


1-1-2011

Data-Driven Decision Making as a Tool to Improve Software Development Productivity

Mary Erin Brown
Walden University

Follow this and additional works at: <https://scholarworks.waldenu.edu/dissertations>

 Part of the [Business Administration, Management, and Operations Commons](#), [Databases and Information Systems Commons](#), [Library and Information Science Commons](#), and the [Management Sciences and Quantitative Methods Commons](#)

This Dissertation is brought to you for free and open access by the Walden Dissertations and Doctoral Studies Collection at ScholarWorks. It has been accepted for inclusion in Walden Dissertations and Doctoral Studies by an authorized administrator of ScholarWorks. For more information, please contact ScholarWorks@waldenu.edu.

Walden University

College of Management and Technology

This is to certify that the doctoral dissertation by

Mary Brown

has been found to be complete and satisfactory in all respects,
and that any and all revisions required by
the review committee have been made.

Review Committee

Dr. David Gould, Committee Chairperson, Management Faculty

Dr. Stuart Gold, Committee Member, Management Faculty

Dr. Louis Taylor, University Reviewer, Management Faculty

Chief Academic Officer

Eric Riedel, Ph.D.

Walden University

2013

Abstract

Data-Driven Decision Making as a Tool to Improve Software Development Productivity

by

Mary Erin Brown

M.S. Arizona State University, 1998

M.A. Western Michigan University, 1976

B.A. Western Michigan University, 1970

Dissertation Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

Management

Walden University

August 2013

Abstract

The worldwide software project failure rate, based on a survey of information technology software manager's view of user satisfaction, product quality, and staff productivity, is estimated to be between 24% and 36% and software project success has not kept pace with the advances in hardware. The problem addressed by this study was the limited information about software managers' experiences with data-driven decision making (DDD) in agile software organizations as a tool to improve software development productivity. The purpose of this phenomenological study was to explore how agile software managers view DDD as a tool to improve software development productivity and to understand how agile software development organizations may use DDD now and in the future to improve software development productivity. Research questions asked about software managers', project managers', and agile coaches' lived experiences with DDD via a set of interview questions. The conceptual framework for the research was based on the 3 critical dimensions of software organization productivity improvement: people, process, and tools, which were defined by the Software Engineering Institute's Capability Maturity Model Integrated published in 2010. Organizations focus on processes to align the people, procedures and methods, and tools and equipment to improve productivity. Positive social change could result from a better understanding of DDD in an agile software development environment; this increased understanding of DDD could enable organizations to create more products, offer more jobs, and better compete in a global economy.

Data-Driven Decision Making as a Tool to Improve Software Development Productivity

by

Mary Erin Brown

M.S. Arizona State University, 1998

M.A. Western Michigan University, 1976

B.A. Western Michigan University, 1970

Dissertation Submitted in Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

Management

Walden University

August 2013

UMI Number: 3591716

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3591716

Published by ProQuest LLC (2013). Copyright in the Dissertation held by the Author.

Microform Edition © ProQuest LLC.

All rights reserved. This work is protected against unauthorized copying under Title 17, United States Code



ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

Dedication

This proposal is dedicated to my husband, Mike Brown, who supported my goals, to my dad, Mr. Phillip Stackpoole who taught me to love learning, to Dr. Paul Doelle who taught me the importance of moving in the direction you want to go whether the wind is at your back or in your face, and to my sisters and brothers who made all the difference in my life.

Acknowledgments

I would like to thank the dedicated faculty and staff of Walden University School of Management who shared their knowledge and expertise with me and who provided encouragement along the way, especially Dr. David Gould, Dr. Stuart Gold, and Dr. Louis Taylor who provided valuable guidance along the way.

Table of Contents

List of Tables	v
List of Figures	vii
Chapter 1: Introduction to the Study.....	1
Background of the Study	2
Problem Statement	4
Purpose of the Study	5
Research Questions	6
Conceptual Framework.....	8
Nature of the Study	11
Definition of Terms.....	12
Assumptions.....	16
Scope and Delimitations	17
Limitations	18
Significance of the Study	18
Summary and Transition.....	19
Chapter 2: Literature Review	21
The Literature Search Strategy	22
Organization of the Review	24
Conceptual Foundation	24
Current Understanding of Data Driven Decision Making	25
Current Research in Software Methods	34

Current Research in Software Development and KM	51
Current Research in Software Methods and Analytics	61
Current Research in Software Methods, KM, and Analytics.....	63
Research Methods in the Current Literature	64
Research Methods for Research.....	78
Research Approaches in the Current Literature	79
Research Approach for Used for this Research	80
Research Process Used for this Research	81
Summary and Conclusions	81
Chapter 3: Research Method.....	84
Research Design and Rationale	85
Research Questions	85
Central Concept	86
Research Tradition	87
Rationale	88
Role of the Researcher	90
Researcher Role	90
Relationships.....	90
Management of Bias / Relationships	91
Other Ethical Issues	91
Methodology	92
Participant Selection Logic	92

Instrumentation	96
Procedures for Pilot Studies.....	97
Procedures for Recruitment, Participation, and Data Collection.....	98
Data Analysis Plan.....	100
Issues of Trustworthiness.....	102
Trustworthiness.....	103
Ethical Considerations	105
Dissemination of Findings	108
Summary and Transition.....	108
Chapter 4: Results.....	110
Pilot Study.....	111
Research Setting.....	113
Demographics	114
Data Collection	117
Data Analysis	118
Evidence of Trustworthiness.....	127
Research Results	128
Research Question 1	129
Research Question 2	131
Research Question 3	153
Comparison of Themes by Research Participant Demographics.....	170
Research Question 4	175

Summary	178
Chapter 5: Discussion, Conclusions, and Recommendations	180
Interpretation of the Findings.....	181
Limitations of the Study.....	190
Recommendations.....	191
Implications.....	192
Conclusion	193
References.....	195
Appendix A: Qualitative Research Schedule.....	211
Appendix B: Agile Scrum Process Versus the Traditional Waterfall Process	218
Appendix C: Informed Consent Form	220
Appendix D: Research E-mail Invitation.....	224
Appendix E: Curriculum Vitae	225

List of Tables

Table 1. Pros and Cons of Agile Software Development Methodologies	46
Table 2. Research Participant Roles and Years of Agile Software Development Experience	115
Table 3. Research Participant Project Size	116
Table 4. Research Participant Agile Software Development Methods Used	117
Table 5. Codes That Emerged From The Data	120
Table 6. Categories That Emerged From The Data Analysis Process	121
Table 7. Km Themes That Emerged From The Data By Category	125
Table 8. Analytic Themes That Emerged From The Data By Category	126
Table 9. Current Use Of Analytics By Analytic Type.....	134
Table 10. Current Use Of Analytics - Category By Agile Coaches	135
Table 11. Current Use Of Analytics - Category By Project Managers.....	136
Table 12. Current Use Of Analytics - Category By Software Managers.....	136
Table 13. Current Use Of Analytics – Swabok Activity By Analytic Type.....	142
Table 14. Current Use Of Knowledge Management By Km Process.....	144
Table 15. Current Use Of Km - Category By Agile Coaches.....	145
Table 16. Current Use Of Km - Category By Project Managers	145
Table 17. Current Use Of Km - Category By Software Managers.....	146
Table 18. Current Use Of Km – Swabok Activity By Km Activity	153
Table 19. Future Use Of Analytics All	155
Table 20. Future Use Of Analytics - Category By Agile Coaches	156

Table 21. Future Use Of Analytics - Category By Project Managers	156
Table 22. Future Use Of Analytics - Category By Software Managers	157
Table 23. Future Use Of Analytics – Swebok Activity By Theme	162
Table 24. Future Use Of Knowledge Management	163
Table 25. Future Use Of Km - Category By Agile Coaches	165
Table 26. Future Use Of Km - Category By Project Managers.....	165
Table 27. Future Use Of Km - Category By Software Managers.....	165
Table 28. Future Use Of Km – Swebok Activity By Km Activity.....	170
Table 29. Analytic Themes Unique To Research Participants With <5 Years Of Experience	171
Table 30. Analytic Themes Discussed By Research Participants – Unique For Project Size	173
Table 31. Km Themes Discussed By Research Participants – Unique For Project Size	174
Table 32. Analytic Themes Discussed By Research Participants Using Xp	175

List of Figures

Figure 1. The 3 critical dimensions of software organization productivity improvement...9	9
Figure 2. Literature Review: Number of Articles by Topic.....21	21

Chapter 1: Introduction to the Study

Although software project failure rates have decreased over the past 10 years, Ambler (2012), Emam and Koru (2008), Mieritz (2012), and the Standish Group (n.d.) found that the software project success rate still needs to be improved and Fitzgerald (2012) stated that there is a crisis in software development because software development productivity has not kept pace with the advancements in hardware. The social implications for improved software development productivity included the opportunity for organizations to compete more effectively in a global economy. If software development productivity improved, more software products may be developed, which would potentially decrease the cost of software products and increase the number of individuals who could experience the benefits.

Brynjolfsson, Hitt, and Kim (2011) found that data driven decision-making (DDD) improved organizational output and productivity by 5-6%. If DDD can improve organizational output and productivity, then a better understanding of DDD in software organizations may enable software organizations to improve output and productivity. This study was conducted to better understand the meaning of DDD in software organizations.

A qualitative research study is described in this dissertation. The problem addressed by this research study is discussed in Chapter 1, as is the purpose for the study and the research questions. Research plans should describe the research in as much detail as possible; therefore, the conceptual framework, assumptions, scope, delimitations, and limitations of the research are discussed in Chapter 1 and the implications for social

change are explained. The three dimensions of software organization productivity improvement, which were defined by the Software Engineering Institute (SEI) Capability Maturity Model Integrated (CMMI), provided the conceptual framework for the research study. The operational definitions used to explore DDD in this qualitative research study and to measure DDD in this qualitative research study are provided to minimize ambiguity. The background of this research study is provided before the detailed plan is discussed to explain why this research study was needed.

Background of the Study

There is a need to improve software project success according to Ambler (2012), Emam and Koru (2008), Mieritz (2012), and the Standish Group (n.d.). Agile software development methods were developed to improve software project success (Rao, Naidu, & Chakka, 2011). Agile software methods are based on the Agile Manifesto, which states that software development should focus on delivering working software; consequently, agile methods are intended to provide value to customers by iteratively delivering working code to customers. Although the failure rate for software projects that used traditional software development methods is 50%, the failure rate for software development projects that used agile software development methods is 40% (Ambler, 2012).

Brynjolfsson et al. (2011) found that DDD improved organizational productivity by 5-6%; however, based on a review of the literature, few research studies have explored the use of DDD as a tool to improve software development productivity in either a traditional software development environment or an agile software development

environment. Although Brynjolfsson et al. defined DDD as “data and business analytics” (p. 1), Chandler, Hostmann, Rayner, and Herschel (2011), stated organizations need to define analytics because analytics have been defined many ways.

Brynjolfsson et al. (2011) argued that DDD was related to the knowledge management (KM) processes of “knowledge creation, accumulation, retention, and transfer” (p. 4); however, Brynjolfsson et al. did not state that KM and DDD are equivalents. Although Chan and Thong (2010) found that there was a positive relationship between the agile practices of pair programming, collective ownership, and coding standards with the KM outcomes of knowledge creation, knowledge retention, and knowledge transfer, the meaning of DDD in the context of an agile software environment has not been defined. The research focused on agile software management’s understanding of DDD, which includes agile software management’s understanding of analytics and the KM processes of knowledge creation, accumulation, retention, and transfer to improve software development productivity.

Based on a review of the literature, a few research studies explored the use of KM to improve software development productivity in a traditional software development environment (Slaughter & Kirsch, 2006) or in an agile software development environment (AlaAli & Issa, 2011; Amescua, Bermon, Garcia, & Sanchez-Segura, 2010; Neves Rosa, Correia, & Neto, 2011; Pikkarainen, Haikara, Salo, & Abrahamson, 2008; Tessem & Mauer, 2007). One research study was found that used both analytics and KM to improve productivity in a traditional software development environment. Intelligent agents were used to enhance a knowledge management system (KMS) to manage defects

in a traditional software development environment (Abdullah, Talib, & Misran, 2011b). However, no research studies were found that explored the use of analytics and KM to improve productivity in an agile software development environment.

Qualitative research methods were used to explore management's understanding of DDD as a tool to improve software development productivity. The intent was to better understand the phenomenon of DDD as a tool to improve software development productivity and to explore how software organizations may use DDD now and in the future to improve productivity in an agile software development environment. If software development productivity is improved, organizations may be able to take advantage of the advances in hardware and compete more effectively in a global economy.

Problem Statement

The problem was the limited information about software managers' lived experiences with DDD in agile software organizations as a tool to improve software development productivity. Although the software project failure rate fell from approximately 50% in 1994 to approximately 26-34% in 2007, the software project failure rate remains unacceptably high (Emam & Koru, 2008). Software methods, such as agile methods, were developed to improve software development productivity (Schwaber, 1995); however, software development improvements have not kept pace with advancements in hardware (Fitzgerald, 2012). DDD, which was found to improve organizational productivity by 5-6% based on a survey of the business practices and information technology investments of 179 publicly traded organizations (Brynjolfsson

et al., 2011), may provide software organizations with the tools to improve productivity; however, software managers need to brainstorm ways to use DDD to improve software development productivity because chief technology officers (CTOs) do not know how to communicate the potential use of DDD to their subordinates (Adrian & Genovese, 2011).

Although some research has been done on the use of KM processes and tools to improve software development productivity in traditional software environments (Abdullah et al, 2011b; Slaughter & Kirsch, 2006) and in agile software environments (AlAli & Issa, 2011; Boehm, Lane, Koolmanojwong & Turner, 2010; Ceschi, Sillitti, Succi, & Panfilis, 2005; Rubin & Rubin, 2011) little research has been done on the use of analytical tools to improve software development productivity in traditional software environments (Hullett, Nagappan, Schuh, & Hopson, 2011; Siwen & Jun, 2010; Zare & Akhaven, 2009) or in agile software environments (Abouelela & Benedicenti, 2010). The research study explored how software managers, project managers, and agile coaches used DDD as a tool to improve software development productivity. An in-depth understanding of DDD as a tool to improve software development productivity in an agile software development environment may encourage software managers to create and share new ways to improve software development productivity and may enable future research that measures the effectiveness of alternative DDD uses to improve software development productivity.

Purpose of the Study

Although Maxwell (2005) preferred to use the word *goal* rather than the word *purpose* to describe the intent of the research, the purpose for this research will be

described. The purpose of this phenomenological study was to explore the lived experiences of software managers' use of DDD in agile software organizations as a tool to improve software development productivity. The purpose was to illustrate impediments to DDD use in software development organizations and to make recommendations for improving DDD use in software development organizations based on the findings from this research study and a review of the literature.

At this stage in the research, software development productivity refers to increasing the amount of deliverable software based on the agile principles (Glazer, Dalton, Anderson, Konrad, & Shrum, 2008) rather than to increasing the lines of code produced per hour or to increasing the number of function points produced per hour. Software development productivity also refers to individual productivity, team productivity, and organizational productivity. DDD refers to data, analytics, knowledge creation, knowledge accumulation, knowledge retention, and knowledge transfer. The software development activities defined by the IEEE Software Engineering Body of Knowledge (SWEBOK), published in 2004, provided a software development framework that is agnostic to the software development methods used; consequently, the SWEBOK (2004) activities are applicable in an agile software development environment.

Research Questions

Miles and Huberman (1994) recommended formulating general questions and if necessary formulating more specific questions related to the general questions. The number of research questions should be limited to six or fewer. Four major questions were formulated for this research study. The purpose for this research study was

exploratory; therefore, qualitative research methods, including in depth interviews, were conducted. The interview questions (see Appendix A) were derived from these research questions. Questions were added to obtain demographic data.

The interview questions were intended to gather qualitative data on the phenomenon under study. Interpretive phenomenological analysis (IPA) research methods described by Smith, Flowers, and Larkin (2009) were used to analyze the responses from the interviews on how DDD may be used in agile software organizations based on the experiences of agile software managers.

The following research questions are based on the lived experiences of software managers, project managers, and agile coaches in agile software environments.

1. What do software managers, project managers, and agile coaches in agile software environments think about the use of DDD to improve software development productivity?
2. How do software managers, project managers, and agile coaches in agile software environments currently use descriptive analytics, diagnostic analytics, prescriptive analytics, and predictive analytics, or knowledge creation, retention, accumulation, and transfer to improve software development productivity?
3. How do software managers, project managers, and agile coaches in agile software environments think descriptive analytics, diagnostic analytics, prescriptive analytics, and predictive analytics, or knowledge creation,

retention, accumulation, and transfer could be used to improve software development productivity?

