People	
scheduling	1	Leadership	Managerial
small	1	Project	
sponsor	1	People	
stakeholders	1	People	
suitable	1	Process	Model
suite	1	Process	Acceptance
tailored	1	Process	Model
team	1	People	
tests	1	Methodologies	Test
timely	1	Process	Execution
tracking	1	Leadership	Managerial
unit	1	Methodologies	Test
waterfall	1	Methodologies	

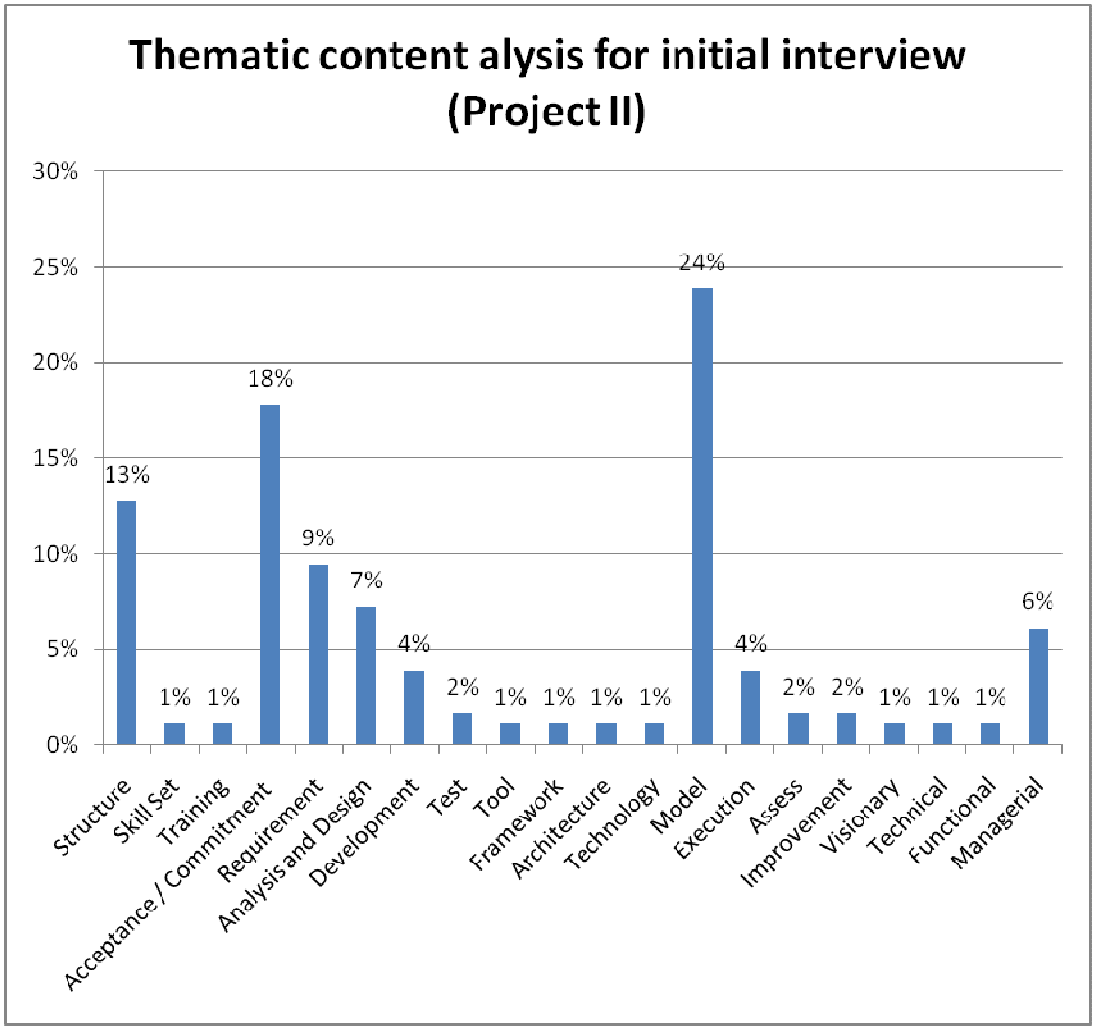


Figure 51 Thematic content analysis for initial interview (Project II)

4.5.3 ESD Agile Assessment Form and BP Checklist

The leadership team had a kick-off meeting to review and select an Agile process that balanced between the plan-driven and Agile methodologies. The Agile Program Assessment Form (version 4.0) and Best Practice Checklist were used to support the selection. Figure 52 is an artifact from the kick-off meeting; Figure 53 is the agile process overview, prepared as part of the Agile Procedure Document approved by the Organization

Process Group; and Figure 54 is the initial version of the Agile Technical Framework based on the kick-off meeting.

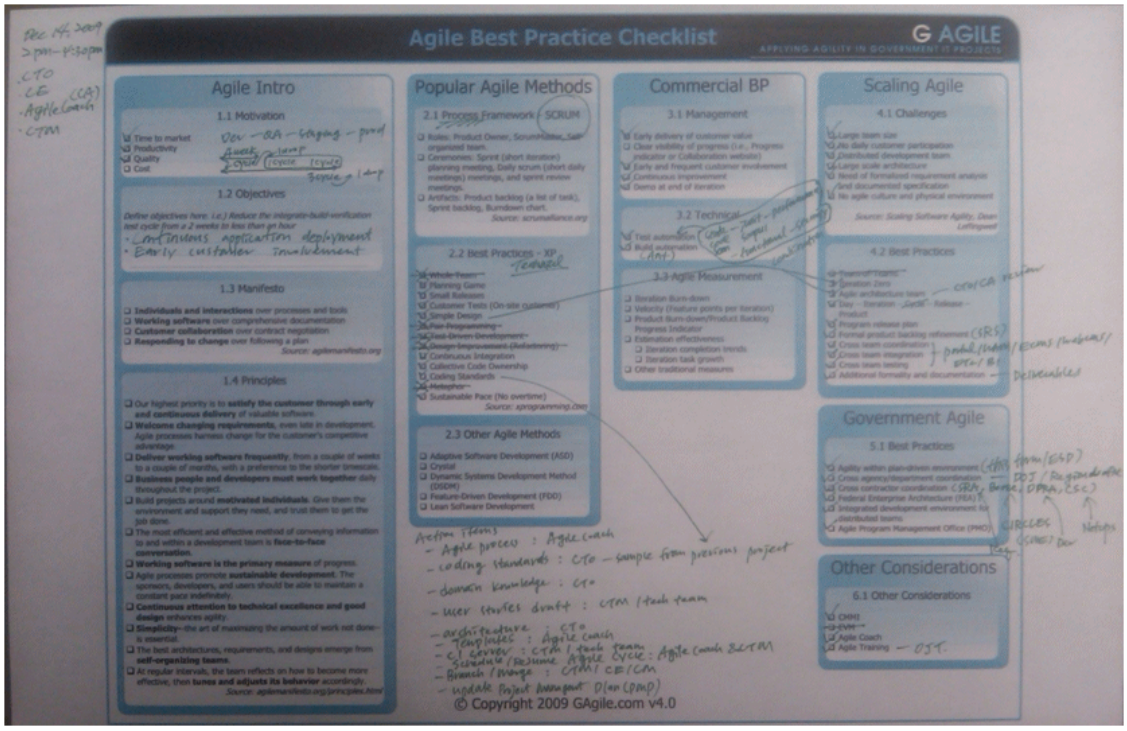


Figure 52 Agile Best Practices Checklist (Project II)

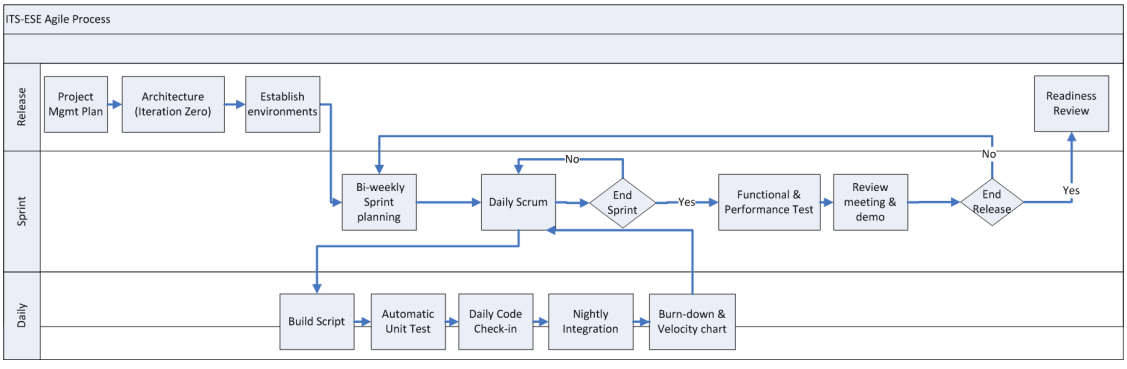


Figure 53 Agile Process based on Agile Best Practices Checklist Review (Project II)