4. What obstacles do software managers, project managers, and agile coaches in agile software environments think their organization needs to overcome to improve software development productivity?

Conceptual Framework

The conceptual framework for the research was based on the three critical dimensions of software organization productivity improvement defined by the Software Engineering Institute's (SEI) Capability Maturity Model Integrated (CMMI) published in 2010, as shown in Figure 1. Organizations focus on processes to align people, procedures and methods, and tools and equipment to improve productivity. According to the CMMI (2010), productivity may be improved if organizations define processes, establish process improvement goals, and measure the outcomes. Organizations need to train people to use procedures and methods that are intended to achieve the process improvement goals and organizations need to provide people with tools and equipment that will enable the people to achieve the desired outcomes to improve productivity.

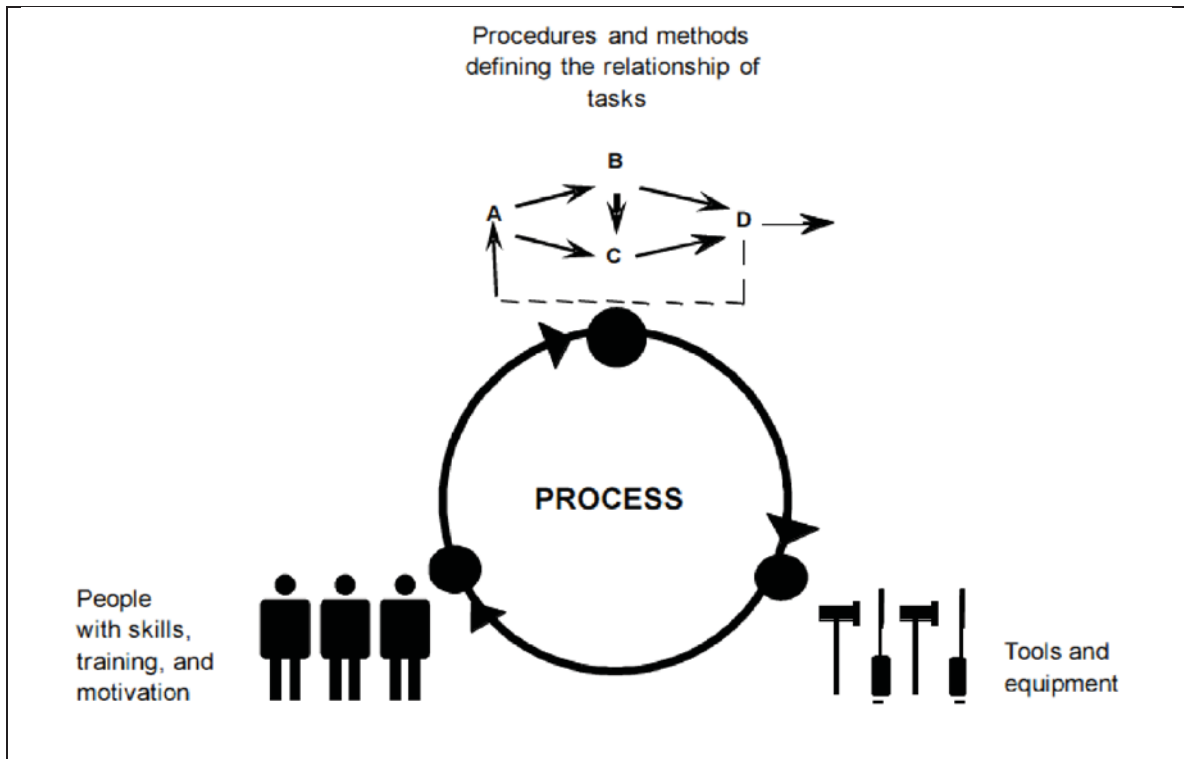


Figure 1. The three critical dimensions. Reprinted from <http://www.sei.cmu.edu>, by the CMMI Product Team, 2010, Retrieved from <http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm>

Agile software methods may improve software development productivity (Balijepally, Mahapatra, Nerur, & Price, 2009; Ballou, 2008; Boehm et al., 2010; Eccles, Smith, TanBelle, & van der Watt, 2010; Glazer et al., 2008; Ionel, 2009; Lalsing, Kishnah, & Pudaruth, 2008; Layman, Williams & Cunningham, 2006; Shull et al., 2010; Sutherland, Jakobsen, & Johnson, 2007; Zhang & Patel, n.d.); consequently, if people were trained to use agile software development methods, productivity may be improved. DDD, which includes the use of data, analytics, and KM tools and techniques to make decisions, was found to improve organizational productivity (Brynjolfsson et al., 2011).

However, as discussed in Chapter 2, little research was found on the use of DDD to improve productivity in either a traditional software development environment or an agile software development environment and no research was found on software management's understanding of DDD as a tool to improve software development productivity. Consequently, this research was intended to fill this gap in the literature by exploring software management's understanding of DDD as a potential tool to improve productivity in an agile software development environment.

A review of the literature revealed the need for additional research into the meaning of DDD and the related topics of business intelligence (BI), artificial intelligence (AI), business analytics, data mining, knowledge management, and entity resolution and analysis (Adrian & Genovese, 2011; Herschel, 2011; Lingling, Jun, Yong, & Xiaohui, 2009). There are many potential uses for BI and business analytic software; however, CTOs and chief information officers (CIOs) do not know how to communicate the potential to the organization (Adrian & Genovese, 2011). If CTOs, CIOs and their subordinates had a better understanding of analytics, they could brainstorm ways in which the technologies could be used to improve decision-making. In addition to the BI and AI capabilities, developed primarily in medicine and finance, organizations should prepare to take advantage of natural language processing, pattern recognition, pattern matching, the ability to process large volumes of data, and rich media types.

The participants were selected from software teams who are currently using agile software development methods. Some forms of DDD may already be in use in agile software development environments because agile software methods were developed to

improve productivity (Schwaber, 1995). Once the current DDD methods are identified, they could be shared, which would increase the number of people trained to use these tools and techniques. This research is intended to provide a better understanding of DDD as a tool to improve productivity in the context of agile software development.

Nature of the Study

Software development failure rate is high and agile software methods were developed to improve software development success. Although DDD can improve organizational output and productivity, organizations need to define DDD within the context of the problem. Based on a review of the literature on DDD and agile software development, the research on DDD as a tool to improve software development productivity is in the initial stages; therefore, qualitative research methods will be used to explore the meaning of DDD within the context of agile software development.

Qualitative research methods are used when more needs to be known about a topic (Patton, 2002), when the nature of the research is exploratory, and when there is insufficient data available to develop hypotheses (Sullivan, 2001). The qualitative research methods used for this research are the interpretive phenomenological analysis (IPA) research methods described by Smith et al. (2009). The IPA methods are based on the philosophical views of Husserl, Heidegger, Merleau-Ponty and Sartre, the hermeneutics, which are based on the philosophic views of Schleiermacher, Heidegger, and Gadamer, and idiography. According to Smith et al., the IPA researcher believes that each individual develops perspectives through their own unique experiences. The IPA researcher interprets the meaning of the phenomenon by examining the part in relation to

the whole and the whole in relation to the part and the IPA researcher is focused on explaining the phenomenon in relation to the individual rather than in relation to the group.

Qualitative data was obtained by interviewing software managers, project managers, and agile coaches. The research participants were selected based on their familiarity with agile software development methods, their experience as software managers, project managers, and agile coaches, and their interest in participating in the research study. The interviews were transcribed and analyzed to identify major themes, common responses, and unique responses to the interview questions. The interviews and the literature review served as the basis for my interpretation of the phenomenon of DDD as a tool to improve software development productivity in an agile software development environment.

Definition of Terms

The research on management understanding of DDD used the following operational definitions. Operational definitions describe the concepts measured in a research study (Sullivan, 2001). The purpose for operational definitions is to indicate which words will be used to define terminology within the framework of the research study.

Agile: Agile is used to describe a software development framework that includes multiple processes including Scrum. All of the agile software development processes emphasize collaboration, teamwork, adaptability, and frequent and iterative software delivery (Cohn, n.d. a).

Crystal: is a set of people centered rather than process centered agile software development methodologies (Cockburn, 2008). Data Driven Decision Making: Data, analytics, and the knowledge management processes for knowledge creation, accumulation, retention, and transfer (Brynjolfsson et al., 2011).

Descriptive analytics: Answer the questions what happened and what is happening and are used to measure and manage performance. Examples include reports, dashboards, and scorecards (Salam & Cearley, 2012). Descriptive analytics may be used to identify alternative solutions but may not provide an optimal solution (Turban, Sharda, & Delen, 2005).

Design improvement: is an extreme programming (XP) software development practice that is based on the concept of continuous improvement. Software developers are expected to refactor or optimize the code design with each iteration (Jeffries, n.d.).

Diagnostic analytics: Answers the questions why did it happen and what are the key relationships. Diagnostic analytics are used to understand outliers and variance, to create profiles, and to classify data. Examples include machine learning, interactive visualization, data mining and modeling, and content analytics (Salam & Cearley, 2012). Diagnostic analytics may be used to identify the underlying causes for irregularities (Turban et al., 2005).

Dynamic systems development methodology (DSDM): is an agile software project management methodology developed by the DSDM consortium. (DSDM consortium, n.d.).

Extreme Programming (XP): “Extreme Programming is a discipline of software development based on values of simplicity, communication, feedback, courage, and respect. It works by bringing the whole team together in the presence of simple practices, with enough feedback to enable the team to see where they are and to tune the practices to their unique situation” XP practices include “simple design, pair programming, test-driven development, and design improvement” (Jeffries, n.d.).

Feature driven development (FDD): is an agile software methodology consisting of 5 iterative activities beginning with developing a model of the system, followed by developing an organized list of features. A subset of the features is selected for the next iteration and then the selected features are designed and built. The process is repeated until all of the features described in the model are built (Ambler, n.d.).

Knowledge accumulation: the process of acquiring, capturing, or obtaining knowledge (Gold, Malhotra, & Segars, 2001).

Knowledge creation: the process in which explicit and tacit knowledge is shared between individuals and groups within an organization through socialization, externalization, combination, and internalization (Nonaka & Takeuchi, 1995).

Knowledge Management (KM): The first generation of KM is “a systematic discipline and set of approaches to enable information and knowledge to grow, flow, and create value in an organization” (Rao, 2005, p. 3). The definition of the second generation of KM is “information in action” (O’Dell & Hubert, 2011, p. 2).

Knowledge retention: the process of organizing and preserving or storing knowledge (Mansour, Alhawari, Talet, & Al-Jarrah (2011).

Knowledge transfer: the process of distributing knowledge to people other than those who generated, produced, or created the knowledge (Mansour et al., 2011).

Lean software development methodology: is based on the manufacturing processes developed by Toyota and like agile software methodology, lean is focused on people rather than on processes (Bielicki, n.d.).

Model driven development (MDD): is an iterative software development methodology; however, unlike agile software methodology, which is based on communication and collaboration, MDD requires that models of the system be developed before the software is coded.

Pair programming: An XP software development practice in which two programmers sit side by side to develop the same code (Jeffries, n.d.).

Predictive analytics: Answers the questions what will happen, how risky is it, and what if it happened. Predictive analytics are used to forecast and test hypothesis and to model risk. Examples include forecasting applications, predictive models, and content analytics (Salam & Cearley, 2012).

Prescriptive analytics: Answers the questions what is the best option, how can an optimal solution be reached, and what should happen. Prescriptive analytics are used for risk management, business optimization, and recommending the best action. Examples include modeling, simulation, optimization, and visualization (Salam & Cearley, 2012).

Scrum: “Scrum is an agile approach to software development. Rather than a full process or methodology, it is a framework. So instead of providing complete, detailed descriptions of how everything is to be done on the project, much is left up to the

software development team. This is done because the team will know best how to solve the problem they are presented” (Cohn, n.d., b, para. 2).

Simple design: An XP practice, which encourages simple but adequate software design that ensures continuous improvement, can be made to working software (Jeffries, n.d.).

Test-driven development: An XP software development practice in which software is tested immediately after each small code module is developed to ensure working code is delivered with each cycle or iteration (Jeffries, n.d.).

Traditional software methods: are software development methods that are focused on process rather than on people and managing explicit knowledge, such as the waterfall method (Bjornson & Dingsoyr, 2008).

Assumptions

The research was based on several assumptions. First, the communication between the myself and the research participants was open and honest because the research participants were assured of privacy and their identities will not be made public. Second, the research participants knew enough about the situation in their software organization to propose solutions for the future. Third, given DDD definitions, the research participants were able to identify examples of DDD as a tool to improve software development productivity. Fourth, the research participants may have had different opinions on what data is needed to design and produce software products.

Scope and Delimitations

The focus of the research study was limited to an in-depth understanding of the phenomenon of DDD to improve software development productivity in an agile software development environment. The research included measuring how frequently the research participants identify analytics and KM as a potential tool for improving software development productivity in each software development activity. The research study did not include measuring how well DDD is used to improve software development productivity or how well DDD is used to improve product design or development.

The findings from the research study may or may not be generalizable beyond the population under study. The qualitative research study included research participants who work on local software projects and software projects in other U.S. locations. The participants were selected based on their in-depth knowledge of agile methods in software organizations.

Although there are other agile software methods, Scrum methods were selected for the research study because of the popularity of Scrum (Rao et al., 2011). In addition to Scrum methods, the research participants discussed other software development methods because, “Scrum is an agile project management framework that can be used alone or in coordination with any Agile process or processes” (Northern, Mayfield, Benito, & Casagni, 2010, p. 3). Scrum methods are frequently used with other software development methods, which means that the research participants could have discussed other software development methods.

Limitations

The research questions were limited to software management's understanding of DDD, which includes software management's understanding of analytics and KM to improve software development productivity. The research participants were limited to software managers, project managers, and agile coaches in the United States who use agile software development methods. The use of analytics and the combined use of analytics and KM to improve software development productivity are relatively new and a limited number of research studies were found on the use of DDD to improve software development in either a traditional software environment or an agile software development environment.

Significance of the Study

Cappelli and Kowall (2011) stated "agile software development methods are pushing software changes to the market faster" (p. 8). If change is introduced more quickly by agile software methods, then agile software managers may need to make decisions faster. DDD may enable agile software managers to make decisions at the speed of change.

If DDD improves organizational output and productivity then organizations can benefit from a better understanding of DDD. A review of the literature indicated that there is no universal definition for DDD and that the definition of DDD is dependent upon the context. A better understanding of DDD in software organizations could enable software organizations to find ways to improve output and productivity. The meaning of

DDD may expand and mature as software organizations discover the potential for analytics for both software product design and software development.

Positive social change could result from a better understanding of DDD in an agile software development environment. If DDD, which includes data, analytics, and KM, enabled agile software managers to make better decisions, software development productivity may be improved, and software organizations would be better able to compete in a global economy. If software development productivity were improved, software organizations may create more products that take advantage of the advances in hardware and software organizations may create more jobs.

Summary and Transition

A qualitative research study on the phenomenon of DDD in the context of agile software development was discussed in this chapter. The software project failure rate continues to be higher than desired for an applied discipline (Emam & Koru, 2008). Software project success needs to be improved if organizations are to remain competitive in a global economy. Software organizations depend upon trained people who know how to use methods and tools to improve software development productivity (CMMI Product Team, 2010).

Software organizations may improve productivity by using Agile software development methods, which were developed to improve software development productivity (Schwaber, 1995) and software organizations may use DDD as a tool to improve decision making because Brynjolfsson (2001) found that organizational output and productivity was improved when organizations used DDD as a tool to improve

decision making. However, a better understanding of DDD in the context of agile software development may enable software managers to find ways to use DDD to improve software output and productivity. Brynjolfsson et al. proposed that DDD is related to KM and Chan and Thong (2010) found that three agile practices were positively related to three KM practices; however, additional research was needed to understand the meaning of DDD within the context of agile software development.

The current literature on DDD, software methods, and KM was reviewed and the results of this literature review are discussed in Chapter 2 of this dissertation. The process used to review the literature is discussed at the beginning of the next chapter followed by a review of the literature on each topic. Although research could be found on each topic, few studies examined the topics of DDD, agile software development methods, and KM in combination.

Chapter 2: Literature Review

According to Emam and Koru (2008) organizations could benefit from reducing the combined software project cancellation and failure rate, which they claimed was between 24% and 36%. The problem investigated in the literature review was software development productivity, which included reviewing the literature on the tools, methods, and processes people are trained to use to improve software development productivity. The literature on traditional software development, agile software development, analytics, and KM was reviewed and analyzed to identify the common themes and to identify the need for additional research.

The purpose for this literature review was to gain insight into the tools, methods, and processes people are trained to use to improve software development productivity. This literature review includes a review of the literature on traditional software development methods and agile software development methods because agile software development methods were intended to improve software project success (Rao et al., 2011) and to improve software development productivity (Schwaber, 1995). The literature on DDD, which includes data and analytics, was reviewed because DDD is a tool that improved organizational productivity (Brynjolfsson et al., 2011) and may improve software development productivity. The literature on KM for software development was reviewed because according to Brynjolfsson et al. DDD is likely related to the KM processes of “knowledge creation, accumulation, retention, and transfer” (p. 4). Consequently, a better understanding of KM within an agile software environment

may lead to a better understanding of DDD as a tool to improve software development productivity within an agile software environment.

The process used to review the literature is discussed followed by the conceptual foundation for the research study on management's understanding of DDD. A more in depth discussion of the current literature on DDD, agile software methods, traditional software methods and KM, and agile software methods and KM follows. See Appendix B for a comparison of traditional software methods to agile software methods. The literature review concludes with a discussion of the research method, research approach, and research process used for the research study based on a review of the methods, approaches, and processes discussed in the current literature.

The Literature Search Strategy

An iterative process was used to review the literature for the research study. The literature review process was based on advice from the Walden University library staff demonstrating search techniques at residencies and the techniques on how to conduct a literature review discussed by Machi and McEvoy (2009). Multiple libraries were searched for journal articles including the Walden library, corporate libraries, organization libraries, and public libraries. Keyword searches were used along with subject searches and author searches for primary and secondary sources.

The topics searched included *software*, *software development productivity*, *agile software*, *analytics*, and *knowledge management*. Searches were based on each topic and then on the topics in combination. The articles were reviewed for relevance. In some

cases articles were eliminated based on the abstract. In other cases, articles were eliminated based on the contents of the article.

The relevant articles were reviewed and critiqued for validity and reliability. The articles were categorized based on the type of article. In some cases the articles presented the author’s view of the topic based on a review of the literature and in other cases the articles presented the results of research. Figure 2 shows the number of articles found by topic. Although many articles discussed the topic of analytics and a few articles discussed the topics of software development and KM and software development and analytics, little research has been conducted on analytics in an agile software environment. This research is expected to begin to fill this gap in the literature.

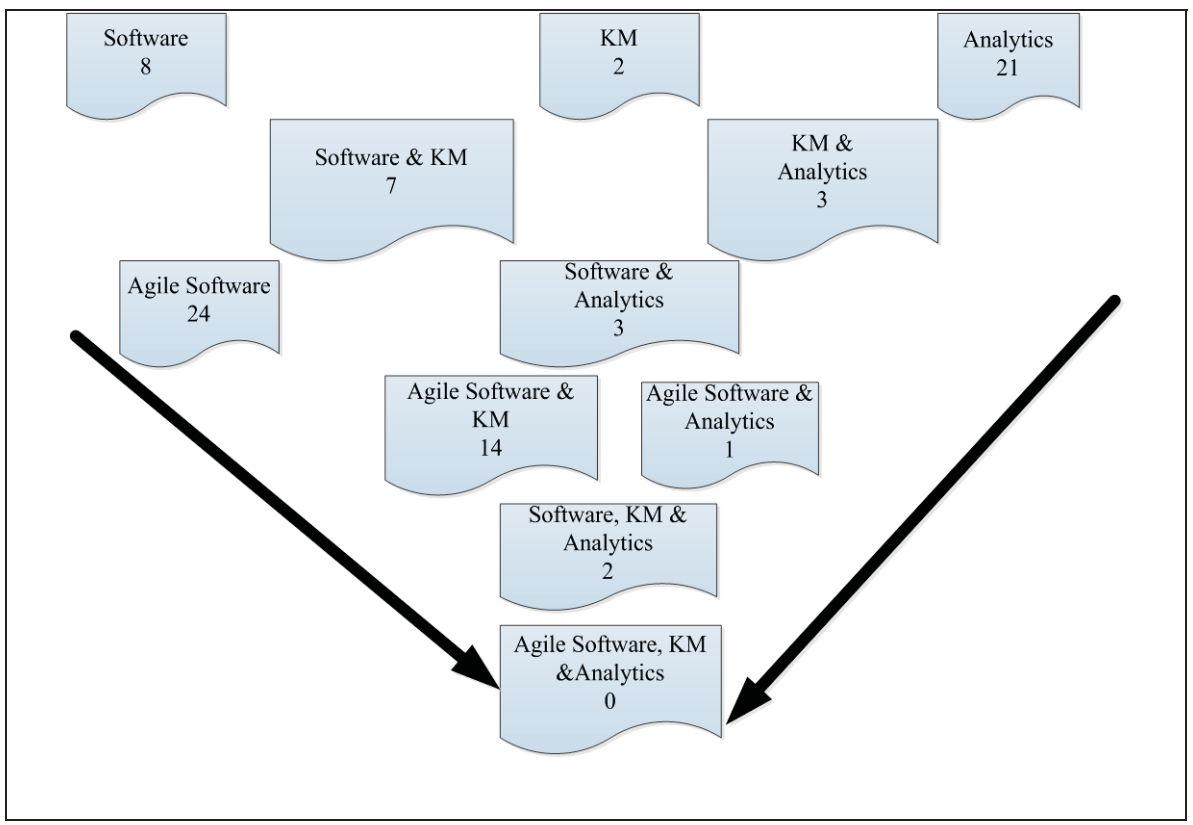


Figure 2. Literature Review: Number of Articles by Topic

Organization of the Review

The literature review was based on a review of general topics to more specific topics. The topics explored included the more general topics of software development, knowledge management, and analytics to the more specific topics of knowledge management in traditional and agile software environments and analytics to improve knowledge management and software development. The literature review culminated in a search for articles that focused on the combined use of knowledge management and analytics in a traditional software development environment and the use of knowledge management and analytics in an agile software development environment.

Conceptual Foundation

The results of the systematic literature review are discussed in this section of the dissertation proposal. The current understanding of DDD is that more needs to be known about the meaning of DDD and that the meaning of DDD is context dependent. The current research on traditional software development methods and agile software development methods are discussed as well as the current research on the use of KM to improve software development productivity. Although some research was found on the use of analytics and KM to improve software development productivity in a traditional software environment, no research was found on the use of analytics and KM in an agile software environment to improve software development productivity.

The research methods, approaches, and processes used in the current literature to study DDD, traditional software development methods, agile software development methods and KM are discussed and the rationale for the research methods, approaches,

and processes used for this research study are presented. Although both quantitative and qualitative research methods were used to study DDD, traditional software development methods, agile software development methods, and KM, only two qualitative research studies and one quantitative research study were found on the use of analytics in a traditional software environment (Hullet et al.,2011, Siwen & Jun, 2010, Zare & Akhaven, 2009), only one qualitative research study was found on the use of analytics in an agile software development environment (Abouelela and Benedicenti, 2010), only one mixed-methods research study and one qualitative research study were found on the use of analytics and KM in a traditional software environment (Abdullah et al, 2011b; Jiang, Eberlein, and Far, 2008), and no research studies were found on the use of analytics and KM in an agile software environment. This qualitative research study is intended to begin to fill this gap in the literature by exploring management's understanding of DDD as a tool to improve software development productivity in an agile software environment.

Current Understanding of Data Driven Decision Making

Based on a review of the literature, there are many definitions for DDD. Some of the definitions found in the literature will be reviewed in this section of this dissertation. If organizations need to define DDD, they need to be aware that the definition of DDD depends upon the context. Once DDD is defined, organizations, including software organizations, may be better able to brainstorm ways to use DDD to improve software development productivity.

DDD definitions. Although Brynjolfsson et al. (2011) equated DDD to “data and business analytics” (p. 1), based on a review of the literature, there does not appear to be

a consistent and universally understood definition of DDD. DDD was referred to as a DSS (Hedgebeth, 2007) and as business intelligence (BI) according to Ivancenco, Boldeanu, and Mocanu (2010). DDD was also referred to as both DSS and BI (Ow & Morris, 2010) and as competitive intelligence (CI) according to Bartes (2011).

Chandler et al. (2011) claimed that organizations should define analytics because there are many meanings of analytics. For example, the meaning of business intelligence, performance management, and analytics can be confusing. Organizations need to define the scope of any business intelligence, performance management, or analytics project to reduce confusion.

DDD was described as analytics and analytics was described as a continuum beginning with descriptive analytics and ending with predictive analytics (Salam & Cearley, 2012). Descriptive analytics are used to describe what happened in the past and what is happening in the present. Diagnostic analytics are used to identify cause of historical events. Predictive analytics are for what if analysis and to test hypothesis and prescriptive analytics are used to recommend an optimal solution.

According to Salam and Cearley (2012), Gartner defined advanced analytics as the use of statistics, data mining, simulation, and optimization to analyze text, images, audio, and video. Advanced analytics produce insights that cannot be accomplished with queries and reports. However, “analytics means different things to different groups within organizations and across the market” (Herschel, Hostmann, Rayner, & Bitterer, 2010, p. 2). Organizations should not seek to reach consensus on a single definition for

analytics; instead, organizations should ensure that the definition for analytics is clear for each initiative or project that uses the term (Herschel et al., 2010).

Analytics refers to a specific advanced BI capability or technique, such as, neural network or self-learning algorithms and not to less advanced BI capabilities, such as, reporting or querying. Analytics refers to the process of using analysis to solve a business problem, such as, creating insight into how to create customer loyalty without specifying a specific BI technique or capability. Analytics means a specific packaged BI application, such as, “web analytics, marketing analytics, or supply chain analytics” (Herschel et al., 2010, p. 3). Analytics refers to the entire domain including BI, analytic applications and performance management.

However, several different BI definitions were discussed along with the analysis of several maturity models that could be used to measure organizational BI maturity (Rajteric, 2010). Additional work would be needed to use any of the maturity models alone or in combination to measure organizational BI maturity. Organizations need to define BI before developing a BI maturity model.

BI was defined as a KM process (Ivencenco et al., 2010). BI is “the process of transforming data into information and then into knowledge. Business Intelligence systems are specialized tools for data analysis, queries and reporting, that support management in the decision-making process” according to Ivencenco et al. (2010, p. 51). BI is intended to improve strategic decision-making rather than to improve daily tactical decision-making.

Although the literature contains alternate DSS definitions Hedgebeth (2007) used the dssresources.com definition of a DSS and Ow and Morris (2010) discussed the need for additional research to determine the cultural specific DSS design and development needs. Bassi (2011) claimed that the meaning of *HR analytics* means different things to different people. HR analytics consist of a set of tools and methods that provide HR statistics as well as predictive analytics. HR analytics provide an evidence-based approach to management on the people side of the business. Although HR does not yet have the skills and knowledge, Bassi argued that HR should lead IT and Finance to implement HR analytics. However, if HR is not prepared to lead the effort to implement HR analytics then IT and Finance need to be prepared to take the lead.

Environment and context matter. Ow and Morris (2010) conducted a quantitative research study using policy capturing methodology to determine how chief technology officers (CTOs) consider, weigh, and integrate data for decision making. Ow and Morris found that CTOs used some but not all of the data they thought they would use to make strategic technology decisions. However, additional research may be needed to determine how decision makers consider, weigh, and integrate data for decision-making. For example, Ow and Morris stated that it is possible that decision makers used heuristics to make decisions when time, knowledge, and computational power were limited.

The meaning of DDD depends upon the context. Ferrand, Amyot, and Corrales (2010) stated that context affected the BI definition for healthcare safety. Rajteric (2010) recommended that organizations define BI before developing a BI maturity model. Yeoh

and Koronios (2010) found that BI critical success factors or CSFs were not likely to be generalizable due to dependence on context and if a more universal definition of BI emerged, organizations would be better able to compare BI maturity across organizations.

The context of this research study is software development. According to Emam and Koru (2008) software development failure rates are high. Approximately 26%-34% of software projects surveyed were cancelled or failed. The most common reasons for software project failure were changes in scope, requirement changes, lack of senior management involvement, budget shortages, and lack of project management skills. The most difficult problem in software development was software development scheduling. Emam and Koru claimed that software estimating, scheduling, and management tools need to be improved and the techniques need to be improved.

KM and DDD. DDD is related to KM because DDD requires knowledge creation, accumulation, retention, and transfer (Brynjolfsson et al., 2011). Individuals are able to use explicit knowledge because explicit knowledge is codified, which means that the knowledge must be captured, organized, stored, and easy to retrieve. Individuals must communicate to share tacit knowledge, which is only in the minds of the individuals who have developed the expertise.

Multiple definitions for KM can be found in the current literature; however, KM is generally defined as the intentional reuse of knowledge to improve organizational process and performance (Mansour et al., 2011). The KM objective is to manage the knowledge that will result in improvements, such as, improved productivity, creativity, and competence rather than to manage all knowledge. Brynjolfsson et al. (2011)

questioned how organizations could retain proprietary knowledge while sharing knowledge within and between organizations.

KM is not about building a repository of knowledge; KM is about “people, process, and technology” (Molaei, 2011, p. 426); however, small organizations may benefit from sharing knowledge with other organizations by developing common knowledge repositories. Molaei (2011) recommended that small organizations share knowledge with other organizations to increase the available expertise. Organizations could minimize the risk of sharing information outside the organization by sharing with similar organizations that do not compete in the same geographic area, who do not have a direct effect on profitability, or who do not have profit as a motive. For example, nonprofit organizations could develop a common knowledge repository, individuals in human resources could develop a common knowledge repository, or individuals who share a common interest, such as agile software development methods could develop a common knowledge repository.

Many authors proposed KM models with as few as three KM processes and as many as seven KM processes (Mansour et al., 2011). To reduce the confusion between KM models, a general KM framework was proposed by Mansour et al. (2011). The general KM model consisted of 10 KM processes including, identifying the need for KM, KM goal review, knowledge identification, knowledge acquisition, knowledge validation, knowledge storage, knowledge distribution, knowledge application, knowledge retention and update, and knowledge training.

Artificial intelligence and KM. Smith and Farquhar (2000) described a ten-year roadmap for KM, which predicted that artificial intelligence (AI) could be incorporated into a KMS. The purpose for the KM roadmap was to encourage the AI community to conduct the research needed to incorporate AI into KM initiatives. Smith and Farquhar claimed that KM could utilize the lessons learned from AI to improve KM knowledge acquisition, representation, and inference. AI could be used to improve KM search capabilities, intelligent agents could be used to improve knowledge retrieval and notification and AI could be used to facilitate the implementation of distributed problem solving technology.

According to Smith and Farquhar (2000) expert systems were intended to provide expert solutions to problems while KM was intended to provide people with expert support to solve problems. Although initially both AI and KM suffered from unmet expectations, Smith and Farquhar claimed that KM had been adopted by a number of organizations. Consequently, “if the AI community is able to develop something of value in this area—the ‘killer app’ for knowledge management—there is an audience waiting to use it” (Smith & Farquhar, 2000, p. 22).

Trends in analytics. The use of real-time analytics to support strategic decision-making will increase (Cappelli & Kowall, 2011). Organizations need to be aware of the interaction between hardware and software because agile software development methods are pushing software changes to the market faster. Laney (2012) provided 10 reasons why organizations should go beyond basic BI capabilities, such as, reporting and querying because “tactical and operational decisions must increasingly be made at a rate

faster than humans are capable of” (p. 2). Organizations should consider advanced analytics, such as, rules and artificial intelligence to:

- Benefit from big data.
- Identify weak signals.
- Embrace complexity, unexpected activity and changing conditions.
- Understand unstructured data.
- Optimize business processes.
- Automate governance, risk and compliance reporting.
- Enable full-sample forensics.
- Evolve to insight and foresight.
- Enhance scenario planning.
- Instigate innovation. (Laney, 2012, p. 2)

Rayner (2011) predicted that over the next 40 years, advanced analytics would mature and take over management decision making while management decision-making will focus on setting strategic direction, innovation, and analytics. Organizations should take advantage of the existing capabilities of advanced analytics. For example, personnel decisions can be improved by using software systems that incorporate advanced analytics. Rayner recommended that organizations use collaborative decision making to brainstorm ways to use analytics, such as, machine learning, predictive analytics, and modeling and simulation.

The use of collaborative decision-making (CDM) will increase due to the economic downturn and reduced travel budgets; however, management may resist adopting CDM if the increased transparency is feared (Schlegel, Salam, Austin, & Rozwell, 2009). CDM combines BI with social networking and Schlegel et al. (2009) stated that CDM is best used for “nonroutine, complex decisions that require iterative human interactions” (p. 1). Organizations have and will continue to increase their use of analytics for performance management in many domains including finance, HR, sales, marketing, and IT (Chandler, 2011).