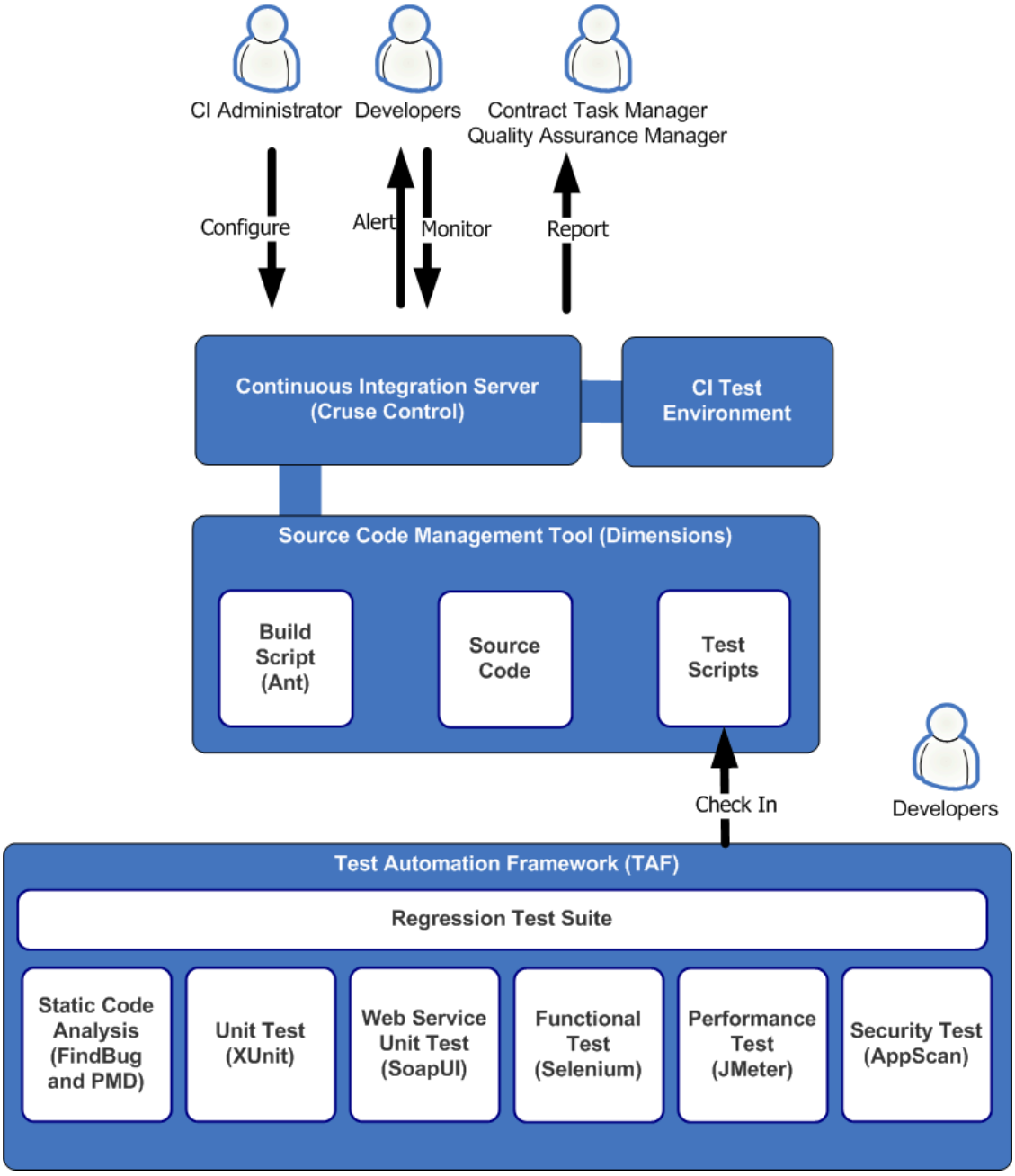


Figure 54 Agile Technical Framework based on Agile Best Practices Checklist Review (Project II)

Figure 55 provides the rating by the assessment, and Figure 56, Figure 57, and Figure 58 provide summative analysis of the assessment.

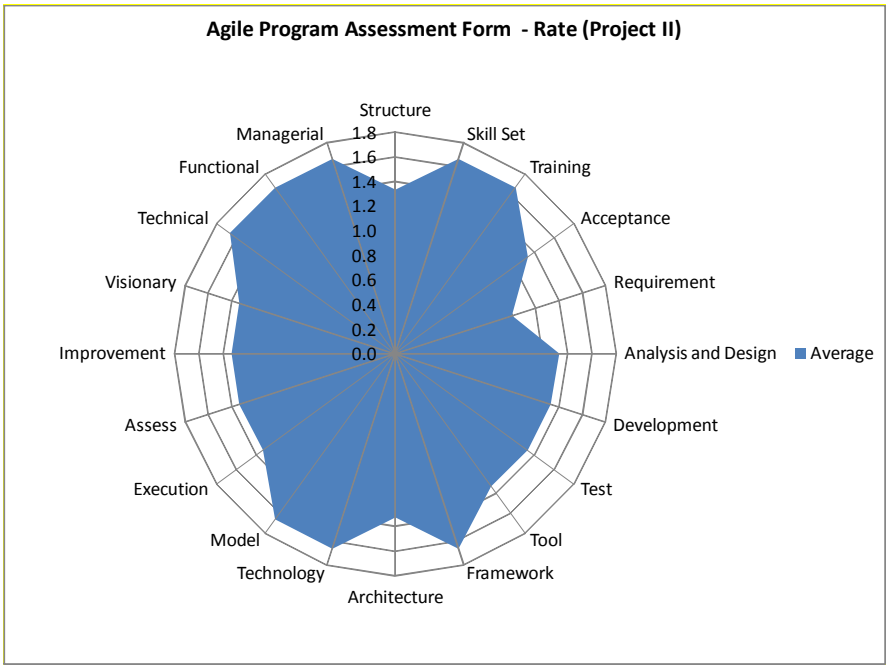


Figure 55 Agile program assessment form – rate (Project II)

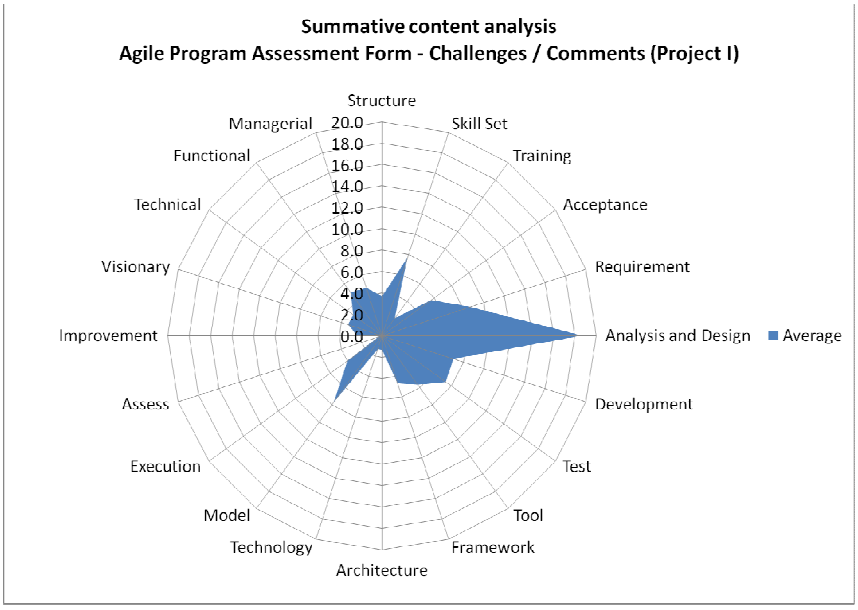


Figure 56 Summative content analysis: Agile program assessment form - challenges/comments (Project II)

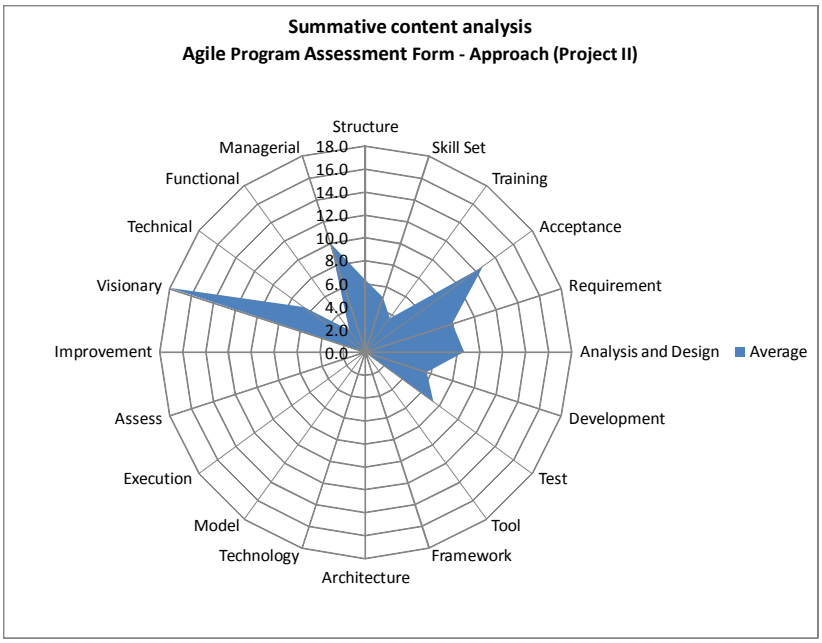


Figure 57 Summative content analysis: Agile program assessment form – Approach (Project II)

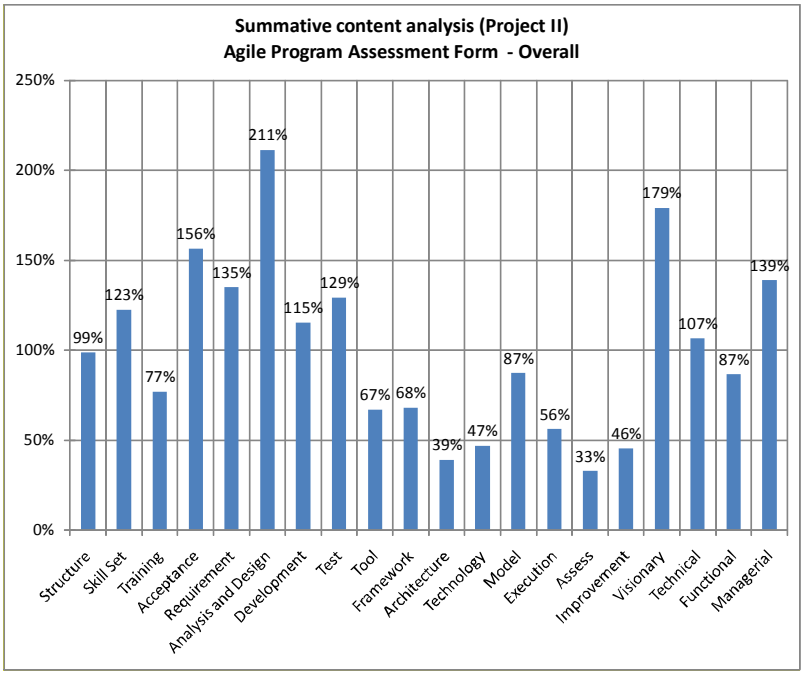


Figure 58 Summative content analysis: Agile program assessment form – Overall (Project II)

4.5.4 Result Interview

Eight key individuals, who participated in the project, were interviewed.

ESD has been implemented to promote efficient responsiveness to changes (a limitation of the plan-driven method and a benefit of the agile method) and predictability (a benefit of the plan-driven method) for the government project (a limitation of agile). As Figure 59 indicates, they responded that the team partially achieved these goals using ESD.

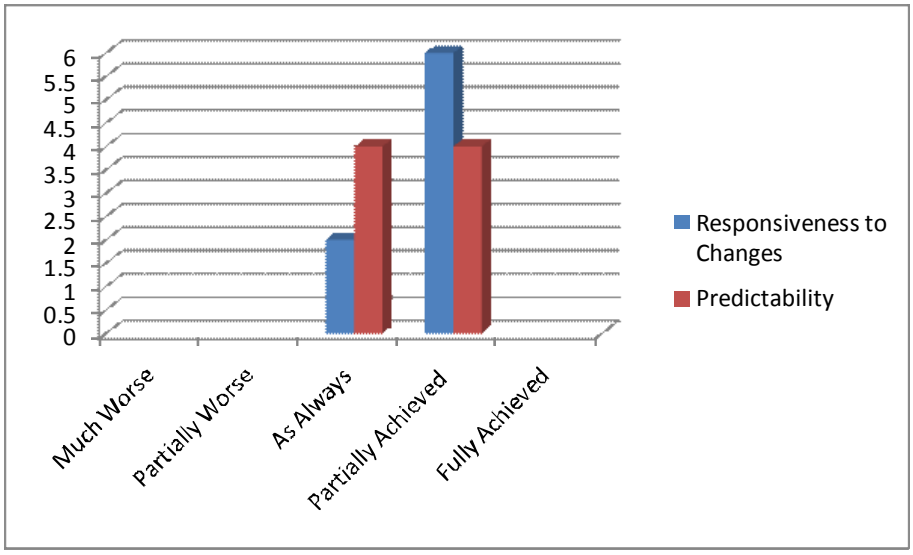


Figure 59 ESD Results (Project II)

As Figure 60 and Figure 61 indicate, 5 respondents said that they will use ESD again while 3 were neutral to using it again. With regard to actively advocating for ESD use in the future, 3 respondents said they would actively advocate for ESD usage while 5 respondents were neutral.