CDM platforms will increase in use within the next five to 10 years for both strategic and tactical decision-making (Chandler, 2011). Over the past year, Chandler (2011) stated that several BI software vendors have already improved the ability for decision makers to collaborate. IT organizations, including software organizations, should be able to develop templates to improve collaborative decision-making. According to Chandler the most difficult barrier organizations need to overcome to increase the use of collaborative decision-making is cultural. If CDM is more likely to thrive in less hierarchical and open organizations, then agile software development organizations could provide an optimal environment for CDM.

Organizations need to select mobile application vendors based on their ability to incorporate analytics (Clark & King, 2011). Managers need to learn about four trends in business analytics that will rapidly change the assumptions about BI (Gassman, Salam, Bitterer, Hagerty, & Chandler, 2011). These trends include the increased use of mobile and tablet devices as a BI platform. The way in which information feeds decision making

will change in the next few years, organizations will change how and where they procure business analytics, and organizations will change the types of BI and analytics they use. Most importantly, the applications and technologies for business analytics were predicted to change frequently in the next few years.

Benefits of a better understanding of DDD. The technical papers provided by Bartes (2011), Hedgebeth (2007), and Ivancenco et al. (2010) discussed the potential benefits of improved BI and DSS. The quantitative research discussed by Brynjolfsson et al. (2011) measured the potential organizational productivity and profitability of DDD. The quantitative research discussed by Ow and Morris (2010) measured the factors decision makers used to make strategic technology decisions.

A better understanding of DDD could enable organizations to develop maturity models, to define CSFs, and compare DDD across organizations. If organizations that use DDD were more productive and profitable than organizations that do not use DDD (Brynjolfsson et al., 2011) then organizations may benefit from a better understanding of DDD. Software development organizations may be able to brainstorm how to use DDD to improve software development if they had a better understanding of DDD.

Current Research in Software Methods

The current literature on traditional and agile software development methods was reviewed and the findings were summarized in this section of this dissertation. The research in agile methods is in the initial stages; therefore, the focus has been on determining what agile means and to what degree agile software development is complex. The current literature in agile software methods was limited to two software

development processes: the requirement engineering process and the software release process that provides opportunities for future research in agile software methods focused on other software development processes, such as, software management, design, development, and test.

The transition from traditional software methods to agile software methods has also received some attention in the literature. While three agile methods were compared, extreme programming (XP), dynamic systems development method (DSDM), and Scrum (Rao et al., 2011), additional research is needed to identify the strengths and weaknesses of other agile methods and to determine how agile methods compare to traditional software development methods. Although Roa et al. (2011) and Zhang and Patel (2011) found that agile methods were best for small projects, Roa et al. proposed that larger projects could be broken into multiple smaller projects and Zhang and Patel proposed that agile methods could be combined with model driven development (MDD) for larger projects.

Traditional software development. Software project failure rate is too high (Emam & Koru, 2008) and software development productivity has not kept pace with advancements in hardware (Fitzgerald, 2012). Software development productivity has generally been defined as the ratio of inputs to outputs and organizations have traditionally measured software development productivity by the ratio of lines of code (LOC) produced to the number of person months consumed (Sudhakar, Farooq, & Patnaik, 2012). Software development productivity is dependent upon people, process, and tools (Wadhwa & Mitra, n.d.).

Rodger, Pankaj, and Nahouraii (2011) examined data from 138 organizations from 1989-2001 to determine the factors influencing software development productivity and time. Rodger et al. (2011) concluded that 4GL languages increased productivity and decreased development time; ICASE tools did not affect productivity or development time, as team size increased productivity and development time increased and as platform complexity increased productivity increased and development time decreased. Contrary to Rodger et al., Dubey (2011) proposed that CASE tools should be integrated and used to prototype software to improve software development productivity.

Hewagamage and Hewagamage (2011) argued that software success needed to improve and software research that could lead to improved software success could benefit from consistently defined terminology and consistently defined relationships between software framework components. Hewagamage and Hewagamage proposed a general software development framework for IT project management based on their review of the Capability Maturity Model-Integrated (CMMI), Project Management Body of Knowledge (PMBOK), Projects in controlled environments (PRINCE2), IT Infrastructure Library (ITIL), and Microsoft Solutions Framework (MSF) frameworks. The general software development framework incorporated the project management phases defined by the PMBOK and the software engineering phases defined in the SWEBOK (2004). The phases defined in the PMBOK are starting, planning, executing, and closing. The phases defined in the SWEBOK (2004) are requirement engineering, software design, software implementation, and software testing and deployment.

Although individual software engineers were able to improve the accuracy of their estimates to complete tasks when they were provided with historical data on their own performance, their productivity did not improve (Elminir, Khereba, Elsoud, & El-Hennewy, 2009). The Personal Software Process (PSP) was developed to enable software engineers to measure their productivity and the quality of their work. The skilled engineers were able to reduce interruptions and increase the quality of the software delivered while management was able to identify the least skilled engineers and remove them from the project. Elminir et al. (2009) assumed the engineers would accurately self-report using the PSP; however, the PSP may foster competition rather than cooperation between team members.

Churchman inquiry systems. Linden et al. (2007) stated that there is a lack of continuity in the design of information systems and knowledge management systems (KMS) and they proposed that Churchman's inquiry systems could provide a theoretical basis for future information systems and KMS research. Linden et al. summarized each of the five inquiry systems proposed by Churchman in order to enable the reader to understand the Leibnizian, Hegelian, Kantian, Lockean, and Singerian inquiry systems without having access to Churchman's out-of-print book. Linden et al. explained the philosophical viewpoint of each of the inquiry systems and compared the inquiry systems to enable the reader to understand the key characteristics of each inquiry system.

Linden et al. (2007) claimed more is required to be known about Churchman's inquiry systems based upon their own research and the research found in the literature. As a result of their analysis, Linden et al. compared seven opportunities to apply each of

Churchman's inquiry systems to information system and KMS design and development including input, given, process, output, guarantor, IT support, and applicable situations. For example, input to a Lockean inquiry system would be based on elementary observations while the input to a Singerian inquiry system would be based on units and standards. The output from a Lockean inquiry system would be taxonomy while the output from a Singerian inquiry system would be a new standard or exoteric knowledge.

The Leibnizian inquiry system does not accept input and knowledge is deductive (Linden et al., 2007). A system that checks medication dosages recommended by physicians is an example of a Leibnizian inquiry system. A Lockean inquiry system accepts input, knowledge is inductive, and properties are labeled. Google's image labeling database is an example of a Lockean inquiry system. A Kantian inquiry system has the same characteristics as a Lockean inquiry system and a Kantian inquiry system uses models to find the best fit for the data. A Hegelian inquiry system has the same characteristics as a Kantian inquiry system and a Hegelian inquiry system is able to synthesize conflicting theses to arrive at a new thesis.

Information systems as wicked systems. According to Linden et al. (2007) information systems that are developed in complex environments where stakeholders have different perspectives are referred to as wicked systems. "Wicked situations are characterized by the multiplicity of stakeholders involved, the pervasive nature of conflicts among their perspectives, the lack of firm criteria for determining an optimal answer and the complex interconnectedness of numerous problem elements" (Linden et al., 2007, p. 863). The input to a Hegelian inquiry system design represents the different

perspectives of the stakeholders and the different perspectives are synthesized to account for opposing views.

Linden et al. (2007) stated that the Singerian inquiry system is based on the Leibnizian, Hegelian, Kantian, and Lockean inquiry systems. Although the Leibnizian, Hegelian, Kantian, and Lockean inquiry systems do not adequately address real world whole systems; the Singerian inquiry system is holistic and agile. The Singerian inquiry system addresses whole systems and is open to change when new information becomes available. Singerian inquiry systems generate exoteric knowledge, which is knowledge that is intended for a broad audience as opposed to esoteric knowledge, which is intended for a narrow audience.

As organizations are faced with more complex environments it is more likely that information systems will require methodology tailored for wicked systems development. Linden et al. (2007) described an information system design approach based on Churchman's Hegelian inquiry system. However, Linden et al. stated that the Singerian inquiry system is the most appropriate inquiry system for designing wicked information systems.

The pursuit of actionable knowledge. According to Linden et al. (2007) inquiry is the "process of searching for the truth, that is, for facts, information and knowledge" (p. 837) and actionable knowledge enables the decision maker to "act effectively within a domain of interest" (Linden et al., 2007, p. 838). Lingling et al. (2009) defined actionable knowledge as knowledge that has been transformed from rough knowledge. Rough knowledge is data that was mined from a data warehouse. Lingling et al. claimed

that rough knowledge should be transformed to make it actionable. DDD may be synonymous with inquiry and the pursuit of actionable knowledge.

Linden et al. (2007) agreed with Churchman that information system researchers should make moral and ethical decisions when designing information systems. “The designer is moral if he or she serves a client who has a legal or moral right to expect that the system will serve the client’s interest and these interests themselves are legal or moral” (Linden et al., 2007, p. 847). Linden (2010) developed a website based on Churchman’s Singerian inquiry system and the Connectedness Caretaker Principle and Linden concluded that the website was an ethical platform because the research participants were required to consider the ethical implications of their decisions.

Linden et al. (2007) defined five design characteristics of Churchman’s inquiry systems. The data that would be needed to design an information system based on Churchman’s inquiry system design characteristics includes a software development methodology, which would provide a framework that is generalizable and repeatable. Data would be needed to determine the differences between the user’s behavior patterns and data would be needed to estimate how well the user’s behavior met the overall system goals. Data would also be needed to communicate the goals to the software development team so that the information system design reflected the goals and data would be needed to ensure the integrity of the whole system was maintained.

Nonseparability and decomposition principles. Nonseparability and decomposition refer to the relationship of the parts of a system to the whole system (Linden et al., 2007). Although the integrity of an information system is dependent upon

the relationship of the parts to the whole system, an information system should be designed so that the parts are separable. The integrity of an information system is dependent upon how the information system adapts to change and data would be needed to ensure that the designer could “predict the effects that the change will have on the overall system performance” (Linden et al., 2007, p. 848).

Linden et al. (2007) claimed that Churchman stated “human intuition can be faulty” (Linden et al., 2007). Although Churchman and Brynjolfsson et al. (2011) agreed on the weaknesses of intuition, Churchman hypothesized that human intuition could be valuable if it could be incorporated into an information system. Brynjolfsson et al. found that instead of relying on intuition, DDD improved organizational performance and profitability. If software management understanding of DDD can be better understood in agile software development environments, then software managers may be able to use DDD to improve the development of wicked systems.

Agile software development. Quality and productivity may be improved by using agile software methods, such as, XP (Layman et al., 2006). Agile software development methods were developed to improve software development productivity and to decrease the time-to-market (Ballou, 2008). Although the current research in agile software methods explored the opinions of agile projects managers toward agile methods, the meaning of complexity within agile software projects, the challenges of transitioning from traditional software methods to agile software methods, the need for models in agile software projects, RE, and test and release, metrics are needed to compare the benefits of

agile software development to the benefits of traditional software methods (Ballou, 2008).

Complexity in agile software development. Agile software development methods challenge the assumption that change and uncertainty are controlled by a high degree of formality; consequently, agile software development methods are focused on learning and innovation rather than on optimization and control (Nerur & Balijepally, 2007). Software development is frequently focused on complex problems that are difficult to resolve. Rather than knowing the solution at the beginning of a project, the solutions emerge as more is known about the problem space. Nerur and Balijepally (2007) argued that multiple perspectives should be considered, assumptions should be questioned. Conflict should be resolved through argumentation, and what if scenarios should be used to imagine and prepare for a preferred future state.

Pelrine (2011) identified which agile software development techniques were complex and which agile software development techniques were not complex based on responses from over 300 individuals involved in agile software development projects. The research participants were asked to classify the agile software project techniques based on the Cynefin sense-making framework, which has been used to classify activities as simple, complicated, complex, or chaotic. The research participants rated 21% of the agile software development tasks as simple or unknown and 79% of the agile software development tasks as complicated, complex, or chaotic.

Because the majority of agile software development tasks were considered complicated, complex, or chaotic, Pelrine (2011) proposed that software tasks, such as

estimating, benefit from a probe-sense-respond model rather than from a reductionist methodology. Pelrine claimed that “the ‘apply-inspect-adapt’ model of agile development is a probe-sense-respond model” (p. 36), which establishes system boundaries, determines what will work and what does not work and then adapts as more is learned about the evolving system. Pelrine stated that a deeper understanding of the relationship between complexity and agile software development is needed.

Sutherland et al. (2007) argued that agile software development methods are intended to manage change rather than complexity. Process discipline is needed to manage complexity. By using both the Capability Maturity Model – Integrated (CMMI) developed by the Software Engineering Institute (SEI) and agile software development methods, software teams can adapt to changing requirements and manage complexity by using a disciplined approach to process (Glazer et al., 2008; Sutherland et al., 2007). However, success is not guaranteed and software managers need to be aware of the risks associated with the transition from traditional software methods to agile software methods.

Transitioning from traditional methods to agile methods. There are multiple agile software methods and software managers need to be aware of the strengths and weaknesses of each agile method to select which agile methods to adopt (Qumer & Hendersen-Sellers, 2008; Rao et al., 2011). Software managers need to be aware of how decision making can be influenced when making the transition from traditional software methods to agile software methods (McAvoy & Butler, 2009) and software managers also need to be aware of the physical environment, including the room layout and noise

(Eccles et al., 2010). Software managers need to be prepared to tailor the agile processes to meet different needs (Clutterbuck, Rowlands, & Seamans, 2009), and to be flexible enough to adjust to the changing requirements of the software team throughout the transition process (Ganesh & Thangasamy, 2012).

McAvoy and Butler (2009) found that the Abilene paradox and groupthink influenced two software teams when they were making the transition from traditional software methods to agile software methods. Groupthink was defined as dysfunctional consensus in which a group agrees to a solution due to the perceived influence of one or more individuals. The Abilene paradox was defined as group decision-making based on unanimous agreement with a proposed solution; however, all of the group members silently disagree with the decision. Agile software development managers need to balance team cohesion and team empowerment to avoid the pitfalls of groupthink and the Abilene paradox (McAvoy & Butler, 2009).

Although Ionel (2009) found little empirical research on agile methodologies in the literature, Balijepally et al. (2009) conducted a study, which compared paired programming to individual programming on less complex tasks and more complex tasks. Balijepally et al. found that although paired programming methods did not improve performance, paired programming improved software quality. Although improved software quality may result in less rework, Balijepally et al. did not equate quality to productivity.

Based on the results of two different case studies, agile software methods were found to improve morale, which increased team creativity, problem solving (Omar, Syed-

Abdullah, & Yasin, 2011) and adaptability to changing requirements (Clutterbuck et al., 2009). Transitioning to agile software methods increased the need for communication with the team and between the team and external entities. Agile methods, such as XP encourage communication; however, inadequate communication was found to be at the root of all problems.

Organizations need to select the agile methods they will use when they transition from traditional software methods to agile software methods. Sharp, Robinson, and Petre (2009) found those organizations that transition to agile software development methods should consider the social and the notational effect of agile methods, such as use of story cards and the wall. Story cards are used to document requirements and the wall is used to display the story cards so that the work in progress is visible to the stakeholders. Organizations that choose to use automated methods to develop and display story cards may not benefit from the social benefits of face-to-face communication.

Rao et al. (2011) reviewed the literature on agile software development, conducted interviews and three case studies on software organizations in India to identify the agile software development methodologies in use and the issues experienced by these software organizations. Rao et al. found that extreme programming (XP), dynamic system development method (DSDM), Scrum, feature driven development (FDD), lean software development and Crystal were discussed in the literature; however, based on three case studies, Rao et al. were able to identify the pros and cons of XP, DSDM, and Scrum as shown in Table 1. Rao et al. also found that communication and coordination

was a challenge when there was more than one agile software team or when there were many stakeholders.

Table 1

Pros and cons of agile software development methodologies

	XP	DSDM	Scrum
Pros	Works well for small projects.	Technique independent process. Efficient use of time and budget. Requirements evolve over time.	Works well for small projects. Requirements can be prioritized.
Cons	Does not work well when limited to 1 developer due to pair programming requirements. May be difficult to identify all of the software problems because testing and development are conducted by the same person. Poor customer collaboration.	Requires end-user involvement, which may not be possible on all projects.	Team dynamics not improved if limited to 1 developer. Poor customer collaboration if customer is off-site.

Note. From Rao et al. (2011)

Rao et al. (2011) identified the following benefits of transitioning from traditional software methodology to agile methodology: “adaptability to change, short time frames of releases, continuous feedback from customers, high-quality and bug free software” (p. 43). Although agile software development methodologies worked best for small projects,

Roa et al. suggested that larger projects be broken down into several smaller projects.

Zhang and Patel (n.d.) proposed that agile methodologies could be combined with model driven development (MDD) for larger projects.

Software managers need to consider the people issues when transitioning from traditional software methods to agile software methods, Lalsing et al. (2008) found that there was a positive relationship between the size of the agile software teams and productivity. Based on the analysis of three case studies, the smaller team was able to deliver the required functionality on-time 90% of the time while the largest software team delivered the required functionality on-time 30% of the time. Lalsing et al. argued that managers should be aware of the exponential increase in communication channels as team size increases when transitioning from traditional software methods to agile software methods.

The perception in European software organizations was that some agile methods were more useful than other agile methods and some agile practices were more useful than other agile practices. Salo and Abrahamsson (2008) found that more European organizations had adopted agile XP practices than Scrum practices. The XP practices of open office space, a forty-hour work week, coding standards, continuous integration, and collective ownership were implemented more frequently than other XP practices and the practice of maintaining a software backlog was the most frequently implemented Scrum practice.

Models still needed. Khan, Al-Bidewi, and Gupta (2011) claimed that agile methodologies were developed to overcome the complexity of object-oriented

methodologies but agile has not successfully replaced the need for models. Khan et al. claimed that additional research was needed to develop an object-oriented methodology that works. Zhang and Patel (n.d.) described a Motorola case study that combined agile methodologies with MDD to develop a real-time telecommunication system. Software is iteratively developed in both MDD and agile methodologies. While documentation was limited based on agile methodologies, Zhang and Patel developed MDD models before the software was developed.

Zhang and Patel (n.d.) found that automating the software code development process based on the MDD models improved the software quality. Agility was also improved by streamlining the system engineering, development, and testing processes to ensure usable code was delivered after each cycle of testing. Zhang and Patel proposed using a combined MDD and agile methodology for large projects with multiple releases.

Requirement engineering in agile software environments. Lee and Xia (2010) and Ramesh, Lan, and Baskerville (2010) focused on how agile software teams developed software requirements. Lee and Xia claimed that agile software management must determine how to balance agility. Ramesh et al. identified two risks agile software managers must manage when developing requirements.

Lee and Xia (2010) used both quantitative and qualitative research methods to study the relationship between agile software team autonomy and diversity and the extensiveness of an agile software team's response to requirement changes and the efficiency of an agile software team's response to requirement changes. Lee and Xia also studied the relationship between the extensiveness of an agile software team's response to

requirement changes and project cost, schedule, and functionality. Lee and Xia found that that agile software requirement changes could have both positive and negative effects on on-time completion and on-budget completion; therefore, Lee and Xia recommended that agile software managers balance software team autonomy and diversity to successfully deliver the functionality that meets the customer expectations for quality, cost, and schedule.

Ramesh et al. (2010) conducted a qualitative research study to determine how agile requirement engineering (RE) was conducted in practice. Ramesh et al. conducted 16 case studies and Ramesh et al. interviewed managers, project managers, developers and others to obtain an in-depth understanding of the strengths and weaknesses of agile RE compared to traditional RE. Ramesh et al. (2010) “identified six agile RE practices and 7 challenges to RE” (p. 455), which were condensed into a list of nine agile RE practices and challenges.

Ramesh et al. (2010) compared how well nine agile RE practices and challenges mitigated risk to how well traditional RE practices mitigated risk. Three agile RE practices mitigated risk, three agile RE practices exacerbated risk, and three agile practices neither mitigated nor exacerbated risk. Two of the nine agile practices were considered intractable while seven of the agile RE practices were considered tractable (Ramesh et al., 2010). Intractable risks are risks that are difficult while tractable risks are easy to manage.

Ramesh et al. (2010) identified two intractable risks introduced by agile RE practices. The agile RE practice of modeling only functional requirements exacerbated

the risk of ignoring non-functional requirements and Ramesh et al. categorized this risk as intractable. Although agile RE practices encouraged customer participation, in some cases it was difficult or impossible to obtain customer concurrence and in some cases the customers lacked the required expertise. The negative impact on agile RE would be high if the customer participation was inadequate or if the customer lacked the required expertise and it would be difficult to mitigate the impact of this risk; therefore, Ramesh et al. categorized this risk as intractable. Ramesh et al. recommended that agile software managers select the RE practices based on the software engineering environment.

Software test and release in agile software environments. Agile software development is incrementally released which means that the software must be tested for each cycle or iteration. Test-driven development (TDD) methods have been used to ensure that software is tested as it is developed for each cycle or iteration. Shull et al. (2010) found that TDD improved the mean time to fix software based on an interview with a Microsoft manager whose teams use TDD.

Agile software managers depend upon the software team to report the burndown rate for each software cycle or iteration. The burndown is a measure of the work completed during each cycle or iteration. In addition to measuring the completed work for each iteration, Rinko-Gay (2009) recommended that agile testers report the number of tests in scope for the current build, the cumulative number of tests passed and failed, the cumulative number of open and closed defects and the total number of reopened defects for each iteration. At the end of the project, agile software teams should use Pareto

analysis to provide in depth analysis of the defects found, when and where they were found and the root cause for each defect (Rinko-Gay, 2009).

Smith (2011) discussed the Gartner philosophy of using agile methods to release software code into production for cloud computing. Trust between development and operations was defined as critical to successful software release for cloud computing. Smith claimed that software development for cloud computing required improved application lifecycle management (ALM), which could be accomplished by using automated regression testing and continuous integration to release software frequently while maintaining service levels.

Current Research in Software Development and KM

The current literature on traditional software development and KM revealed that a KM tool could potentially benefit the software architecture definition process, the requirement engineering process, and the software estimating process in a traditional software development environment. Additional research is needed to determine the applicability of the research findings on traditional software development and KM in an agile software environment. A review of the current literature on traditional software development and KM did reveal that it was feasible to use DDD in the form of intelligent agents to improve defect management in a traditional software environment.

Traditional software development and KM. Bjornson and Dingsoyr (2008) reviewed the literature from 1999 through August 2006 on software development and knowledge management. Bjornson and Dingsoyr discussed Buono's and Poulfelt's (2005) claim that KM is moving from the first generation in which knowledge was

managed through the use of technology to the second generation in which knowledge will be managed through action. Knowledge that is managed through action will take into consideration the interaction between individuals within the social setting. In Nerur and Baliyepally's study (as cited in Bjornson & Dingsoyr, 2008) software organizations that use traditional software development methods focus on managing explicit knowledge while software organizations that use agile software development methods focus on managing tacit knowledge.

Bjornson and Dingsoyr (2008) used the KM framework developed by Earl (2001), which classified KM into seven schools to analyze the literature. The seven schools include three technocratic schools: systems, cartographic, and engineering, one economic school, and three behavioral schools: organizational, spatial, and strategic. Bjornson and Dingsoyr found most of the literature on software development and KM focused on the technocratic school and the behavioral school with little focus on the economic school which means that the research focused on the KM processes and tools but not on "creating revenue streams from the exploitation of knowledge and intellectual capital" (Earl, 2001, p. 218).

Bjornson and Dingsoyr (2008) concluded that future research should provide in-depth studies of KM in software organizations, such as ethnographic studies and future research should focus on the schools relevant to agile software development particularly the organizational school, the cartographic school, and the spatial school. This means that organizations may benefit from additional research in "the creation, sharing, and the use of knowledge as a resource" (Earl, 2001, p. 218). Software organizations may benefit

from additional research in how software organizations can provide a knowledge map of the organization by identifying who knows what (Earl, 2001).

Boden, Avram, Bannon, and Wulf (2009) discussed two case studies that illustrated how cultural and social issues affect knowledge sharing in software development. Boden et al. proposed that traditional software development projects can use a technocratic or behavioral approach to knowledge management; but agile software development projects require a second-generation approach to KM because agile software development processes focus on social interaction and customer collaboration rather than on documentation and codification. Boden et al. found that there was less conflict and more knowledge sharing when social capital was high and interpersonal relationships were formed between individuals on the geographically dispersed teams.

Slaughter and Kirsch (2006) found that performance improvement increased when knowledge transfer between individuals was frequent and directions were not used or when knowledge transfer was infrequent and directions were made available to the individuals to support their performance. Knowledge was transferred more frequently when the team members were in close proximity, when they were in a hierarchical relationship, or when they worked in different units of an organization. Directions were used more frequently when the team members were not in close proximity, when they were in a hierarchical relationship, or when they worked in different units of an organization.

Traditional software architecture and KM. Abdullah, Shah, and Talib (2011a) designed a KM architecture and a prototype tool used to create, maintain, share, and

distribute knowledge during the software architecture development process. The KM architecture was based on the architecture tradeoff analysis method (ATAM), which consists of four phases: “presentation, investigation and analysis, testing, and reporting” (Abdullah et al., 2011a, p. 4). Research participants completed a survey after the KM requirements were defined and after the KM tool prototype was developed to determine how well the prototype met the research participant’s expectations. Although 80% of the research participants accepted the KM tool, Abdullah et al. (2011a) did not describe the survey population and additional research is needed to determine if the findings apply to software projects using agile software methods.

Traditional software requirement engineering and KM. Jangping, Qingjing, Dejie, and Hongbo (2010) and Jiangping, Hui, Dan and Deyi (2010) focused on how traditional software development teams could benefit from KM to develop software requirements. Jangping et al. proposed a KM model to improve knowledge transfer during the software requirement development process. The results were based on the responses from one hundred and six staff members from the Guang-dong Software Organization to a 46-question survey. Jangping et al. found that there was a negative relationship between knowledge transfer and the ambiguity of the knowledge and there was a negative relationship between knowledge transfer and the systemization of the knowledge. There was a positive relationship between knowledge transfer and trust, technical support, incentives, willingness to transfer knowledge, capacity for absorption and capacity of knowledge impartation. Jangping et al. controlled for the research participants’ number of years of experience, position, and qualifications.

Jangping et al. (2010) proposed a model for knowledge creation during the software requirement development process based on a review of the literature and a case study of a New York organization. Jangping et al. found that knowledge creation during the software requirements process benefited from a diverse project team. Subject matter experts provided valuable information to the knowledge creation process and effective project management and methodology contributed to the success of the knowledge creation process during the software requirement development process at the New York organization. Additional research is needed on KM in an agile software environment to determine if knowledge transfer and knowledge creation are affected by the same factors as Jangping et al. and Jiangping (2010) found in a traditional software requirement environment.

Traditional software estimation and KM. Software organizations need to prepare and collect software data to estimate software *size, effort, cost, and schedule*; however, existing software estimating tools are inadequate for estimating 4GL software development projects because existing software estimating tools did not adequately account for software complexity or interaction of 4GL applications (Patil & Nageswara Yogi, 2011). A 4GL software-estimating tool was developed and validated by Patil and Nageswara Yogi (2011) and they concluded that their software-estimating tool more accurately estimated the software effort than the existing software estimating tools. If agile methodologies are used, then as claimed by Patil and Nageswara Yogi software teams need to collect data to improve software development estimation.

Traditional software development and knowledge codification. Sholla and Nazari (2011) interviewed software managers, software developers, and project managers at four medium-sized organizations to identify KM codification success criteria. KM codification was defined as the process of making tacit knowledge explicit and KM was defined as active knowledge that is shared via an intranet. Sholla and Nazari identified four success criteria software organizations need to successfully implement intranet enabled KM codification strategies. Software organizations need to create a knowledge sharing culture, software organizations need to maintain a consistent focus on KM, and software organizations need to update the KM tools, as the organizational strategy and processes change, and software organizations need to align the KM strategy with the business goals.

Although Sholla and Nazari (2011) mentioned Smith and Farquar's (2000) study of KM from an AI perspective, Sholla and Nazari did not explore artificial intelligence (AI) or the use of analytics in software development organizations. A variety of KM tools were used to varying degrees within each organization that participated in the research study. Organizations may benefit by implementing KM tools that focus on skill management and people to minimize entry cost and increase visibility to KM (Sholla and Nazari). Organizations could also benefit by tailoring KM tools to provide the knowledge needed by team members other than management.

Agile software development and KM. Based on a review of the literature on agile software development and knowledge management, Chan and Thong (2010) found that little work had been done to determine the relationship between the use of agile

software development practices and KM. Chan and Thong gathered data from 288 agile software developers and based on the data collected, Chan and Thong concluded that there was a positive relationship between three agile software development practices and KM. The three agile software practices of pair programming, collective ownership, and coding standards positively affected the outcome of knowledge creation, knowledge retention, and knowledge transfer in an agile software environment.

Ceschi et al. (2005) claimed that software project failure was due to issues related to people and project management rather than technology. Software development teams were better able to reduce the risk of project failure by using agile software development methods rather than traditional software development methods (Ceschi et al., 2005). The agile software development teams were better able to deliver the required functionality on time and improve productivity by using more effective communication methods and knowledge transfer methods than traditional software development teams.

Based on a review of the literature on agile software development and KM from 2001-2011, Neves et al. (2011) found that agile teams created knowledge by developing working software, by responding to change, and by interacting and collaborating with customers and team members. Although several advantages and opportunities to using agile software development methods were identified, several weaknesses and threats to using agile software methods were also identified. Productivity may be improved by using agile software methods because the goal of the iterative process is to frequently deliver working software; however, productivity may be negatively affected by the need for more experienced team members to take the time to train less expert team members.

Although conflicts may occur within agile software development teams, agile software teams have higher job satisfaction and are more motivated than traditional software teams (Tessem & Maurer, 2007). Although tacit knowledge is created through interaction between individuals, interaction may be difficult to facilitate on large software projects or on projects where team members are not co-located (Ryan & O'Connor, 2009). Investments in architecture are required to enable agile software methods to work on large software projects (Boehm et al., 2010).

Agile software development methods minimize the RE documentation developed which may reduce the maintainability of the software products delivered (AlAli & Issa, 2011; Rubin & Rubin, 2011). Although the agile manifesto encouraged “maximizing the amount of work not done” (Beck et al., Twelve Principles, 2001), code that cannot be maintained may increase the overall system cost. AlAli and Issa (2011) proposed developing reusable use cases to increase the documentation developed during each software cycle or iteration while reducing the level of documentation effort required.

Rubin and Rubin (2011) proposed embedding knowledge gained from traditional RE into agile software to improve the documentation needed for software maintenance. The proposed solution combined knowledge gained from data modeling, behavior modeling, enterprise modeling, and domain modeling while eliminating the overlap in the various modeling approaches. The solution was a set of classes that model “actors, roles, resources, services, goals, constraints, transitions, and states” (Rubin & Rubin, 2011, p. 125). The use of a Wiki may improve learning across agile software teams and enable less experienced engineers to work independently (Amescua et al., 2010).

Levy and Hazzan (2009) claimed that knowledge management implementation efforts encountered the same barriers as agile software development implementations. Levy and Hazzan compared nine arguments that arise when agile software development processes are introduced in an organization to nine arguments that arise when knowledge management processes are introduced in an organization. Although agile software project managers understand the importance of KM in agile software development projects, Levy and Hazzan claimed that agile software project managers should know how to apply knowledge management in agile software development implementations.

Levy and Hazzan (2009) recommended six KM activities that could be integrated into agile software development processes:

1. Assign one team member to the role of knowledge manager.
2. Make KM a topic at a retrospective meeting.
3. Make KM a topic and at planning meetings.
4. Use the project board to assess the value of new knowledge.
5. Include KM metrics with the agile software project metrics.
6. Adapt the KM activities along with the agile software continuous improvement efforts.

Mishra and Mishra (2011) reviewed the literature from 2000-2011 on global software development (GSD). GSD projects present unique challenges for software management due to the geographic dispersion of software teams. Mishra and Mishra found that most of the research had been done on “project management, process

management, knowledge management and requirements management areas while configuration, risk, and quality management issues” (p. 48) received limited attention.

Although only a few of the articles reviewed by Mishra and Mishra (2011) discussed agile software methods from a GSD perspective, Mudumba and Lee (as cited in Mishra & Mishra, 2011) found that agile methods reduced risk in GSD projects. Mishra and Mishra also stated that KM was found to be a critical component of successful GSD. However, Mishra and Mishra found that more than one of the articles reviewed, recommended additional research be done to determine how to manage knowledge from a variety of sources and formats.

Qumer and Hendersen-Sellers (2008) developed the agile adoption and improvement model (AAIM) and the agile software solutions framework (ASSF), which includes the Agile Toolkit. The AAIM was developed to enable managers to determine which agile practices to implement at each stage of agile maturity. Managers select agile practices from one of six agile stages in the AAIM, which are associated with one of three AAIM maturity blocks. Managers select agile practices from the prompt block when the agile transition is initiated. Managers select agile practices from the crux block when the software organization is ready to implement the core agile practices and managers select agile practices from the apex block when the organization is ready to focus on quality and learning.