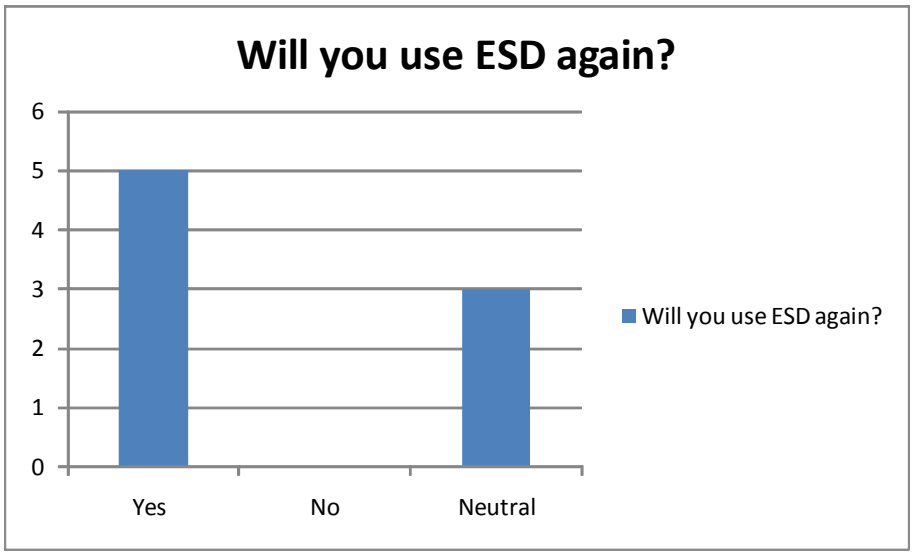


Figure 60 Will you use ESD again? (Project II)

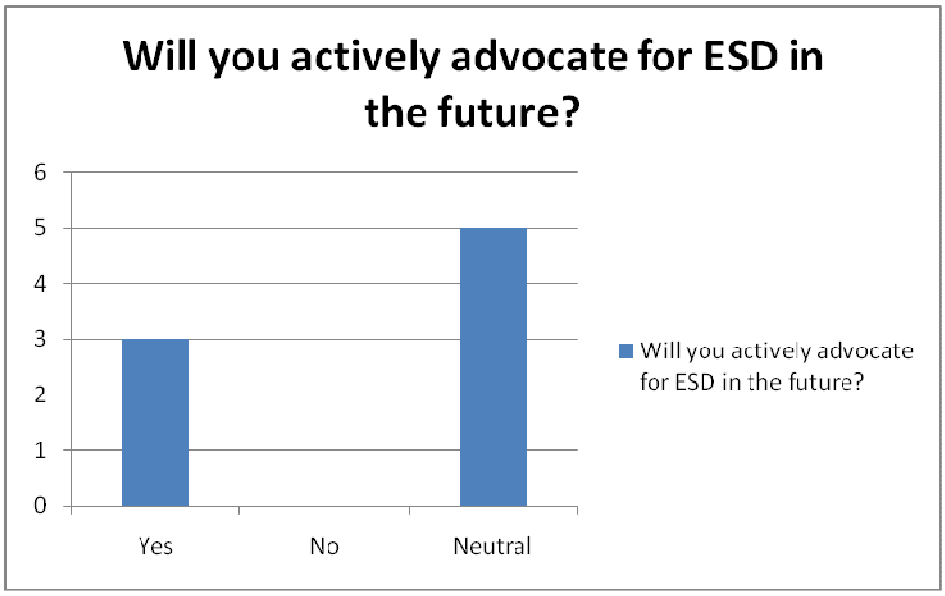


Figure 61 Will you actively advocate for ESD in the future? (Project II)

4.5.5 Observer's Note

Based on ESD, this development team adopted and implemented the Agile Software Development.

Some key activities are presented below.

- December 14, 2009: Formed the Agile PMO and completed the Agile BP Checklist
- December 15, 2009: Drafted the Agile Process Document
- December 18, 2009: Performed proof of concept for a Continuous Integration tool
- December 23, 2009: Conducted an Agile Software Development Process Training to 10 cross-functional team members
- January 5, 2010: Held the first sprint meeting, Sprint 0 Planning meeting
- January 6, 2010: Held the first Daily Sprint Meeting (DSM)
- January 19, 2010: Sprint 0 Demo and Sprint 1 Planning meeting
- January 20, 2010: Presented Agile Sprint 0 briefing to the customer committee
- February 1, 2010: Sprint 1 Demo
- February 2, 2010: Sprint 2 Planning meeting

4.3.5 Research Challenges

The researcher faced the following challenges:

- Transition from a functional-based team to a feature-based team: The team structure was functional-based. Several mentoring and training session were conducted by the researcher.
- Lack of leadership: The technical lead had no experience using Agile on a large application development project. Additional technology leadership was added after two sprints.

- Misunderstanding of Agile: Some team members believed that Agile was about just cutting corners.
- Transition from multiple assignments to a dedicated team: During the transition, the team members were not dedicated to this development task. The researcher recommended a team-wide restructuring, with one group dedicated to the Agile project and another group dedicated to a non-Agile project.
- Understanding of team's capacity: The team kept requesting additional resources because they felt they were behind schedule due to a lack of personnel resources. However, due to the multi assignments, the team was unable to track planned versus actual hours per assignment or task. This resulted in the team questioning its true size and capabilities.

4.6 Testing of Research Questions

4.6.1 Research Question 1

Table 19 provides the research results of Research Question 1, 'Is it possible to formalize the eclectic approach so that it can be adopted by projects in the same way traditional approach has been used?'

Table 19 Testing of Research Question 1

Research Projects	Testing of Research Questions
Project I Cycle 1	YES A program level Agile handbook was developed based on ESD.
Project I Cycle 2	Yes

Research Projects	Testing of Research Questions
	A program level SDLC Process document including Agile was developed based on ESD. Knowledge gained from Project I was presented at Agile 2009 Conference as a solution to executive leadership challenges. A corporate vice present (VP) from $31 billion Fortune 100 company sent an email to the researcher that 'Yours was the best presentation I attended. Would you be able to provide feedback on our AGILITY pitch to our C level, particularly our CIO?' Comments based on ESD and materials of ESD were provided. Presented as part of a corporate wide Software Learning Series.
Project II Cycle 1	YES Agile process document was developed based on ESD. Agile technical framework based on ESD was selected.
Project II Cycle 2	YES Agile PMO was formed. Lifecycle Selection Document was updated to include Agile based on ESD. Agile Process Document was developed based on ESD. The team conducted Agile Software Development Process Training based on ESD. Planning meeting, daily scrum meetings, internal demo, and customer

Research Projects	Testing of Research Questions
	demo were scheduled and performed. The program director reviewed the accomplishments from Sprint 0; she was pleased and approved continuing Agile based on ESD.

4.6.2 Research Question 2

Table 20 provides the research results of the Research Question 2, 'Is ESD well accepted by new practitioners?'

Table 20 Testing of Research Questions 2

Research Projects	Testing of Research Questions
Project I	Based on the interview with 24 participants as illustrated in Figure 62, 18 of them (75% compared to 55% average from other methods) used ESD-like method (applying process fragments from various methods into one project), 17 of them (71% comparing to 46% average from other methods) used ESD-like method n government, 16 of them (66% compared to 37% average from other methods) will use ESD-like method for other government projects, and 15 of them (63% compared to 34% average from other methods) will actively advocate for ESD-like method n the future. After performing ESD, 2 of the key stakeholders responded they (100% compared to 37% average from other methods) will use ESD for other government projects, and all of them (100% compared to 34% average from other methods) will actively advocate for ESD use in the future.
Project II	Based on the interview with 9 participants as illustrated in Figure 64, 1

Research Projects	Testing of Research Questions
	of them (11% compared to 39% average from other methods) used ESD-like method (applying process fragments from various methods into one project), 1 of them (11% compared to 33% average from other methods) used ESD-like method in government, none of them (0% compared to 20% average from other methods) will use ESD-like method for other government projects, and none of them (0% compared to 13% average from other methods) will actively advocate for ESD-like method in the future. After performing ESD, 8 of key team members responded that 5 of them (63% compared to 20% average from other methods) will use ESD for other government projects, and 3 of them (38% compared to 13% average from other methods) will actively advocate for ESD in the future.

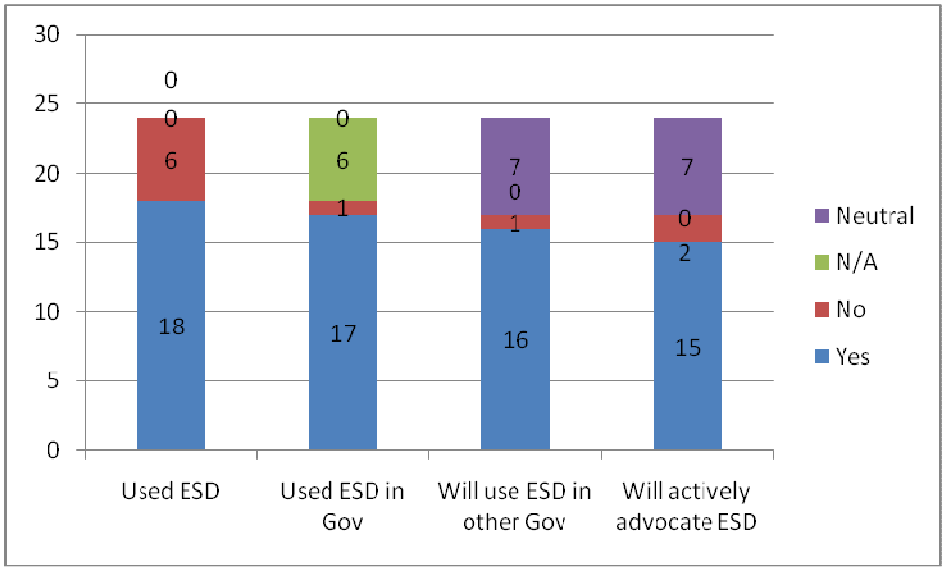


Figure 62 Experiences of ESD-like (Project I)

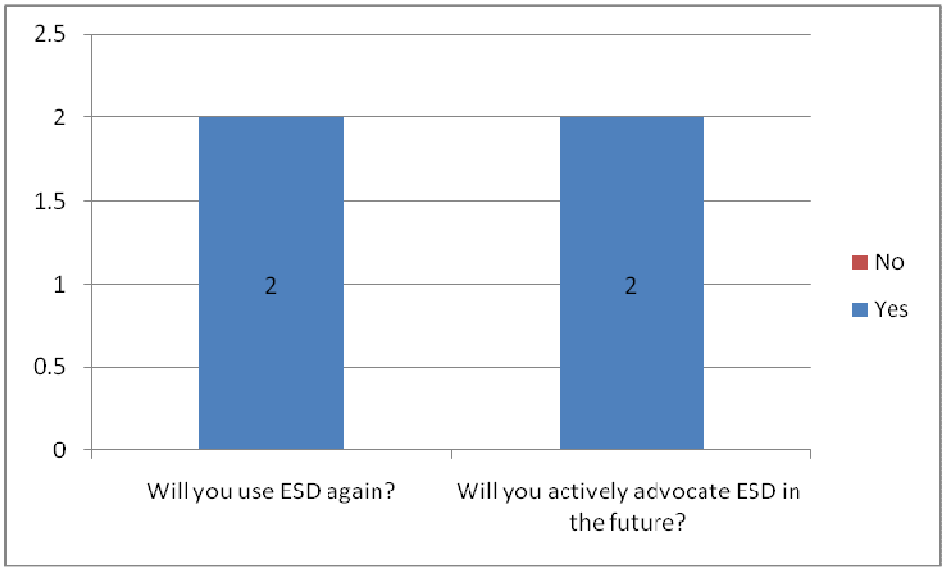


Figure 63 Experiences of ESD (Project I)

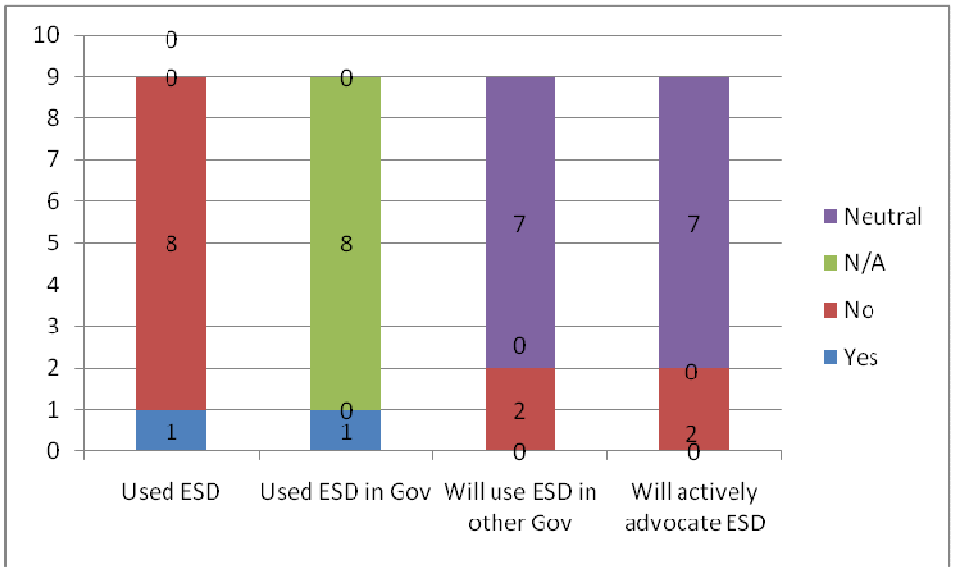


Figure 64 Experiences of ESD-like (Project II)

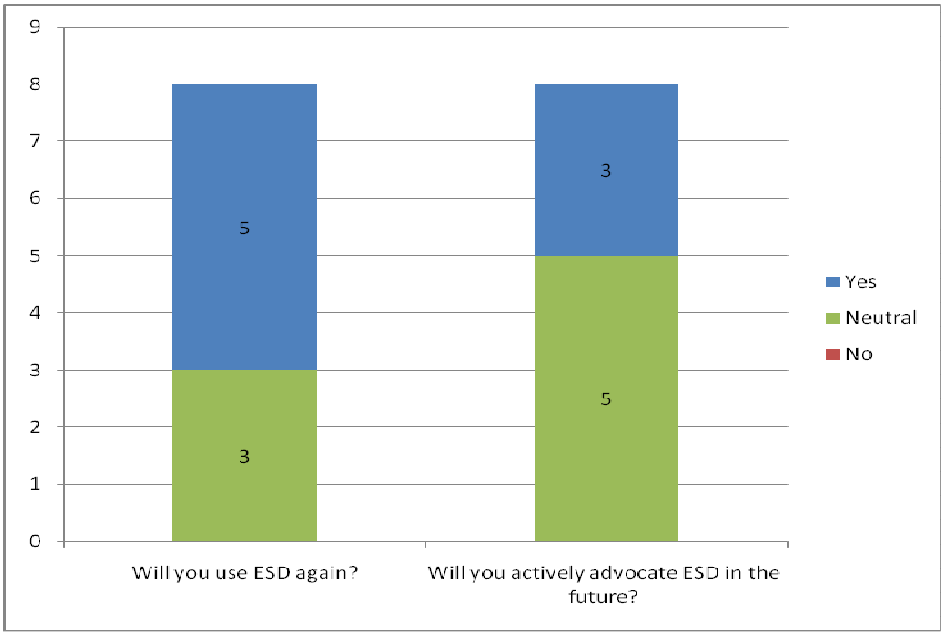


Figure 65 Experiences of ESD (Project II)

4.6.3 Research Question 3

Table 21 provides research results for Research Question 3, 'Does this pentagon represent an acceptable management tool? If not, what criteria would be needed to make it one?'

Table 21 Testing of Research Question 3

Research Projects	Testing of Research Questions
Pilots	The ESD Agile Program Assessment Form (also known as ESD pentagon) was changed based on the three pilot studies. The five assessment factors were not changed, but a comprehensive, practical, easy to use Agile Best Practice Checklist was added. It became a more useful management tool.

Research Projects	Testing of Research Questions
Project I Cycle 1	One reviewer says that his organization has not implemented any agility, and the rating is 0 for all five factors. The ESD Assessment Form itself may not be the main management tool. Therefore, the BP Checklist and Assessment Form are used together as a single management tool.
Project I Cycle 2	Updated based on the comments. Rearranged the evaluation factor. Enhanced the Assessment Form and BP Checklist to provide a commercial look and feel.
Project II Cycle 1	Version 4 was developed to provide a well-accepted format.
Project II Cycle 2	Version 4 was used continuously. Very well accepted.
Overall	Agile Best Practice Checklist was added. Usability and format was enhanced. Project factor (Scope, Schedule, Budget, Quality, and Response to Changes) was added.

Chapter 5: Conclusions and Recommendations

5.1 Conclusions

The private sector has widely utilized Agile Methodology for a number of years, and the government sector has started adopting this adaptive method. Compare to process-oriented and predictive traditional plan-driven methodology, agile methodology is described as people-oriented and adaptive. This study provided many references of limitations of authentic agile methods applied to government software development. The government sector manages software development projects differently from the private sector and faces unique challenges. Change is often not embraced, especially when each module or task is delivered from different contractors. The government usually requires a well-defined, planned, controlled, auditable, and tested project.

The ESD approach provides strategic and tactical best practices for government programs including data collection tools and analytical processes to balance between traditional environment and Agile methodologies. The theory behind ESD is the selective use of the right tools, methodologies, processes, and human resources by project leadership at the right time in the software development lifecycle, within the confines and structures already defined for large-scale and contract-based government projects. ESD was initially developed in 2004 by a software development team at Northrop Grumman Mission Systems for U.S. Department of Defense projects. It is not a proprietary methodology. As a pragmatic rather than a dogmatic framework, it has been applied to additional projects with continuous refinement.

Software development engineers and managers often select and use different portions of existing methodologies. Other peer-reviewed eclectic approaches, such as Situational

Method Engineering (SME), Living Software Development Process, and Other Agile-influenced Hybrid Approaches, share the philosophy and vision of ESD but present different implementation approaches. This study provides an action research-based framework to refine and validate other eclectic methodologies.

While this eclectic approach has been accepted by some government software development projects, the government sector demands the formal validation of this eclectic approach. To contribute to the software engineering industry and academic community with respect to meeting user needs, this study used action research to apply ESD practices to real government projects in a cyclic manner. In addition to the validating and refining ESD, this study answered three research questions about formalization, acceptance, and effectiveness of the ESD.

As this research needed a cyclic, responsive, and participative research approach, action research was utilized for actions to invoke change and research to increase the understanding. It combined theory and practice. This study was developed and conducted based on Susman and Wood-Harper's Action Research Cycle, and guidelines of previous action researches in software engineering, especially one by Ned Kock and lessons learned from his doctoral research, and Information System Action Research Framework by Lau. This study will contribute to future researches about a structured formal validation of new methodologies in the software engineering field.

This study consists of 3 pilots and two comprehensive action research projects. The researcher launched GAgile.com after the three pilots (Pilot A, B, and C) to provide commercial level accessibility and credibility. Project I and Project II are software engineering projects for the U.S. Federal Government. Project I is a $700 million nine-year

contract, and Project II had a $5 million budget for FY 2010. Following the conclusion of Project I, the knowledge gained in the action research was presented at the Agile 2009 Conference in Chicago, Illinois, USA. It was very well accepted by other large organizations including financial institutions.

Semi-Structured Experience Interview, Program Assessment Form, Best Practice Checklist, Result Interview, and Observation Notes were collected along with other necessary data. In addition to the above quantitative analysis, the researcher performed the quantitative (Summative content analysis) and qualitative (Thematic content analysis and Directed approach analysis) content analysis based on answers from the semi-structured interview. As part of the Project II, the researcher developed an Agile Process document and an Agile Technical Framework for the client organization.

Even though this study had an agreement from the executive leadership of the client organization, it took more than 3 years to complete the study including the planning stage. When an action research study requires not only large number of participants but also large size projects, it requires persistent and long term commitment from the researcher. Also, conducting pilots were essential to understanding the client organization and communicating the process to the participant.

The following section 5.2 summarizes answers for the research questions.

5.2 Synthesis of Research Questions

Research Question 1, which was answered as yes, states that "Is it possible to formalize the eclectic approach so that it can be adopted by projects in the same way the traditional approach has been used?" Utilizing ESD formally, two large programs, including a $700M government program which was operated by a $42B revenue company,

have introduced and executed the agile method to the plan-driven formal organization. The ESD execution process was tested using three pilot programs and the two programs with two cycles each have validated that the formalized ESD can be adopted by projects in the same way traditional approaches have been used. For example, an agile handbook, a Software Development Life Cycle Model selection procedure document, and an agile process document were developed based on the ESD. Also, artifacts and templates including an experience survey, an agile best practices checklist, a project assessment form, a result survey, and an observation note have been adopted by the practitioners. The ESD was peer reviewed and presented at two international conferences, as well as cross-program level and corporate level webinars.