The ASSF was developed to provide a comprehensive framework for agile implementation, which, in addition to people, process, and tools, included knowledge, governance, the Agile Toolkit, and alignment with the business (Qumer & Hendersen-

Sellers, 2008). The Agile Toolkit was a KMS that was intended to assist managers in their selection of the appropriate agile practices. Although Qumer and Hendersen-Sellers (2008) claimed that agile methods could be used in large and small software projects, Pikkarainen et al. (2008) argued that agile methods do not provide the communication required to support complex development or larger decentralized software development.

Current Research in Software Methods and Analytics

Although research has been published on the use of analytics to improve software development including RE, software testing, and software estimating, little research has been published on the use of analytics to improve software development. Traditional software methods were used to explore the use of analytics to determine which functionality to include in an electronic game, to test software, and to estimate the software development schedule. Agile software methods were used to explore the use of analytics to estimate the software completion date and a DSS was developed to aid in the selection of prioritizing requirements.

Traditional software development and analytics. Based on a research study in electronic gaming, analytics were used to better understand user behavior (Hullett et al., 2011). Descriptive analytics were used to analyze the usage patterns of game players. The data revealed that some of the game content was underused. The decision was made to remove 20% of the content in future releases and to provide feedback to the user, which would improve their gaming experience. Hullett et al. (2011) argued that the research in gaming applied to software development in general.

Siwen and Jun (2010) developed a software-testing tool using multi-agents to extract data from unified modeling language (UML) and to develop test cases. Although UML has successfully been used to develop test cases, the test cases could not be extended. The multi-agent tool enabled the software testers to develop rules that enabled the multi-agent tool to extend the test cases. Siwen and Jun concluded that their multi-agent tool was feasible based on applying the tool to an aviation software project. Additional research is needed to determine the feasibility of using the multi-agent tool in an agile software environment.

Software projects that use traditional software methods rely on schedules that are developed at the beginning of the project and are dependent upon uncertain data. Zare and Akhavan (2009) developed a fuzzy logic algorithm to account for pessimistic estimates, most likely estimates, and optimistic estimates. The fuzzy logic algorithm also accounted for the probability that there would be loops in the schedule when software developers repeated activities. Zare and Akhavan found that the fuzzy algorithm was more accurate than the schedule, based on the critical path method (CPM), when the scheduling methods were applied to the same software project in Iran.

Agile software development and analytics. A Bayesian network was used to model an agile software project that used the XP method (Abouelela & Benedicenti, 2010). The model was used to estimate the completion date and the defect rate for each software release. Abouelela and Benedicenti (2010) claimed that the model accurately predicted the completion date for each software release based on the results of two case studies.

Current Research in Software Methods, KM, and Analytics

Some research was found in the literature that discussed the potential use of analytics and KM to improve software development in a traditional software development environment. Abdullah, Talib, and Misran (2011b) discussed how an agent based KMS could improve software defect management and Jiang et al. (2008) developed a DSS that used case-based reasoning to select RE techniques. Abdullah et al. (2011b) developed an agent based KMS to improve software defect knowledge sharing. The KMS was based on the personal software process (PSP) and the team software development process (TSP) framework “of forms, guidelines, and procedures” (Abdullah et al., 2011b, p. 347) to develop the agent based KMS. The agent based KMS used four agents: a profiling agent, a notification agent, a reminder agent, and a scheduling agent.

Twelve officers at the Malaysian Qualification Agency (MQA) information technology department completed a preliminary survey on software defect management processes and the knowledge needed to manage defects. After the agent based KMS was developed, the 12 officers completed a final survey. Based on the final survey results, Abdullah et al. (2011b) reported that the agent based KMS correctly categorized the software defects, and the notification, reminder, and scheduling agents were effective; however, 10% of the survey respondents rated the accuracy of the agent based KMS as poor.

The effectiveness of the agent based KMS developed by Abdullah et al. (2011b) was not evaluated within the context of a software development project. The agent-based KMS was not designed for use in an agile software development environment.

Additional research could determine the effectiveness of the agent based KMS developed by Abdullah et al. (2011b) within a traditional software development environment and additional research could determine how to develop an agent based KMS for use within an agile software development environment.

Jiang et al.(2008) argued that software teams do not have adequate knowledge of all of the available RE techniques and the strengths and weaknesses of each technique when they are selecting RE techniques for software projects. Three case studies were used to evaluate the use of a prototype DSS to select RE techniques to use for a software project. Case based reasoning (CBR), frame-based-reasoning, and relational reasoning was used to develop the DSS.

Although Jiang et al. (2008) found that the prototype DSS did improve the understandability of the requirements and fewer requirement changes were needed, Jiang et al. stated that additional research was needed to generalize the findings beyond the case studies included in their research. The prototype DSS included 46 RE techniques; however, additional techniques may be added in the future. Additional rules may also be added to identify additional situations in which each technique would work well and situations in which each technique would not work well. Additional rules may also be added to identify potential cost reductions and user-defined rules, which define constraints, based on the project characteristics.

Research Methods in the Current Literature

The research methods used in the current literature are discussed in this section of the research proposal. The research methods used to study analytics are discussed

followed by a discussion of the research methods used to study software development in traditional and agile software development environments. The research methods used to study KM and KM in traditional and agile software environments are discussed. This section of the research proposal concludes with a discussion of the research methods used to study analytics in traditional and agile software environments followed by a discussion of the research methods used to study the use of analytics and KM in a traditional software environment.

Analytics research methods. Based on a review of the current literature, two authors used qualitative research methods while the remaining authors used quantitative research methods to study analytics. Qualitative research methods were used to answer questions, such as, what framework can be used to discover BI metrics (Ferrand et al., 2010) and what are BI CSFs (Yeoh & Koronios, 2010)? Quantitative research methods were used to answer questions, such as, does DDD improve organizational productivity and profitability (Brynjolfsson et al., 2011), and how do CTOs consider, weigh, and integrate data (Ow & Morris, 2010)?

Gartner's proprietary research methods were used to answer questions, such as, what are the capabilities of IBMs Watson (Adrian & Genovese, 2011), what are the trends in BI (Gassman et al., 2011), and how will analytic applications evolve over time (Herschel, 2011)? Chandler (2011) discussed how analytics would be used to improve performance management and Schlegel et al. (2009) discussed how CDM would increase in use for non-routine complex decisions. Several authors conducted literature reviews, book reviews, document analyses, or system analyses to answer questions, such as, what

is the state of BI in Romania (Ivancenco et al., 2010), and what are the trends in HR analytics (Bassi, 2011)?

Traditional software development research methods. Emam and Koru (2008) improved upon the research on software project success conducted by the Standish Group by describing their research methods, which included quantitative research methods. The claim that the software project failure rate had decreased since 2008 and software development productivity had not kept pace with the advancements in hardware was based on a review of the literature (Fitzgerald, 2012). Although software development productivity improved, Fitzgerald (2012) argued that software development productivity has not kept pace with hardware improvements that will enable the number of hardware devices connected to the Internet to increase from 35 billion in 2010 to over 100 billion by 2020.

Quantitative research methods were used to determine that productivity increased when traditional software teams used 4G languages, as the development platform complexity increased, and as team size increased (Rodger et al., 2011). Quantitative research methods were also used to determine if the order in which people, process, and tools were implemented affected software project success (Wadhwa & Mitra, n.d.). Although the order in which people, process, and tools were implemented did not affect software project success, software project success was affected by how closely people, process, and tools were aligned with the strategic goals of the organization.

Dubey (2010), Hewagamage and Hewagamage (2011), and Sudhakar et al. (2010) published technical papers on traditional software methods. CASE tools can be used to

improve software development productivity; consequently, Dubey categorized CASE tools into 18 categories so that managers could make more informed decisions about the use of CASE tools. Dubey argued that additional CASE tools are needed to automate each phase of the software development process.

Hewagamage and Hewagamage (2011) claimed that the terminology used in software engineering was inconsistent based on a review of the literature on CMMI, PMBOK, PRINCE2, ITIL, and MSF. Software project success may improve if software teams used common terminology and a common framework that defines the relationship between the terms. Hewagamage and Hewagamage proposed a common software development framework; however, additional research is needed to determine the feasibility of their hypothesis that their proposed software development framework could serve as a common framework and that their common framework would improve software project success.

One question that needs to be answered when discussing software project success is, how is success measured? Sudhakar et al. (2010) attempted to answer that question by reviewing the literature to determine how software development productivity had been defined. Sudhakar et al. found that lines-of-code (LOC) was the most commonly used measure of productivity, although more than 10 definitions of productivity were found in the literature. Productivity can be improved by reducing interruptions and by improving the quality of the software produced and software developers can use historical data to improve software estimation and to reduce defects (Elminir et al., 2009).

Agile software development research methods. Clutterbuck et al. (2009), Ganesh and Thangasamy (2012), Layman et al. (2006), McAvoy and Butler (2009), Omar et al. (2011), Qumer and Henderson-Sellers (2008), Ramesh et al. (2010), Sharp et al. (2009), and Zhang and Patel (n.d.) used qualitative research methods to study agile software methods. Clutterbuck et al. (2009) observed how the individuals on a software team consisting of seven members tailored the agile Scrum and XP methods to develop a web application. Although key information was shared between all stakeholders when agile methods were used, the benefits of using agile methods were dependent upon the skills and experience of the software team members.

Zhang and Patel (n.d.) described how MDD was combined with agile methods to improve software development productivity in a telecommunications project. McAvoy and Butler (2009) explored negative influences on decision making in agile software environments, Ramesh et al. (2010) explored the risks and rewards of agile software practices during the requirement engineering process. Agile software teams must adjust to many process and cultural changes as they transition from traditional software methods to agile methods (Ganesh & Thangasamy, 2012). By remaining flexible, four software teams were able to overcome some of the difficulties they encountered when they transitioned from traditional software methods to agile software methods (Omar et al., 2011).

Qualitative research methods were used to determine if the software quality and productivity were better than industry averages when agile software methods were used (Layman et al., 2006). Although Layman et al. found that quality and productivity were

better than industry averages; Layman et al. discussed how “availability of data, tool support, cooperative personnel, and project status” (p. 10) influences the outcome of case studies. Consequently, reliability and validity can be improved if researchers account for these factors when conducting case studies.

Qumer and Hendersen-Sellers (2008) used qualitative research methods to test the feasibility of the agile model and framework they developed to assist organizations that are transitioning from traditional software methods to agile methods. Based on the results of two case studies, Qumer and Hendersen-Sellers concluded that the AAIM and the ASSF, which included a KMS, were effective in assisting managers to gradually introduce agile software practices. Although Rao et al. (2011) claimed that agile methods are effective in small organizations; Qumer and Hendersen-Sellers argued that large organizations might successfully transition to agile methods by using the AAIM and ASSF, which enables management to gradually introduce agile practices over time.

Sharp et al. (2009) conducted a qualitative research study of the use of two physical artifacts in agile software development, the story cards and the wall. Story cards are used to document requirements when the agile Scrum method is used. The story cards are placed on a wall, which is used to communicate progress. The story cards and the wall serve a social and a notational purpose; therefore, teams who are considering the use of automated story cards and the wall need to consider the potential negative social effect of limiting or reducing face to face communication.

Ballou (2008), Balijepally et al. (2009), Pelrine (2011), and Salo and Abrahamsson (2008) used quantitative research methods to study agile software methods.

Ballou discussed a research study conducted by QSM Associates on behalf of one of the agile tool vendor companies. QSMA compared the software project results from 29 agile software projects to the results from 7,500 traditional software projects. The agile software projects were delivered 37% faster than the average traditional software project and the agile project teams were 16% more productive than the average traditional software team.

Salo and Abrahamsson (2008) used quantitative research methods and surveyed team members from 35 projects in 13 organizations in eight European countries on the use of agile XP methods and Scrum methods. Although Salo and Abrahamsson found that XP was used more than Scrum, the research study did not explain why the software organizations used XP methods more than Scrum methods. The least used XP practices were TDD, pair programming, shared code ownership, and on-site customer. The most commonly used Scrum practice was the requirement backlog; however, the research study did not explain why the European software organizations chose to use some agile practices more than other agile practices.

Based on the results of a laboratory experiment, Balijepally et al. (2009) concluded that software developers who use agile XP practice of pair programming did not outperform software developers who did not use pair programming; however, the software developers who used pair programming were more satisfied and more confident than the software developers who do not use pair programming. Although Pelrine (2011) found that 79% of the agile software development tasks were complicated, complex, chaotic, Balijepally et al. found no difference between the performance of the software

developers who used pair programming and those who did not use pair programming based on task complexity.

Eccles et al. (2010), Lalsing et al. (2008), Lee and Xia (2010), Rao et al. (2011), and Smith (2011) used qualitative and quantitative research methods to study agile software practices. Smith claimed software development and operations should work together to focus on the business outcomes rather than on process compliance. Eccles et al. found that software team productivity can be improved by locating agile software development teams in the same location although collocated teams may experience more interruptions. Rao et al. (2011) compared the benefits of three agile software development methods, XP, DCDM, and Scrum to traditional software methods and Lee and Xia studied the effect of software team response extensiveness, software team response efficiency, software team autonomy, and software team diversity on software project on-time completion, on-budget completion, and software functionality. Based on an analysis of the project budgets, schedules, and defects, and based on observation of three agile software teams of different sizes, Lalsing et al. claimed that agile methods work best for smaller software teams.

Glazer et al. (2011), Ionel (2009), Khan et al. (2011), Nerur and Baliyepally (2007), Rinko-Gay (2009), Shull et al. (2010), and Sutherland et al. (2007) published technical papers on agile software methods. Glazer et al. argued that software development is dependent upon people, process, and technology. Consequently, Glazer et al. argued that software projects could benefit from both agile software methods, which

focus on people and SEI CMMI methods which focus on process to improve software quality and productivity.

Based on a review of the literature from 1998-2009 on agile software methods, Ionel (2009) stated that additional empirical research was needed to determine how well agile methods improved software quality and productivity. The research on agile software methods primarily consisted of case studies and anecdotal evidence. Khan et al. (2011) argued that although agile methods were developed to improve productivity, additional research is needed to systematically develop methods and technologies that are scalable and incorporate processes that are understandable.

KM research methods. Mansour et al. (2011) and Molaei (2010) published technical papers on KM. KM is needed to enable organizations to innovate, compete, and improve productivity (Mansour et al., 2011). Organizations use knowledge as an input to production processes, to control production processes, to process knowledge, and to design processes. Because knowledge is a critical component of organizational success, Mansour et al. developed a general KM framework that consolidated 16 KM processes described in the literature and because small organizations need to find ways to share knowledge with similar organizations, Molaei developed a KM model. However, additional research is needed to validate the effectiveness of the proposed KM framework developed by Mansour et al. and the KM model developed by Molaei.

Traditional software development and KM research methods. Quantitative research methods were used to determine that trust improved knowledge transfer in a Software Process Improvement (SPI) team (Jangping et al., 2010), and to determine how

KM should be used to manage knowledge during the software architecture development process. Qualitative research methods were used to evaluate the use of Churchman's inquiry systems to develop a KMS (Linden, 2011) and to determine the effectiveness of knowledge transfer methods in a traditional software development environment (Slaughter & Kirsch, 2006). Two authors used both qualitative and quantitative research methods to study software and KM to answer questions, such as, what influences knowledge creation in the software requirements process (Jiangping et al., 2010), what success criteria can software organizations use for KM codification initiatives (Sholla & Nazari, 2010).

Agile software development and KM research methods. Quantitative and qualitative research methods were used to determine how to measure team knowledge sharing in an agile software development environment (Ryan & O'Connor, 2009). The Team Tacit Knowledge Measure (TTKM) was developed and validated; however, Ryan and O'Connor (2009) found that although the TTKM could be used to measure effectiveness, the TTKM could not be used to measure team efficiency. Team effectiveness measured how well the team interacted and met the project goals and objectives and efficiency measured how well the team adhered to the project budget and schedule.

Boehm et al. (2010) also use both qualitative and quantitative research methods to study KM and agile methods. Boehm et al. claimed that approximately 5% of all software projects were large. Large software projects have over 25 team members and if agile software methods are used on large software projects, project success requires that

both architecture and agility be adequately addressed. The appropriate mix of agile and architected methods is dependent upon “the system’s size, criticality, and requirements volatility” (Boehm et al., 2010, p. 3).

Quantitative research methods were used to develop the Incremental Commitment Model (ICM), which Boehm et al. (2010) developed to enable managers to determine the appropriate mix of agile and architected methods to use for large software projects. Based on the results of five case studies, the ICM enabled managers to select the appropriate mix of architected and agile practices to use. Boehm et al. stated that the criteria for selecting the appropriate mix of agile and architected practices will continue to evolve and additional research will be needed to mature the ICM.

Based on a qualitative research study, which compared communication between two software teams that used agile XP and Scrum practices, such as, daily meetings and open office space, agile practices improved internal and external communication (Pikkarainen et al., 2008). The knowledge sharing and transfer methods used to develop requirements were insufficient when the number of requirements was large. Pikkarainen et al. recommended that traditional methods might be needed to manage knowledge on larger software projects.

Qualitative research methods were also used to describe the cultural influences that affect KM in global software engineering environments including agile software development environments (Boden et al., 2009). Rubin and Rubin (2011) used qualitative research methods to validate their proposed agile documentation process and Tessem and Maurer (2007) used qualitative research methods to determine if agile

methods increased job satisfaction and motivation. Although agile methods favor working software over documentation, Rubin and Rubin argued that the lack of documentation might increase the dependency on collaboration between stakeholders and increase software maintenance complexity. Although Rubin and Rubin described how documentation could be developed on an agile software project, additional research is needed to determine the feasibility and generalizability of the proposed agile documentation methods. Additional research is also needed to determine if the findings that agile methods increase job satisfaction and motivation are generalizable beyond the case study conducted by Tessem and Maurer.

Quantitative research methods were used to determine the feasibility of reusable use cases in agile software development and although a catalogue of use cases was developed, AlAli and Issa (2011) did not refer to the use case repository as a KMS. Based on six case studies, AlAli and Issa concluded that documentation can be developed when agile software methods are used. The use case catalogue saved time and improved the completeness of the documentation.

Knowledge transfer and sharing was improved when agile software developers had access to the agile processes on a Wiki during the software development process (Amescua et al., 2010). Quantitative research methods were used to test the hypothesis that a KMS would improve learning in an agile software environment. The Wiki-based KMS enabled the junior engineers to work independently and agile software engineers to learn the agile software development process without formal training.

Quantitative research methods were also used to determine if agile software methods reduced the risk of software project failure. A quantitative research study was conducted to answer the question, do the agile practices of pair programming, collective ownership, and coding standards positively affect the KM outcomes of knowledge creation, knowledge retention, and knowledge transfer (Chan & Thong, 2010)? Ceschi et al. (2005) compared the survey results of 20 agile software managers to the survey results of 20 traditional software managers to determine if agile methods improved project management practices. The communication practices in agile methods improved knowledge transfer, which reduced the risk of project failure.

Bjornson and Dingsoyr (2008) and Mishra and Mishra (2011) reviewed the literature on agile software and KM to make recommendations for future research on agile software engineering and KM. Levy and Hazzan (2009) proposed a definition for agile KM based on a review of the literature. Neves et al. (2011) conducted a systematic literature review to determine how agile software teams were affected by knowledge creation and sharing.

Traditional software development and analytics research methods. Hullett et al. (2011) and Siwen and Jun (2010) used qualitative research methods to study the use of analytics in traditional software development environments. Hullett et al. used descriptive analytics to track user interaction with an electronic game. Siwen and Jun used prescriptive analytics to generate test data from UML. Zare and Akhavan (2009) used quantitative research methods to study the use of prescriptive analytics to improve the accuracy of software scheduling.

Analytics and KM research methods. Smith and Farquar (2000) proposed in their qualitative research study on KM that analytics could be used to improve KM and Lingling et al. (2009) proposed in their technical paper that data mined from large data bases needed to be transformed to become actionable knowledge. Additional research is needed to determine how analytics could be used to improve KM in an agile software environment and to determine how to use analytics to improve software development productivity in an agile software environment.

Agile software development and analytics research methods. Abouelela and Benedicenti (2010) used qualitative research methods to study the use of analytics to improve agile software methods. A Bayesian network was used to predict the software defect rate of projects using agile XP methods. Although this study demonstrates the potential use of analytics to improve software development in agile environments, additional research is needed to generalize the findings of this study beyond the cases under study and to measure the impact on software development productivity.

Traditional software development, analytics, and KM research methods. Abdullah et al. (2011b) conducted a research study on analytics, in the form of intelligent agents, and KM using both qualitative and quantitative research methods to manage defects in a traditional software development environment and Jiang et al. (2008) used qualitative research methods to validate the DSS they developed to enable managers to select RE techniques for “requirements elicitation, requirements analysis & negotiation, requirements documentation, and requirements validation” (p. 118). The DSS used case-based reasoning to prescribe the appropriate RE techniques. Although Jiang et al. found

that the DSS was affective when it was applied in one case study, Jiang et al. stated that the RE technique DSS was a prototype and additional research was needed to validate future enhancements to the RE technique DSS.

Research Methods for Research

Qualitative research methods were used for the research study on management's understanding of DDD as a tool to improve software development productivity. Although both qualitative and quantitative research methods were used to study traditional software development, agile software development, traditional software development and KM, and agile software development and KM, quantitative research methods were predominantly used to study analytics. Few research studies were found on the use of analytics on software development or on the use of analytics and KM on software development. Organizations need to define DDD within the context of the problem (Herschel et al. 2010; Ferrand et al., 2010; Yeoh & Koronios, 2010); therefore, the qualitative research study on software management's understanding of DDD is intended to fill this gap in the literature by exploring the meaning of DDD within an agile software development environment.

Based on a review of the literature, the research on the use of DDD as a tool to improve software development productivity is nascent. Patton (2002) stated that it is appropriate to use qualitative research methods when more needs to be known about a topic. More needs to be known about DDD; consequently, qualitative research methods were used for a research study on software management's understanding of DDD as a tool to improve software development productivity in an agile software environment.

Qualitative research may provide a better understanding of how software managers consider, weigh, and integrate data for decision-making and qualitative research may provide additional information on the meaning of data and analytics to improve software development productivity.

Research Approaches in the Current Literature

Based on a review of the current literature, case study was the predominant qualitative research approach used to study traditional software development, Agile software development, traditional software development and KM, and Agile software development and KM. According to Creswell (2007) the case study approach is used to describe a bounded system that attempts to resolve a problem. Ferrand et al. (2010) and Yeoh and Koronios (2010) used the case study approach to study analytics. Ferrand et al. focused on problems related to healthcare safety in Canadian hospitals and Yeoh and Koronios focused on problems related to five large data warehouse implementations.

One researcher used an ethnographic approach to study the role of artifacts in agile software development (Sharp et al., 2009) while survey was the predominant quantitative research approach used to study analytics. Although few research studies were found on the use of data, analytics, and KM to improve software development productivity, Abdullah et al. (2011b) used both interviews and surveys to study the use of an agent based KMS to reduce software defects in a traditional software environment. Additional research is needed to explore the use of data, analytics, and KM in an agile software environment.

Research Approach for Used for this Research

The qualitative research approach used for the research study on software management's understanding of DDD is a phenomenological approach. Creswell (2007) noted phenomenological research "seeks to understand the meaning of experiences of individuals about this phenomenon" (p. 94). A review of the literature revealed the need for additional research into the meaning of DDD and the related topics of BI, AI, business analytics, data mining, knowledge management, and entity resolution and analysis within the context of the problem (Adrian & Genovese, 2011; Herschel, 2011; Lingling et al., 2009; Yeoh & Koronios, 2010).

According to Patton (2002) there is a difference between phenomenological inquiry and a phenomenological research approach. Phenomenological inquiry is a worldview that is focused on the shared reality of individuals while the phenomenological research approach is a study that describes what individuals experience and how they experience what they experience. Although Smith and Farquar (2000) recommended that AI be used to improve KM and Abdullah et al. (2011b) determined that intelligent agents could improve software defect management in a traditional software development environment, a better understanding of the phenomenon of DDD within an agile software environment is needed.

The research participants were selected based on their familiarity with agile software development methods, their experience as software managers, project managers, and agile coaches, and their interest in participating in the research study. According to Nerur and Baliyepally (2007) agile software development methods are focused on

learning and innovation rather than on optimization and control. Just as Linden et al. (2007) recommended the use of Churchman's inquiry systems to design and develop information systems in complex environments, agile software development methodology was developed to improve productivity in complex environments.

Research Process Used for this Research

The qualitative research process planned for this research is the IPA research process described by Smith et al. (2009). Although Smith et al. described a series of steps; the research process will remain flexible in keeping with their guidance. The data will be analyzed as it is collected to identify clusters of meaning. The IPA process is discussed in more detail in Chapter 3.

Summary and Conclusions

The software project failure rate needs to be reduced (Emam & Koru, 2008) and although the software project failure rate has decreased since 2008, software development productivity has not kept pace with hardware advancements (Fitzgerald, 2012). Agile software methods were developed to reduce the software project failure rate and (Rao et al., 2011) and to improve productivity (Schwaber, 1995). Although agile methods work best for smaller projects, software projects can be broken up into smaller units or combined with other methods like MDD for larger projects (Zhang & Patel, n.d.).

Brynjolfsson et al. (2011) found that DDD improved organizational output and productivity and although DDD was defined as data and business analytics; multiple definitions for analytics were found in the literature. Organizations need to define analytics and the scope of any initiative (Salam & Cearley, 2012). Based on a review of

the literature, the meaning of DDD within an agile software environment has not been defined and few research studies have explored the use of DDD as a tool to improve software development productivity.

Slaughter and Kirsch (2006) found that knowledge transfer improved productivity in a traditional software development environment and although agile methods may increase knowledge creation, productivity may be negatively impacted when more experienced software developers have to take time to train less experienced software developers (Neves et al., 2011). KM may improve productivity when agile software teams work in the same location (Pikkarainen et al., 2008), when agile software teams use a WIKI to share knowledge (Amescua et al., 2010), and when agile software teams use reusable use cases (AlaAli & Issa, 2011). Tessem and Mauer (2007) claimed that Agile methods lead to increased job satisfaction which results in increased productivity and although Abdullah et al. (2011b) found that the use of KM and analytics improved defect management in a traditional software environment, no research was found on the use of KM and analytics in an agile software environment.

Additional research is needed to determine if knowledge creation, accumulation, retention, and transfer may improve decision making in an agile software environment and to determine how the improved decision-making results in improved productivity. The qualitative research study was intended to explore the use of DDD, which includes data, analytics, and KM, as a tool to improve productivity in an agile software environment. A better understanding of the phenomenon of DDD within an agile software environment may enable software teams to brainstorm ways to use DDD as a

tool to improve software development productivity. The research methodology and procedures for the qualitative research study are discussed in Chapter 3.

Chapter 3: Research Method

The problem researched in this dissertation was the limited information about software managers' experiences with DDD in agile software organizations as a tool to improve software development productivity. The purpose for this research was to better understand software manager's attitudes toward the use of DDD as a tool to improve software development productivity, to better understand how DDD is currently used in an agile software environment, and how DDD could be used in an agile software environment to improve software development productivity.

Although software development productivity has improved, software development productivity needs to continue to improve (Emam & Koru, 2008). Global competition and advances in hardware have increased the opportunities and the challenges for software development organizations and the software organizations that can take advantage of hardware advances and bring products to market quickly will be more likely to survive and thrive (Fitzgerald, 2012). According to Brynjolfsson et al. (2011) productivity may be improved if organizations use DDD; however, organizations, including software organizations, need to define DDD within their own context, and explore how they can use DDD to improve productivity (Chandler et al., 2011; Ferrand et al., 2010; Herschel et al., 2010; Rajteric, 2010; Yeoh & Koronios, 2010).

The design of a qualitative research study is described in this chapter as well as the rationale for selecting the qualitative research approach that was used to understand the meaning of DDD in an agile software environment. The role of the researcher is described, the research setting will be described, and the research sampling methods is

described. This chapter also covers the measures used to increase trust between the researcher and the research participants.

Research Design and Rationale

The research design should be based on the research questions rather than on the familiarity of the researcher with a research approach. The research questions along with the central concept of the research study are discussed. The rationale for a phenomenological qualitative study is provided based on a review of the research methods and approaches historically used to answer similar questions.

Research Questions

The problem researched in this dissertation was the limited information about software managers' experiences with DDD in agile software organizations as a tool to improve software development productivity; therefore, qualitative research methods were used, including in depth interviews. The research schedule used to conduct the interviews was derived from these research questions. The following four questions were formulated for this research study:

1. What do software managers in agile software environments think about the use of DDD to improve software development productivity?
2. How do software managers in agile software environments currently use descriptive analytics, diagnostic analytics, prescriptive analytics, and predictive analytics, or knowledge creation, retention, accumulation, and transfer to improve software development productivity?

3. How do software managers in agile software environments think descriptive analytics, diagnostic analytics, prescriptive analytics, and predictive analytics, or knowledge creation, retention, accumulation, and transfer could be used to improve software development productivity?
4. What obstacles do software managers in agile software environments think their organization need to overcome to improve software development productivity?

Central Concept

The central concept of the research study was software development productivity. According to the CMMI (2010), productivity may be improved if organizations define processes, establish process improvement goals, and measure the outcomes.

Organizations need to train people to use procedures and methods that are intended to achieve the process improvement goals and organizations need to provide people with tools and equipment that will enable the people to achieve the desired outcomes to improve productivity.

Agile software development methods, such as Scrum, are software development methods that are intended to improved productivity. Organizations have used DDD as a tool to improve productivity; however, more knowledge about DDD as a tool is needed to improve software development in an agile software environment. This research study was intended to explore the meaning of DDD in an agile software development environment and to identify how DDD can be used to improve software development productivity.

Research Tradition

Quantitative research methods were developed to test hypotheses (Chen, 1998) while qualitative research methods were developed to “identify the (socially constructed) patterns and regularities in the world” (Moses & Knutsen, 2007, p. 192). Quantitative research methods are based on a positivist view of the world while qualitative research methods are based on a non-positivist view of the world. The positivist view of the world assumes that the world is governed by rules and the purpose for research is to discover the rules, patterns, and regularities that make the world work. The non-positivist or constructivist worldview is that reality is subjective and each individual creates his or her own reality.

Moses and Knutsen (2007) claimed that there is a hierarchy of quantitative research methods, which are based on the positivist or naturalist worldview. Experiments are at the top of the hierarchy followed by nonexperimental methods, such as, statistics, comparison, and case study. The purpose for experimental research methods is to explain cause and effect by testing hypotheses. Experimentation requires the researcher to deliberately manipulate the variables; however, it is not always practical, ethical, or desirable to conduct an experiment (Moses & Knutsen, 2007). Nonexperimental methods include statistics and comparative methods. Comparative methods are intended to identify causal relationships while statistics are intended to identify the rules, patterns, and regularities in nature (Sullivan, 2001).

Although Maxwell (2005), Miles and Huberman (1994) and Patton (2002) claimed that a variety of approaches to qualitative research have been described in the

literature, Creswell (2007) described five approaches to qualitative research, which encompass the primary approaches to qualitative research. According to Moses and Knutsen (2007) there is no hierarchy to the approaches to qualitative research. The five approaches to qualitative research described by Creswell include: narrative research, phenomenology, grounded theory, ethnography, and case study.

The researcher should select the narrative approach to qualitative research when the purpose for the research is to describe chronological events, happenings, or the stories of a single individual, such as a biographical account of an individual's lived experiences. The phenomenological approach to qualitative research should be selected when the purpose for the research is to describe the lived experiences of several individuals with a phenomenon. The researcher should select the grounded theory approach to qualitative research when the research is intended to result in a theory. The researcher should select the ethnographic approach to qualitative research when the research is intended to improve the understanding of the research participant's culture and the researcher should select the case study approach to qualitative research when the purpose for the research is to study one or more groups to better understand an issue or a problem (Creswell, 2007).

Rationale

The purpose of this phenomenological study was to explore the lived experiences of software managers' use of DDD in agile software organizations as a tool to improve software development productivity rather than to "precisely state theories and derive testable, quantitative predictions from them" (Sullivan, 2001, p. 20); therefore, qualitative research methods were used for the research study rather than quantitative research

methods or mixed methods. Quantitative research methods may be used when enough information is known about a phenomenon; however, based on a review of the literature, more needs to be known about the phenomenon of DDD as a tool to improve software development productivity in an agile software environment. According to Patton (2002) “qualitative methods facilitate study of issues in depths and detail” (p. 14) and the intent of the research study was to gain an in depth understanding of software management’s perspectives on DDD.

More needs to be known about the use of DDD as a tool to improve decision making in agile software environments before a theory can be generated as to how DDD could be used to improve software development productivity; therefore, the research study did not use the grounded theory approach. The ethnographic approach was not used for the research study because there was no intent to understand the culture of agile software managers or agile software development teams and the case study approach was not used because the focus of the study was to explore potential solutions to improve software development productivity rather than to better understand problems or issues within agile software development teams. The narrative research approach was not used because the purpose for the research study was to understand the lived experiences of several individuals rather than to describe the history of a single individual.