Research Question 2, which was answered as 70% positive, states that "Is ESD well accepted by new practitioners?" Approximately 35 % of the participant group responded that they will use or advocate for a software development methodology in average. Before the study, around 63% of the participant group responded that they will use or advocate for ESD-like (applying process fragments from various methods into one project) software development methodology After the study, 100% of the participant group responded that they will use or advocate for ESD software development methodology This group of participants in general has a neutral opinion of any software development methodologies, and a positive opinion of ESD-like software development methodologies. Following the study, this group of participants reported a more positive opinion of ESD. This study verified that ESD is well accepted by new practitioners.

From Project II, around 15 % of the participant group responded that they will, on average, use or advocate for a software development methodology. Prior to the study, none

of the participant group responded that they will use or advocate for ESD-like like (applying process fragments from various methods into one project) software development methodology After the study, 50% of the participant group responded that they will use or advocate for ESD software development methodology This group of participant in general has a negative opinion of software development methodology, and a very skeptical opinion of ESD-like software development methodology. Following this study, this group of participants reported a positive opinion of ESD. This study verified that ESD is well accepted by new practitioners.

However, the value of this quantitative analysis with Research Question 2 is very limited, and it can be used as only reference data. After the study, the research concluded that qualitative analysis with Research Question 1 and 3 provide more values to the software engineering field than what the quantitative analysis with Research Question 2 does.

Research Question 3 states that “Does this pentagon represent an acceptable management tool? If not, what criteria would be needed to make it one?" Based on the summative and thematic content analysis of semi-structured interview results, in addition to traditional quantitative analysis, this question was answered as YES. As a result a new assessment factor with four sub factors was added to increase the quality of the management tool. Also, utilizing Evaluating and Specifying Learning phases of action research, the research organization and ESD community has experienced a significant responsiveness and improvement in ESD and its tools.

5.3 GAgile Objective

Although the practitioner approved of the concept and content of ESD, it was clear from the result of the pilot study that the presentation format was important. The early developed methods including the Waterfall and spiral models are simple conceptually and are maintained by the academic community. However, recently developed methods including, RUP, XP, and Scrum benefit from being a more commercialized product, providing a reference model, training, certification, tools, a community, and conferences. ESD has been launched and renamed to GAgile to better serve for this practitioner.

Findings from this action research study provide not only a formalized empirical study of applying ESD to the real programs but also a practical and commercial-like process, templates, and structure.

5.4 Recommendation for Future Work

This action research has validated the formalized approach of ESD for real programs. Using this time-consuming action research method, this study has not only formalized the ESD method but also enhanced the concept, content, process, and templates. Using the ESD method, a plan-driven $700M program and another program in a $42B company adopted the Agile method. As a result, the GAgile product was launched. Based on this AR study, survey research and case research associated with training has to be planned to collect more empirical data from a large number of actual implantations of ESD, now GAgile.

A tool can be developed to generate a proposed process based on the experience interview, an agile best practice checklist, an assessment form, a result interview, and an observation note. It serves as a plug-in to the portfolio and project management tool, or the

agile lifecycle management tool. Such a tool can be used by the real program as action research.

References

1. Salamango, M. and J. Cunningham. *Leading the Agile Way: Duty. Honor. Delivery.* in *Agile 2006*. 2006. Minneapolis, MN.
2. Babuscio, J. *How the FBI learned to catch bad guys one iteration at a time.* in *Agile 2009*. 2009. Chicago, USA.
3. Morgan, D. *Covert Agile: Development at the Speed of…Government?* in *Agile 2009*. 2009. Chicago, USA.
4. McMahon, P.E., *Lessons learned using agile methods on large defense contracts.* CROSSTALK, The Journal of Defense Software Engineering, 2006. **19**(5): p. 25-30.
5. Sime, A. *How to sell a traditional client on an agile project plan*. in *Agile 2009*. 2009. Chicago, USA.
6. Lee, M.-G. *Executive Leadership Challenges for Agile Adoption.* in *Agile 2009*. 2009. Chicago, USA.
7. Eckstein, J., *Agility: coming of age*, in *Invited Talks in 6th International Conference, XP 2005*. 2005, Springer-Verlag GmbH: Sheffield, UK.
8. Scott, J., R. Johnson, and M. McCullough. *Executing Agile in a Structured Organization: Government*. in *Agile 2008*. 2008. Toronto, ON.
9. Cuellar, R. and J. York. *Succeeding with Agile in the US. Federal Government Arena*. in *Agile 2007*. 2007. Washington D.C.
10. Franklin, T. *Adventures in Agile Contracting: Evolving from Time and Materials to Fixed Price, Fixed Scope, Fixed Schedule Contracts*. in *Agile 2008*. 2008. Toronto, ON.
11. Wankerl, J. and B. Raines. *Making an Enterprise Agile Transition Happen in the Face of Federal Bureaucracy*. in *Agile 2009*. 2009. Chicago, USA.
12. Koehnemann, H. *Experiences Applying Agile Practices to Large Systems*. in *Agile 2009*. 2009. Chicago, USA.
13. Gnatz, M., et al., *The living software development process.* Software Quality Professional, 2003. **5**(3): p. 1-22.
14. Henderson-Sellers, B., *Method engineering for OO systems development.* Communications of the ACM, 2003. **46**(10): p. 73-78.
15. Henderson-Sellers, B. and M.K. Serour, *Creating a dual-agility method: The value of method engineering.* Journal of Database Management, 2005. **16**(4): p. 1-23.
16. Keenan, F. *Agile process tailoring and problem analysis (APTLY).* in *26th international conference on software engineering, 2004*. 2004: IEEE.
17. Lindvall, M., et al., *Agile software development in large organizations.* IEEE Computer, 2004. **December 2004**: p. 26-34.
18. Lycett, M., et al., *Migrating agile methods to standardized development practice.* IEEE Computer, 2003. **June 2003**: p. 79-85.
19. Ralyté, J. and C. Rolland. *An assembly process model for method engineering.* in *13th international conference on advanced information systems engineering, CAISE01*. 2001. Switzerland: Springer-Verlag.
20. Executive Office of the President of the United States, *Analytical perspectives, Budget of the United States Government, Fiscal Year 2006*. 2005, U.S. Government Printing Office. p. 173-179.
21. Office of E-Government & Information Technology, *Updated Federal IT Spending for Budget Year 2010*. 2010.
22. Chabrow, E., *State of the union*, in *InformationWeek*. 2005. p. 40 - 47.

23. Mosquera, M., *Health IT contract failure part of VA mismanagement pattern, inspector says*, in *Government Health IT*. 2009.
24. Hedgpeth, D., *Census Back to Pen and Paper*, in *Washington Post*. 2008.
25. Arnott, S., *Government IT problems since 1997: How the overspend adds up to £1.5bn*, in *Computing*. 2003.
26. Young, T., *NHS given six months to get IT right*, in *Computing*. 2009.
27. BBC News, *IT failure 'causes £130m arrears'*, in *BBC News*. 2008.
28. The Australian Today, *Myki delayed in $216m hit*, in *The Australian Today*. 2008.
29. The Japan Times, *Special panel to investigate pension fiasco*, in *The Japan Times*. 2007.
30. Pearce, S., *Government IT Projects, POST Report 200*. 2003, Parliamentary Office of Science and Technology, The United Kingdom Parliament.
31. Boehm, B. and R. Turner, *Balancing agility and discipline: a guide for the perplexed*. 2003: Addison Wesley.
32. Lee, M.-G., P. Yu, and T. Jefferson, *Eclectic software process methodology and successful software development*, in *Software Engineering 2005*. 2005, International association of science and technology for development (IASTED): Innsbruck, Austria.
33. Oliveira Basto da Silva, J.G.A. and P. Rupino da Cunha. *Reconciling the Irreconcilable? A Software Development Approach that Combines Agile with Formal*. in *The 39th Annual Hawaii International Conference on System Sciences*. 2006.
34. Cohen, S.J. and W.H. Money. *Bridge Methods: Complementary Steps Integrating Agile Development Tools & Methods with Formal Process Methodologies*. in *the 41st Hawaii International Conference on System Sciences - 2008*. 2008. Waikoloa, Big Island, Hawaii.
35. Alleman, G.B., M. Henderson, and R. Seggelke. *Making agile development work in a government contracting environment: measuring velocity with earned value*. in *Agile Development Conference, 2003. ADC 2003*. 2003: IEEE.
36. Upender, B. *Staying agile in government software projects*. in *Agile 2005*. 2005. Denver, Colorado.
37. Willison, J.S., *Agile software development for an agile force*. CROSSTALK, The Journal of Defense Software Engineering, 2004. **17**(4): p. 16-19.
38. Nerur, S., R. Mahapatra, and G. Mangalaraj, *Challenges of migrating to agile methodologies*. Communications of the ACM, 2005. **48**(5): p. 73-78.
39. Davis, A.M., E.H. Bersoff, and E.R. Comer, *A strategy for comparing alternative software development life cycle models*. IEEE Transactions on Software Engineering, 1988. **14**(10): p. 1453-1461.
40. Sorensen, R., *A comparison of software development methodologies*. CROSSTALK, The Journal of Defense Software Engineering, 1995. **8**(1).
41. Verner, J.M. and N. Cerpa. *The effect of department size on developer attitudes to prototyping*. in *International conference of software engineering*. 1997: ACM.
42. Bren, S., et al. *Information management and federal government agencies: case studies*. in *IEEE International Professional Communication Conference, 2001. IPCC 2001*. 2001. Sante Fe, NM, USA.
43. Ferguson, J.R. and M.E. DeRiso, *Software acquisition: a comparison of DoD and commercial practices*, in *Special Report CMU/SEI-94-SR-9*. 1994, Software Engineering Institute: Pittsburgh, PA.
44. OECD, *The hidden threat to e-government: Avoiding large government IT failures*. 2001, OECD Public Management Policy Brief.

45. Computing Services and Software Association, *Getting IT right for government: A review of public sector IT projects*. 2000, Computing Services and Software Association.
46. Fletcher, P.D., *FirstGov: The portal to the U.S. federal government*, in *New Models of Collaboration. A Guide for Managers*. 2003, Center for Technology in Government, University of Albany, SUNY: Albany, NY.
47. Sall, K. *How the US Federal Government is using XML*. in *XML Conference & Exposition 2003*. 2003. Philadelphia, PA.
48. Clark, E.K.B., et al., *Mission-critical and mission-support software: A preliminary maintenance characterization*. CROSSTALK, The Journal of Defense Software Engineering, 1999. **12**(6): p. 17 - 22.
49. Dick, B., *Action research: action and research. Available at* http://www.scu.edu.au/schools/gcm/ar/arp/aandr.html. 2002.
50. Fowler, M., *The new methodology*. 2003.
51. Cockburn, A., *Selecting a project's methodology*. IEEE Software, 2000. **July/August 2000**: p. 64 - 71.
52. http://www.agilemanifesto.org/. 2001.
53. Highsmith, J. *Agile Project Management—Innovation in Action*. in *Agile 2009*. 2009. Chicago, USA.
54. Royce, W.W. *Managing the development of large software system*. in *IEEE WESCON*. 1970.
55. Vliet, H.V., *Software engineering: principles and practice*. 2000, Chichester, England: John Wiley.
56. Pressman, R.S., *Software engineering: a practitioner's approach*. 2001, Boston, MA: McGraw-Hill.
57. Cantor, M., *Software leadership, a guide to successful software development*. 2002, Boston, MA: Addison-Wesley.
58. Schrage, M., *Never go to a client meeting without a prototype*. IEEE Software, 2004. **21**(2): p. 42 - 45.
59. Martin, J., *Rapid application development*. 1991, New York: Macmillan.
60. Bayer, S. and J. Highsmith, *RADical software development*. American Programmer, 1994. **7**: p. 35-42.
61. Berger, H. and P. Beynon-Davies. *Issues impacting on the project management of a RAD development approach of a large, complex government IT project*. in *8th pacific Asia conference on information systems*. 2004. Shanghai, China.
62. Boehm, B.W., *A spiral model of software development and enhancement*. IEEE Computer, 1988. **21**(5): p. 61-72.
63. Cronin, D., *Early and often: How to avoid the design revision death spiral*. Gain: Journal of Business and Design, 2006(March 2, 2006): p. 2 - 13.
64. Boehm, B.W., et al., *Using the winwin spiral model: a case study*. IEEE Computer, 1998. **31**(7): p. 33-44.
65. Boehm, B.W. and R. Ross, *Theory-W software project management: principles and examples*. IEEE Transactions on Software Engineering, 1989. **15**(7): p. 902-916.
66. Kruchten, P., *The rational unified process, an introduction*. 1998, Reading, MA: Addison-Wesley.
67. Williams, L. and A. Cockburn, *Agile software development: it's about feedback and change*. IEEE Computer, 2003: p. 39-43.
68. http://www.agilemanifesto.org/principles.html. 2001.
69. The C3 Team, *Chrysler goes to "Extremes"*, in *Distributed Computing*. 1998. p. 24-28.

70. Beck, K., *Embracing change with extreme programming.* IEEE Computer, 1999. **32**(10): p. 70-77.
71. Jeffries, R., *What is extreme programming?* http://www.xprogramming.com/xpmag/whatisxp.htm. 2001.
72. Auer, K. and R. Miller, *Extreme programming applied: playing to win*. 2002, Boston [Mass.]; London: Addison-Wesley.
73. Cockburn, A., *Learning from agile software development - part two.* CROSSTALK, The Journal of Defense Software Engineering, 2002b. **15**(11).
74. Highsmith, J., *Agile software development ecosystems*. 2002, Boston: Addison-Wesley.
75. Highsmith, J., *Messy, exciting, and anxiety-ridden: adaptive software development.* American Programmer, 1997. **X**(1).
76. Rising, L. and N.S. Janoff, *The Scrum software development process for small teams.* IEEE Software, 2000. **17**(4): p. 26 - 32.
77. ControlChaos.com, http://www.controlchaos.com/about/.
78. Palmer, S.R. and J.M. Felsing, *A practical guide to feature-driven development, Chapter 4 Feature-driven development--processes*. The Coad Series. 2002: Prentice Hall.
79. DSDM Consortium and J. Stapleton, *DSDM: business focused development*. 2003, London, Boston: Addison-Wesley. 239.
80. DSDM Consortium, http://na.dsdm.org/en/about/lifecycle.asp.
81. Elssamadisy, A. and G. Schalliol, *Recognizing and responding to "bad smells" in extreme programming*, in *ICSE 02*. 2002, ACM: Orlando, Florida, USA.
82. Cao, L., et al. *How extreme does extreme programming have to be? Adapting XP practices to large-scale projects*. in *37th annual Hawaii international conference on system sciences*. 2004.
83. Drobka, J., D. Noftz, and R. Raghu, *Piloting XP on four mission-critical projects.* IEEE Software, 2004. **21**(6): p. 70-75.
84. Elssamadisy, A. *XP on a large project - A developer's view: Extended abstract*. in *XP Universe*. 2001. Raleign, NC.
85. Salo, O. *Improving software process in agile software development projects: results from two XP case studies*. in *EUROMICRO*. 2004.
86. Kahkonen, T. *Agile methods for large organizations - building communities of practice*. in *Agile Development Conference*. 2004.
87. Lippert, M., et al., *Developing complex projects using XP with extensions.* IEEE Computer, 2003. **36**(6): p. 67 - 73.
88. Jacobi, C. and B. Rumpe, *Hierarchical XP: Improving XP for large scale projects in analogy to reorganization processes*, in *Extreme programming examined*, G. Succi and M. Marchesi, Editors. 2001, Addison-Wesley: Boston. p. 83 - 102.
89. Leffingwell, D., *Scaling Software Agility: Best Practices for Large Enterprises*. 2007: Addison-Wesley Professional.
90. Schwaber, K., *The enterprise and scrum*. 2007, Redmond, WA: Microsoft Press.
91. Larman, C. and B. Vodde, *Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum*. 2008: Addison-Wesley Professional.
92. Maples, C. *Enterprise Agile Transformation: The Two Year Wall*. in *Agile 2009*. 2009. Chicago, USA.
93. Wilby, D. *Roadmap Transformation: From Obstacle to Catalyst*. in *Agile 2009*. 2009. Chicago, USA.
94. Williams, S., L. Knudsen, and I. Gat. *Agile in the Enterprise Corporation*. in *Agile 2009*. 2009. Chicago, USA.

95. Sutherland, J. and S. Downey. *Shock Therapy: How to Bootstrap a Hyperproductive Team*. in *Agile 2009*. 2009. Chicago, USA.
96. Guckenheimer, S. *Introducing Agile in the Very Large: Microsoft Developer Division's Journey*. in *Agile 2009*. 2009. Chicago, USA.
97. Miller, A. *Distributed Agile Development: Experiments at Microsoft patterns & practices*. in *Agile 2009*. 2009. Chicago, USA.
98. Atlas, A. *Accidental Adoption - The Story of Scrum at Amazon.com*. in *Agile 2009*. 2009. Chicago, USA.
99. Fewell, J. *Marriott's Agile Turnaround*. in *Agile 2009*. 2009. Chicago, USA.
100. Moore, E. *Influence of Large-Scale Organization Structures on Leadership Behaviors*. in *Agile 2009*. 2009. Chicago, USA.
101. McKinney, S. *Leading Agile in an Economic Downturn - "The IBM Transformation Story"*. in *Agile 2009*. 2009. Chicago, USA.
102. Chung, M.-W. and B. Drummond. *Agile @ Yahoo! from the Trenches*. in *Agile 2009*. 2009. Chicago, USA.
103. Striebeck, M. *How to run 4.5 Million tests per day. and why!* in *Agile 2009*. 2009. Chicago, USA.
104. Paulk, M.C., *Extreme programming from a CMM perspective.* IEEE Software, 2001. **18**(6): p. 19 - 26.
105. Veryha, Y., *Integration of security guidelines and assessments into outsourced software development*, in *Software Engineering 2005*. 2005, The international association of science and technology for development (IASTED): Innsbruck, Austria.
106. McMahon, P.E., *Bridging agile and traditional development methods: A project management perspective.* CROSSTALK, The Journal of Defense Software Engineering, 2004. **17**(5): p. 16-20.
107. Turk, D., R. France, and B. Rumpe, *Assumptions underlying agile software-development processes.* Journal of Database Management, 2005. **16**(4): p. 62-87.
108. Gutierrez, L., *Min-Gu Lee introduces cutting-edge software technology at international conference*, in *The I&TSD Reporter*. 2005. p. 20.
109. Reingold, J., *How to read a business book*, in *Fast Company*. 2004. p. 106.
110. Wallin, C. and I. Crnkovic. *Three aspects of successful software development projects "when are projects cancelled, and why?"* in *EUROMICRO*. 2003.
111. Hawley, R., *Were you born to lead?* Engineering Management Journal, 2001. **11**(6): p. 247 - 248.
112. Willerton, D., *After the curtain was pulled away.* IEEE Software, 1999. **16**(5): p. 114-117.
113. Mandl-Striegnitz, P. *How to successfully use software project simulation for educating*. in *31st annual frontiers in education conference, 2001*. 2001. Reno, NV USA.
114. Computer Sciences Corporation, *New IS Leaders (Foundation Report 109)*. 1996.
115. Hancock, J. *Application frameworks before system frameworks*. in *Object-oriented programming, systems, languages, and applications*. 2000. Minneapolis, Minnesota, United States.
116. Ahamed, S.I., A. Pezewski, and A. Pezewski. *Towards framework selection criteria and suitability for an application framework*. in *International conference on information technology: coding and computing, 2004*. 2004: IEEE.
117. Schneider, K. and J.-P.v. Hunnius. *Effective experience repositories for software engineering*. in *25th International conference on software engineering*. 2003: IEEE.
118. Henninger, S. *Tools supporting the creation and evolution of software development knowledge*. in *12th IEEE International conference on automated software engineering*. 1997: IEEE.

119. Louridas, P., *Using Wikis in software development.* IEEE Software, 2006. **March/April 2006**: p. 88-91.
120. Yourdon, E. and L.L. Constantine, *Structured design. Fundamentals of a discipline of computer program and systems design.* 1979, Englewood Cliffs, NJ: Yourdon Press.
121. Booch, G., *Object-oriented analysis and design with applications.* Second edition ed. 1994, Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc.
122. Herbsleb, J., et al., *Software quality and the Capability Maturity Model.* Communications of the ACM, 1997. **40**(6): p. 30 - 40.
123. Fuggetta, A. *Software process: a roadmap.* in *The Future of Software Engineering.* 2000. Limerick, Ireland: ACM.
124. Borjesson, A. and L. Mathiassen, *Successful process implementation.* IEEE Software, 2004. **21**(4): p. 36 - 44.
125. Pfeffer, J. and J.F. Veiga, *Putting people first for organizational success.* The Academy of Management Executive, 1999. **13**(2).
126. DeLong, T.J. and V. Vijayaraghavan, *Let's hear it for B players.* Harvard Business Review, 2003. **June 2003**.
127. Goleman, D., *Leadership that gets results.* Harvard Business Review, 2000. **March-April 2000**.
128. Kotter, J.P., *What leaders really do.* Harvard Business Review, 2001. **December 2001**.
129. Collins, J.C. and J.I. Porras, *Building your company's vision.* Harvard Business Review, 1996. **September - October 1996**.
130. Haeckel, S.H., *Adaptive enterprise: creating and leading sense-and-respond organizations.* 1999, Boston: Harvard Business School Press.
131. Kapoor, S., et al., *A technical framework for sense-and-respond business management.* IBM Systems Journal, 2005. **44**(1): p. 5-24.
132. Boehm, B., *Anchoring the software process.* IEEE Software, 1996. **13**(4): p. 73 - 82.
133. Følstad, A., H.D. Jørgensen, and J. Krogstie. *User involvement in e-government development projects.* in *the third Nordic conference on Human-computer interaction.* 2004. Tampere, Finland: ACM Press New York, NY, USA.
134. Garavalia, L.S., P.A. Marken, and R.W. Sommi, *Selecting appropriate assessment methods: Asking the right questions.* American Journal of Pharmaceutical Education, 2003. **66**(Summer 2003): p. 108 - 112.
135. Parnas, D.L. and D.M. Weiss. *Active design reviews: principles and practices.* in *8th international conference on software engineering.* 1985: IEEE Computer Society Press.
136. Basili, V.R., G. Caldiera, and H.D. Rombach, *The goal question metric approach*, in *Encyclopedia of software engineering, two volume set*, G. Caldiera and D.H. Rombach, Editors. 1994, John Wiley and Sons, Inc: New York City.
137. Maiden, N.A.M., C. Ncube, and A. Moore, *Lessons learned during requirements acquisition for COTS systems.* Communications of the ACM, 1997. **40**(12): p. 21-25.
138. Fairley, R.E. and M.J. Willshire, *Why the Vasa sank: 10 problems and some antidotes for software projects.* IEEE Software, 2003. **20**(2): p. 18 - 25.
139. Peters, T. and N. Austin, *A passion for excellence: The leadership difference.* 1985, New York: Random house.
140. Amis, R., *Time again for management by walking around?* http://management.itmanagersjournal.com/articl.pl?sid=06/01/12/1856259, in *IT Manager's Journal.* 2006.

141. Paré, G. and L. Dubé. *Virtual teams: an exploratory study of key challenges and strategies.* in *International conference on information systems.* 1999.
142. Glazer, H., et al., *CMMI or Agile: Why Not Embrace Both!* in *Agile 2009*. 2008, Software Engineering Institute, Carnegie Mellon University, Technical Note CMU/SEI-2008-TN-003: Chicago, USA.
143. Kock, N., *Communication-focused business process redesign: assessing a communication flow optimization model through an action research study at a defense contractor.* IEEE Transactions on Professional Communication, 2003a. **46**(1): p. 35- 54.
144. Kock, N., *Action research: lessons learned from a multi-iteration study of computer-mediated communication in groups.* IEEE Transactions on Professional Communication, 2003b. **46**(2): p. 105 - 128.
145. Nandhakumar, J. and D.E. Avison, *The fiction of methodological development: a field study of information systems development.* Information Technology & People, 1999. **12**(9): p. 176 - 191.
146. Curtis, B., H. Krasner, and N. Iscoe, *A field study of the software design process for large systems.* Communications of the ACM, 1988. **31**(11): p. 1268 - 1287.
147. Dick, B., *You want to do an action research thesis? Available at http://www.scu.edu.au/schools/gcm/ar/art/arthesis.html*. 1993.
148. Avison, D., R. Baskerville, and M. Myers, *Controlling action research projects.* Information Technology & People, 2001. **14**(1): p. 28 - 45.
149. Baskerville, R.L. and A.T. Wood-Harper, *A critical perspective on action research as a method for information systems research.* Journal of Information Technology, 1996. **11**: p. 235-246.
150. Pardo, T.A. and H.J. Scholl. *Walking atop the cliffs: avoiding failure and reducing risk in large-scale e-government projects.* in *35th annual Hawaii international conference on system sciences.* 2002: IEEE.
151. Briggs, R.O., et al. *Lessons learned using a technology transition model with the US Navy.* in *32nd annual Hawaii international conference on system sciences*. 1999. Maui, HI, USA: IEEE.
152. Fruhling, A. and G.-J.D. Vreede, *Field experiences with eXtreme programming: Developing an emergency response system.* Journal of Management Information Systems, 2006. **22**(4): p. 39-68.
153. Börjesson, A., F. Martinsson, and M. Timmerås, *Agile improvement practices in software organizations.* European Journal of Information Systems, 2006. **15**(2): p. 169 - 182.
154. Dyba, T. and T. Dingsoyr, *What Do We Know about Agile Software Development?* Software, 2009. **26**(5): p. 6-9.
155. Moe, N.B., T. Dingsøyr, and E.A. Røyrvik. *Putting Agile Teamwork to the Test – An Preliminary Instrument for Empirically Assessing and Improving Agile Software Development.* in *10th International Conference, XP 2009*. 2009. Pula, Sardinia, Italy.
156. Fægri, T.E. *Improving General Knowledge in Agile Software Organizations: Experiences with Job Rotation in Customer Support.* in *Agile 2009*. 2009. Chicago, USA.
157. Salo, O. and P. Abrahamsson. *Empirical evaluation of agile software development: The controlled case study approach.* in *International conference on product focused software process improvement.* 2004. Kansai Science City, Japan.
158. Baskerville, R.L. and J. Stage, *Controlling prototype development through risk analysis.* MIS Quarterly, 1996. **20**(4): p. 481 - 504.
159. Abrahamsson, P. and N. Iivari. *Commitment in software process improvement - In search of the process.* in *35th annual Hawaii international conference on system sciences.* 2002: IEEE.

160. Susman, G.I. and R.D. Evered, *An assessment of the scientific merits of action research.* Administrative Science Quarterly, 1978. **23**(4): p. 582-603.
161. Baskerville, R.L., *Investigating information systems with action research.* Communications of the AIS, 1999. **2**(9): p. 1-31.
162. Lau, F., *Toward a framework for action research in information systems studies.* Information Technology & People, 1999. **12**(2): p. 148 - 175.
163. Teasdale, I.A., *Building learning systems: A study of the design and implementation of two corporate learning systems*, in *Department of Instructional Systems Technology*. 2005, Indiana University: Bloomington, Indiana. p. 235.
164. Truex, D.P., R. Baskerville, and H. Klein, *Growing systems in emergent organizations.* Communications of the ACM, 1999. **42**(8): p. 117 - 123.
165. Truex, D., R. Baskerville, and J. Travis, *Amethodical systems development: the deferred meaning of systems development methods.* Accounting, Management and Information Technologies, 2000. **10**: p. 53 - 79.
166. Sommer, R. and B. Sommer, *A practical guide to behavioral research*. Fifth Edition ed. 2002, New Tork; Oxford: Oxford University Press.
167. Rumpe, B. and A. Schroder. *Quantitative survey on extreme programming projects*. in *the third international conference on extreme programming and flexible processes in software engineering, XP2002*. 2002. Alghero, Italy.
168. McKay, J. and P. Marshall, *The dual imperatives of action research.* Information Technology & People, 2001. **14**(1): p. 46-59.
169. Winter, R., *Some principles and procedures for the conduct of action research*, in *New Directions in Action Research*, O. Zuber-Skerritt, Editor. 1996, Falmer Press: London.
170. O'Brien, R., *An overview of the methodological approach of action research*, in *Theory and Practice of Action Research. (English version) Available: http://www.web.ca/~robrien/papers/arfinal.html*, J. Pessoa, Editor. 2001, Universidade Federal da Paraíba: Brazil.
171. Hsieh, H.-F., *Three Approaches to Qualitative Content Analysis.* Qualitative Health Research, 2005. **15**(9): p. 1277 - 1288.
172. *The SAGE Encyclopedia of Social Science Research Methods*, in *The SAGE Encyclopedia of Social Science Research Methods*. 2010.
173. Berelson, B., *Content analysis in communication research*. 1952, Free Press: New York. p. 13-20, 147, 165-8.
174. Krippendorff, K., *Content analysis: an introduction to its methodology*. 2004, Sage Publication.
175. Reinard, J.C., *Introduction to Communication Research*. 4 ed. 2007: McGraw-Hill. 768.

Appendix A Interview Notes about the Participants' and Organization's Experiences and Expectations

Information about the Research Study

"The Validation and Refinement of a Formal Approach to Eclectic Software Development"

You are invited to participate in a research study under the direction of Dr. Theresa Jefferson of the Department of Engineering Management and Systems Engineering, George Washington University (GWU). Taking part in this research is entirely voluntary. You can choose not to participate or decide to withdraw from the study at any time.

The purpose of this study is to examine results of applying eclectic software development approaches to balance agile methodology and traditional plan-driven environment in the government software development community.

The research will be conducted at the following location(s): Lockheed Martin Corporation

A total of 30 participants at approximately 2 projects will be asked to take part in this study. You will be one of approximately 15 participants to be asked to take part at this location.

If you choose to take part in this study, you will participate at two interviews and one evaluation form called ESD form.

The first interview at the beginning of the study is about the participants' professional experiences and his or her expectations of this study. The second interview at the end of the study is about the participant's experiences and result of the proposed method for the observed study. You will also fill out a form to evaluate the proposed method at the end of study. The total amount of time you will spend in connection with this study is 2 hours (30 minutes for each interview and 1 hour for the ESD form)

There are no physical risks associated with this study. There is, however, the possible risk of loss of confidentiality. Every effort will be made to keep your information confidential, however, this can not be guaranteed. Some of the questions we will ask you as part of this study may make you feel uncomfortable. You may refuse to answer any of the questions and you may take a break at any time during the study. You may stop your participation in this study at any time.

You will not benefit directly from your participation in the study. The benefits to science and humankind that might result from this study are: The Eclectic Software Development (ESD) approach provides a framework that capitalizes on the benefits of agile and traditional methodologies for the government sector while minimizing the limitations of both.

You will not be paid for taking part in this study.

The investigator can decide to withdraw you from the study at any time. You could be taken off the study for reasons related solely to you (for example, not following study-related directions from the Investigator) or because the entire study is stopped.

If results of this research study are reported in journals or at scientific meetings, the people who participated in this study will not be named or identified. GW will not release any information about your research involvement without your written permission, unless required by law.

The Office of Human Research of George Washington University, at telephone number (202) 994-2715, can provide further information about your rights as a research participant. If you think you have been harmed in this study, please report this to the Principal Investigator of this study or call the Office of Human Research immediately. Further information regarding this study may be obtained by contacting Min-Gu Lee, a principal contact for this study, at telephone number (301) 675-9869.

To ensure anonymity, your signature is not required in this document unless you prefer to sign it. Your willingness to participate in this research study is implied if you proceed with completing the survey/interview.

*Please keep a copy of this document in case you want to read it again.

Appendix B. Interview Notes about the Participants' and Organization's Experiences and Expectations

Hello, I am Min-Gu Lee, a doctoral student at George Washington University. My research focuses on situational method engineering for government IT projects. The study is sponsored by Department of Engineering at George Washington University. As a part of the research, I will conduct two interviews, one today and one at the end of the project. Our interview today is about the participants' professional experiences and his or her expectations of this study. This will take about 30 minutes. If there are any questions you do not want to answer, please let me know, and we will move on to the next one. If I have your permission, we can start the questions.

Date: _____

Project: _____

The next few questions will be about your position.

1. Have you been involved in the selection, tailoring, or development of software development lifecycle method? ***Yes/No*** Will you be involved in the selection, tailoring, or development of software development lifecycle method? ***Yes/No***

2. What is your total of years of IT experience? _____***Year(s)*** How long have you worked for the government industry? _____***Year(s)***

The next few questions will be about your experiences with software development methodology.

3. Have you participated in any software development projects using the Waterfall method? ***Yes/No*** If Yes, was it a government project? ***Yes/No*** Will you use it for other government projects? ***Yes/Neutral/No*** Will you actively advocate it in the future? ***Yes/Neutral/No***

4. Have you participated in any software development projects using the Prototyping method? ***Yes/No*** If Yes, was it a government project? ***Yes/No*** Will you use it for other government projects? ***Yes/Neutral/No*** Will you actively advocate it in the future? ***Yes/Neutral/No***

5. Have you participated in any software development projects using the Rapid Application Development method? ***Yes/No*** If Yes, was it a government project? ***Yes/No*** Will you use it for other government projects? ***Yes/Neutral/No*** Will you actively advocate it in the future? ***Yes/Neutral/No***

6. Have you participated in any software development projects using the Spiral/ Incremental/Evolutional method? ***Yes/No*** If Yes, was it a government project? ***Yes/No*** Will you use it for other government projects? ***Yes/Neutral/No*** Will you actively advocate it in the future? ***Yes/Neutral/No***

7. Have you participated in any software development projects using the Rational Unified Process (RUP)? ***Yes/No*** If Yes, was it a government project? ***Yes/No*** Will you use it for other government projects? ***Yes/Neutral/No*** Will you actively advocate it in the future? ***Yes/Neutral/No***

8. Have you participated in any software development projects using any agile methods? For example, Extreme Programming (XP), Crystal, Adaptive Software Development (ASD), SCRUM, Feature Driven Development (FDD), Dynamic Systems Development Method (DSDM). ***Yes/No*** If Yes, which agile method was that? _____ If Yes, was it a government project? ***Yes/No*** Will you use it for other government projects? ***Yes/Neutral/No*** Will you actively advocate it in the future? ***Yes/Neutral/No***

9. Have you applied process fragments from various methods into one project? ***Yes/No*** If Yes, was it a government project? ***Yes/No*** Will you use it for other government projects? ***Yes/Neutral/No*** Will you actively advocate it in the future? ***Yes/Neutral/No***

The next question will be about the project.

10. What do you think about applying the agile method in a government project? Do you have any reservations about it?

_____
_____
_____
_____
_____
_____
_____

Thank you very much for your time and effort. Is there anything you would like to add to the interview?

_____
_____
_____
_____
_____
_____
_____

Thanks again. You have my card. If you have any questions about the interview or the results, you can contact me at 301-675-9869 or mingulee@gwu.edu.

Note

- It is necessary to probe for clarification when a researcher receives an unclear or incomplete answer.
- This interview note was developed utilizing content and structures from previously published research (Rumpe & Schroder, 2002; Sommer & Sommer, 2002).

Appendix B Agile Program Assessment Form and Best Practice Checklist

Date: / /2009

Project: _____

This form will help you to evaluate agile best practices for your government software development project. The following diagram provides a list of best practices from popular agile methods in addition to commercial agile best practices, scaling agile best practices, and government agile best practices. This form can be customized to meet the specific requirements of your project.

You can document Challenges, Approaches, and Results. The first column categorizes five project factors (People, Methodologies, Tools, Process, and Leadership), and each project factor will be assessed by customizable sub-factors in the second column.

In the third column, please check one (Initial (1), Managed (2), Defined (3), Quantitatively Managed (4), or Optimizing (5)) to rate these sub-factors, indicating how you feel for each sub-factors in your project.

In the fourth, fifth, and sixth column, please describe challenges your team face, action your team will take, and the result. Your input will be continually updated and revised. You can enter multiple challenges, actions, and results. Also, each result includes an indicator: positive (+), neutral (=), or negative (-) impact.

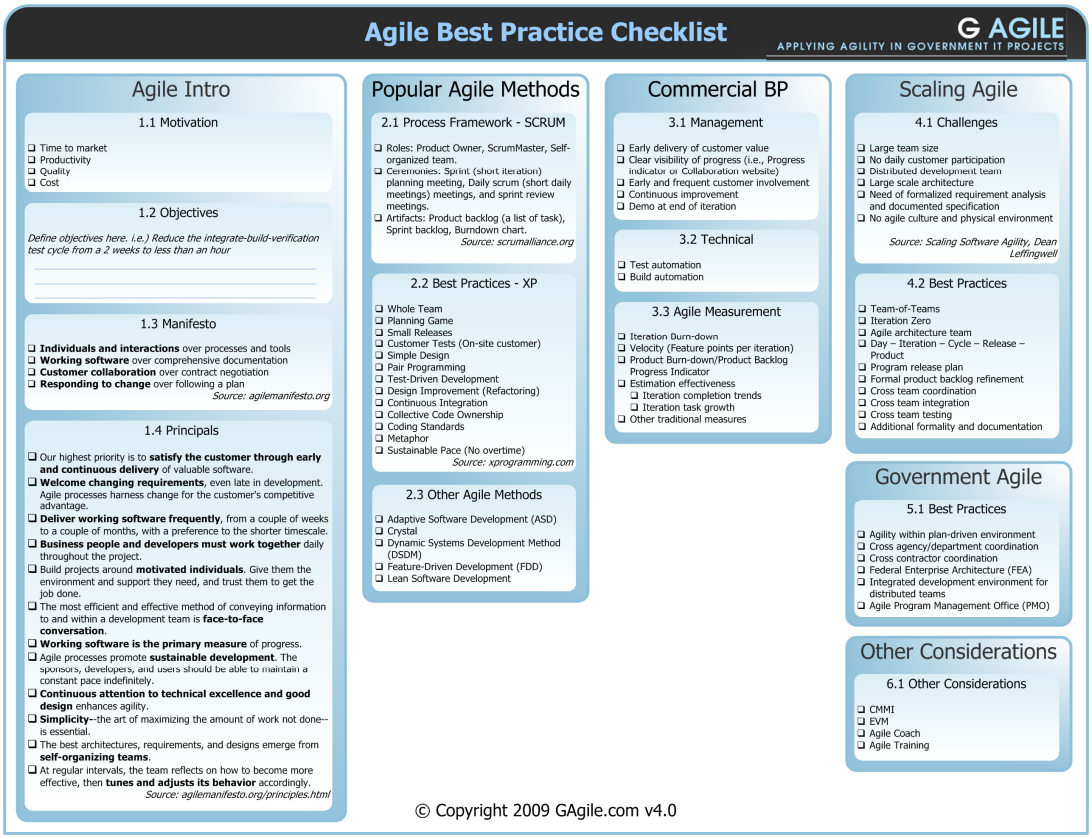


Figure 66 Agile Best Practice Checklist Template

Table 22 Agile Assessment Form

Project Factors	Sub-Factors	Rate	Challenge / Comments	Approach
People	**Structure**	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		

Project Factors	Sub-Factors	Rate	Challenge / Comments	Approach
	Skill Set	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Training	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Acceptance	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
Methodologies	**Requirement**	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Analysis and Design	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		

Project Factors	Sub-Factors	Rate	Challenge / Comments	Approach
	Development	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Test	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
Tools	**Tool**	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Framework	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Architecture	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		

Project Factors	Sub-Factors	Rate	Challenge / Comments	Approach
	Technology	☐Initial (1) ☐Managed (2) ☐Defined (3) ☐Quantitatively Managed (4) ☐Optimizing (5))		
Process	**Model**	☐Initial (1) ☐Managed (2) ☐Defined (3) ☐Quantitatively Managed (4) ☐Optimizing (5)		
	Execution	☐Initial (1) ☐Managed (2) ☐Defined (3) ☐Quantitatively Managed (4) ☐Optimizing (5)		
	Assess	☐Initial (1) ☐Managed (2) ☐Defined (3) ☐Quantitatively Managed (4) ☐Optimizing (5)		
	Improvement	☐Initial (1) ☐Managed (2) ☐Defined (3) ☐Quantitatively Managed (4) ☐Optimizing (5)		

Project Factors	Sub-Factors	Rate	Challenge / Comments	Approach
Leadership	**Visionary**	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Technical	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Functional	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		
	Managerial	☐ Initial (1) ☐ Managed (2) ☐ Defined (3) ☐ Quantitatively Managed (4) ☐ Optimizing (5)		

Appendix C Interview Notes about Participants' Experiences with ESD and Results of the Project

Date: / /2009

Project: .

D1. ESD has been implemented to promote efficient responsiveness to changes (a benefit of the agile method and a limitation of the plan-driven method) and predictability (a benefit of the plan-driven method) for the government project (a limitation of agile). In your project, could these goals be reached?

Responsiveness to Changes	Predictability
☐2= Fully Achieved, ☐1 = Partially Achieved ☐0 = As Always ☐-1 = Partially Worse ☐-2 = Much Worse	☐2= Fully Achieved, ☐1 = Partially Achieved ☐0 = As Always ☐-1 = Partially Worse ☐-2 = Much Worse

Please explain the obstacles or areas of improvement.

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

D2. Will you use ESD again? ☐*Yes/*☐*No/*☐*Neutral* Why/Why not?

_____
_____
_____
_____
_____
_____
_____

D3. Will you actively advocate for ESD in the future? ☐*Yes/*☐*No/*☐*Neutral* Why/Why not?

_____
_____
_____
_____
_____
_____
_____

D4. Further comments

_____
_____
_____
_____
_____
_____
_____

Note

- This interview note was developed utilizing content and structures from previously published research [167].

Appendix D Researcher's Observation Note

E1. Company

Name of company: _____

E2. Project

Name of project: _____

Client sector: _____

Duration of project: _____

Team size: _____

Programming language: _____

Technologies used: _____

Software domain: _____

Development type: _____

Public Sector Project Characteristics	Evaluation
Hard to measure for success because of multiple aims	
Generally not in competition with other projects	
Likely to interact with other departments	
Highly visible to the public and the media	
Constrained by legislation and regulations	
Managed in a risk averse culture	

E3. Observation Note: Responsiveness to Changes

Date	Event / Memo	Criticality

E4. Observation Note: Predictability

Date	Event / Memo	Criticality

E5. Observation Note: Others

Date	Event / Memo	Criticality

E6. Project Control Structure

The following guideline was suggested for controlling action research in the information systems field [148]:

Control Structure	Form	Characteristics	Evaluation
Initiation	Researcher	Field experiment	
	Practitioner	Classic action research genesis	
	Collaborative	Evolved from existing interaction	
Authority	Practitioner	Constructed action warrant	
	Staged	Migration of power	
	Identify	Practitioner and researcher are the same person	
Formalization	Formal	Specific written contact or letter of agreement	
	Informal	Broad, perhaps verbal, agreement	
	Evolved	Informal or formal project shift into the opposite form	

Note

- This interview note was developed utilizing content and structures from previously published research [167].