The phenomenological approach to qualitative research was used for the research study. The purpose for the research was to describe the lived experiences of several software managers, project managers, and agile coaches with the phenomenon of DDD as a tool to improve software development productivity. An assumption was that software

managers understand their own experiences with DDD and it was anticipated that the meaning of DDD in the software environment would emerge by exploring software management understanding of DDD.

Role of the Researcher

According to Patton (2002) the role of the researcher in qualitative research affects the outcome. When the researcher acts as a participant observer the data collected is affected by the researcher's point of view and when the research acts as an onlooker observer, the data collected is affected by the act of the researcher observing. Although the degree to which a qualitative researcher participates in the research may vary, the qualitative research must balance observation with reflection and involvement with detachment to ensure that the effect on the data collected is managed along with the data that is collected (Patton, 2002).

Researcher Role

As the principle researcher, my role in the qualitative research study was predominantly as an observer. However, as an observer, I needed to balance observation with reflection and involvement with detachment as recommended by Patton (2002). I needed to play a dual role as observer of the research participants and observer of the research participants who are observing the phenomenon of DDD as a tool to improve software development productivity (Smith et al., 2010).

Relationships

Through my efforts to become a Certified Scrum Master (CSM), through my attendance at local Scrum gatherings, and through my networking efforts, I have formed

professional relationships with agile software managers, agile project managers, and agile coaches. The research participants were selected from the agile software development community based on their willingness to participate in this research study and on the demographic information they provided. The research participants were provided with information about the nature of the study to be done and I communicated with the potential research participants to obtain their agreement to participate in the research study.

Management of Bias / Relationships

Bias was managed by ensuring that each research participant was aware that participation in the research study was voluntary and that they may opt out of the research study at any time. Although I attend the Scrum gatherings, my role has been limited to that of an attendee and not as a presenter in order to avoid researcher bias. I am also not employed by any of the companies represented by the Scrum community.

Other Ethical Issues

Although the research participants were given a \$10 gift card as a thank you for their participation in the research study, their participation was voluntary. The research participants were asked to sign an informed consent form, as shown in Appendix C if they agreed to participate in the research. The purpose for the research was explained to the research participants and they were assured that their participation and responses would remain confidential.

Methodology

The research methodology used to explore the meaning of the phenomenon of DDD as a tool to improve software development productivity in an agile software environment is discussed in this section of the proposal. The strategy used to select the research participants is discussed, as well the procedures for recruiting the research participants. The data collection instrumentation and the data analysis plan are also discussed.

Participant Selection Logic

A description of the research participants and the process used to select the research participants is described in this section of the proposal. The population from which the research participants were selected is described as well as the selection strategy and the criterion used to select the research participants. The relationship between the researcher and the research participants is discussed as well as the number of research participants.

Population. The population for this research study was the agile software development community. The goal was to understand their lived experiences. The research participants were selected based on their use and understanding of Scrum software development methods.

Software teams who use Scrum software development methods all share a common interest in the use of agile software development methods, which focus on providing software that improves customer value. Agile software development methods encourage decision making by the people who know the most about the situation, face-to-

face communication to improve knowledge sharing, and continuous improvement. The agile software development community consists of software managers, project managers, agile coaches, business analysts, and software developers who are employed by large and small companies.

Sampling Strategy. There are three strategies for determining a sample size: probability sampling, convenience sampling, and purposeful selection (Maxwell, 2005). Probability sampling and convenience sampling are primarily associated with quantitative research while purposeful selection is primarily associated with qualitative research. The purposeful snowball sampling strategy is used in qualitative research because the research participants are expected to have the background and experience to inform the research study (Creswell, 2007). A purposeful snowball or chain sampling strategy was used to determine which members of the agile software development community to interview.

Selection Criteria. The research participants were selected based on their familiarity with agile software development methods, their experience as agile software managers, agile project managers, and agile coaches, and their interest in participating in the research study. Initial contact was made with an agile software manager, an agile project manager, and a Scrum coach through their LinkedIn.com association. I included an invitation to participate in this research study (see Appendix D) along with a letter of informed consent (see Appendix C). The research participants were asked to recommend one or two other members of the agile software development community who meet the selection criteria. I contacted the potential research participants and in addition to

determining if they were willing to participate in the research, I asked the potential candidates to recommend potential research participants based on the selection criteria.

How Participants are Known. I have been able to establish professional relationships within the agile software development community through my efforts to become a CSM, through my networking efforts, and through my attendance at local Scrum gatherings. I trusted that the members of the agile software development community would recommend research participants who meet the selection criteria. I was also able to determine the qualifications of the research participants by collecting demographic data that includes their current role, the number of years of experience with agile methods, and the size of the projects they manage.

Number of Participants / Cases and Rationale. Sample size refers to the number of participants, events, processes and locations in a research study (Maxwell, 2005). The sample size and the process used to select the research participants can affect the reliability, validity, and generalizability of the research findings. The sample size for the research study is based on the IPA, which recommended that between 4 and 10 hours of interviews be conducted for a Ph.D. study and that selecting participants from different user groups would enable the researcher to explore the phenomenon under study from multiple perspectives (Smith et al., 2009). The purpose for the qualitative research study is to better understand the phenomenon of DDD in an agile software environment; therefore, three agile software managers, three agile project managers, and three agile coaches were interviewed. This multiperspectival approach improved the reliability and validity of the research. The total number of interview hours was approximately 12 hours

and the different perspectives “provide a more detailed and multifaceted account of the phenomenon” (Smith, 2009, p. 52). This number of participants is consistent with Moustakas (1994), who considered a range of five to 25 participants acceptable in phenomenological studies.

Identification, Contact, and Recruitment. Initially, informational interviews were conducted with members of the agile software development community to identify three participants for a pilot study. An email, as shown in Appendix D, along with the informed consent form, as shown in Appendix C, was sent through their LinkedIn.com association to three potential contacts with a request to participate in a pilot study. The invitation to participate in the research states that participation is voluntary and that the research participant may opt out at any time.

The participants in the pilot study were asked to recommend other potential research participants. The research participants were provided with the purpose for the research study and the criteria for selecting research participants. The research participants were incentivized by the promise that the research results will be shared with the agile software development community. The research participants were further incentivized, as they received a \$10 gift card when the interview process concluded to thank them for their participation.

Relationship between Saturation and Sample Size. The purpose for phenomenological qualitative research is to provide a deep understanding of the phenomenon under study. For that reason, the sample size is typically small. Successful analysis “requires time, reflection, and dialogue, and larger datasets tend to inhibit all of

these things” (Smith et al., 2010, p. 52). The number of research participants was limited to three software managers, three project managers, and three agile coaches with the expectation that in depth interviews of one to two hours in duration would yield sufficient data from different perspectives without providing redundant data or data that is difficult to analyze.

Instrumentation

According to Smith et al. (2009) the qualitative researcher should develop a research schedule which outlines the open ended questions that will be asked during the interview in the order in which they will be asked. The purpose for the research schedule is to facilitate the communication between the researcher and the research participant. The research schedule is a tool the researcher uses during the interview process from which the researcher may deviate to probe deeper or to explore the phenomenon under study.

The research schedule was used as an aid when each research participant was interviewed. Face to face interviews were conducted whenever possible and the audio portion of all interviews was recorded. The research schedule was intended to ask individuals what they think about the use of DDD as a tool to improve software development productivity and to ask individuals about their past and future use of DDD to improve software development productivity. In addition to collecting data on the use of DDD as a tool to improve software development productivity, the research participants were asked to provide demographic data including their experience with various agile

software development methods and the size and duration of the software projects they manage.

The research schedule was based on the four definitions of analytics provided by Salam and Cearley (2012) and the KM processes of knowledge creation, accumulation, retention, and transfer as discussed by Brynjolfsson et al. (2011). The software development activities defined by the SWEBOK (2004) provided a methodology agnostic software development framework for the development of the research schedule. Although the qualitative researcher may develop a research schedule, the researcher should remain flexible throughout the interview process. The qualitative research schedule for the research study can be found in Appendix A.

A pilot study was conducted to validate the qualitative research schedule. Changes were made to the research schedule before this instrument was used to collect data. The pilot study was intended to better ensure that the questions were clear and understandable.

Procedures for Pilot Studies

According to Sullivan (2001) a pilot study can increase the validity of the research. A pilot study is a miniaturized walk-through of the research procedures and a pilot study of the research procedures for the research study was conducted. Three participants were purposely selected from the agile software development community for the pilot of the qualitative research study. Duplication of responses was avoided by purposely selecting the pilot study research participants. The three research participants were interviewed using the qualitative research schedule as shown in Appendix A.

The responses from the qualitative research schedule were analyzed following the same procedures as the research study. Changes were made to the qualitative research schedule based on the feedback from the participants in the pilot study. The qualitative research schedule was edited to clarify questions and prompts were prepared to better ensure that the research participants would be able to understand the open-ended interview questions (Smith et al., 2009).

Procedures for Recruitment, Participation, and Data Collection

This section of the dissertation describes the procedures that were used to describe from where the data was collected and from whom the data was collected. The exit strategy that was used with the research participants is discussed and the communication procedures that were used following the interviews are discussed. The steps that were taken after the interviews are explained.

Data Collection Details. Face-to-face semistructured interviews were planned as the data collection method for the qualitative research as recommended by Smith et al. (2009). Field notes were taken during the interviews. The research participants were asked to provide documentation that supports or explains their experiences with descriptive analytics, diagnostic analytics, prescriptive analytics, or predictive analytics used during each phase of the software development process. The research participants were also asked to provide documentation that supports or explains their experiences with knowledge creation, accumulation, retention, or transfer during each phase of the software development process. Any documentation provided was analyzed and coded along with the field notes and the interviews for their relevance and contribution to the

understanding of the phenomenon of DDD in software development organizations. The interviews were recorded to ensure that the interviewees' exact words were captured (Smith et al., 2009).

An initial interview was scheduled for one hour with each research participant. A second one-hour interview was scheduled if more time was needed to discuss all the questions on the research schedule or if the research participant had additional information to share. The research participants were asked follow up questions via telephone or email as necessary.

How Participants Exit the Study. The research participants were provided with the purpose for the study and the plan for scheduling interviews during the research participant selection process. The research participants were reminded when the first one-hour interview was scheduled that a second one hour interview would be scheduled if needed. This set expectations for the research participants regarding the start and stop time and the duration of the interviews.

The research participants were provided with my contact information, they were told that they might be contacted to answer questions during the analysis process, and they were told that they would be given several days to review a copy of their interview transcript and make corrections if needed. Trust was built between me and the research participants by setting expectations at the beginning of the research process and by reviewing the research purpose and plan with the research participants at the end of the interview process.

Follow-up Procedures. Immediately after each interview, I wrote notes to capture my thoughts on the interview and to document any observations I did not capture during the interview process. When the interview process was complete, thank you notes were sent to each of the research participants. The research participants were told when they could expect to receive a summary of the research findings and the research participants were told that they might be asked to answer follow-up questions as the data was analyzed.

Data Analysis Plan

There are different approaches to phenomenological analysis and although the IPA data collection and analysis process is flexible, the goal was to systematically analyze the data as recommended by Creswell (2007), Smith et al. (2009), and Patton (2002). The data analysis process began with a description of my own experience with the phenomenon under study including a description of my assumptions, viewpoint, and perspective, which Patton referred to as *epoche*. For the remaining steps in the qualitative data analysis, I followed the seven-step IPA process recommended by Smith et al..

1. The first interview transcription was read and reread to understand the meaning of the whole interview. Extraneous information was identified and unique statements that describe how the research participant's experienced the phenomenon was identified.
2. Comments were made on the interview content including comments on the linguistics and the concepts conveyed.

3. Themes within the interview were identified.
4. Patterns were identified between the emergent themes
5. Steps 1-4 were repeated for the remaining interviews
6. Patterns were identified across the interviews
7. The results of the analysis was interpreted based on the themes identified, the comments made within each interview, and the literature.

In addition to analyzing the interview transcripts, any other data provided by the research participants was systematically analyzed and the themes were coded for all of the qualitative data throughout the data collection and data analysis process as recommended by Maxwell (2005) and Smith et al. (2009). Codes can be created, by forming “organizational, substantive, and theoretical categories” (Maxwell, 2005, p. 97). For the qualitative study, the following organizational categories were defined: *people*, *project setting*, *process*, and *perspectives*.

According to Maxwell (2005) substantive categories can only be assigned during the data analysis process. Substantive categories are subcategories of the organizational categories and they describe the research “participants’ concepts and beliefs” (Maxwell, 2005, p. 97). Substantive categories were defined during the analysis process once the data collection process began. Theoretical categories represent a more abstract framework that can inductively evolve from the data analysis process and the theoretical categories represent the researchers thinking rather than the research participants’ thinking (Maxwell, 2005). The research questions included questions that facilitated categorization and questions that facilitated connecting the themes and categories.

Initially, one theoretical category was defined, *meaning*. Subcodes were defined after the data collection process began.

The demographic data was intended to anonymously describe the research participants based on their background and experience. The demographic data was analyzed to describe the research participant's years of experience and the size of the software projects they manage. Scrum is an agile project management framework that can be used alone or in coordination with any agile process or processes. In addition to Scrum, the research participants may have had experience with other agile software development methods; therefore, the research participants were asked to discuss their familiarity with agile software methods other than Scrum.

Qualitative data analysis (QDA) research tools can make it easier to mix data collection and data analysis for qualitative research. QDA software was used to help ensure that the data was organized throughout the data collection and data analysis processes so that the accuracy of the data provided was not compromised. I used NVivo for the qualitative research study on management understanding of DDD in agile software development organizations.

Issues of Trustworthiness

The ethical considerations and the steps that were taken to protect the rights of the research participants are discussed in this part of the dissertation. The researcher must carefully consider the possible ethical concerns related to each research study because there is no comprehensive list of all possible ethical considerations (Smith et al., 2009). Research participants could potentially be harmed by their involvement in research;

therefore, steps must be taken to protect the rights of the research participants and ensure that no harm results from the research (Sullivan, 2001).

Trustworthiness

The issues of trustworthiness in qualitative research include issues of credibility, transferability, dependability, and confirmability rather than issues of generalizability or repeatability (Guba & Lincoln, as cited in Trochim & Donnelly, 2008). Qualitative research is more about exploring the meaning of things rather than of determining the truth. Therefore, the issues of credibility, transferability, dependability, and confirmability will be discussed as they relate to the research study on DDD as a tool to improve software development productivity.

Credibility. Research results are more likely to be credible or believable if they are free from bias (Patton, 2002). Consequently, the qualitative researcher must remain neutral throughout the research process; however, neutrality is not easily attained. The research study improved credibility by maintaining an awareness of any biases and by reporting both confirming and disconfirming evidence that support any conclusions as recommended by Patton.

Transferability. According to Guba and Lincoln (as cited in Patton, 2002) qualitative research is usually not generalizable; however, qualitative research may be transferable. Qualitative research is not generalizable because the research findings are context dependent. The research findings may be transferable from the context under study to a congruent context. Therefore, the research findings from the research study may be transferable beyond the local software development community to other similar

software development communities. Verification of the transferability of the research findings is beyond the scope of the research study; however, transferability was improved by collecting data from three different groups within the Scrum community: software managers, project managers, and agile coaches.

Dependability. Although the quality of research is dependent upon the ability to repeat the research procedures, qualitative research is conducted in real-world settings where change is inevitable. Therefore, the qualitative researcher should account for the changes that occur during the study (Guba & Lincoln, as cited in Trochim & Donnelly, 2008). The research study accounted for any changes that occur as a result of conducting the research and any changes that occur within the context of the research study.

Confirmability. Because neutrality is difficult to attain, the qualitative researcher can minimize issues of trustworthiness that result from any researcher bias by describing the research procedures in detail (Lincoln & Guba, as cited in King & Horrocks, 2010). If the reader is able to confirm that the researcher's conclusions are reasonable based on the description of the data collection and analysis processes, then the research findings will be trusted. The data collection and analysis processes for the research study have been described in detail in this dissertation.

Intra and Intercoder Reliability. Sullivan (2001) noted reliability could be tested, by measuring how consistently a measure yields the same results each time it is applied. A valid measure is a reliable measure; however, a reliable measure is not necessarily valid. Consequently, measures were taken to improve the reliability of the research.

When there are multiple coders, the reliability of the research is dependent upon the consistent application of the codes; however, when there is only one researcher coding the data intercoding reliability may be improved by coding the same set of data more than once (Sullivan, 2001). The analysis process for the research included analyzing the data more than once, which should improve the reliability of the coding. Intra coding reliability may be improved by ensuring that there are clear operational definitions and by ensuring that the codes have “some conceptual and structured order” (Miles & Huberman, 1994, p. 60). Operational definitions were developed for the codes as they were defined for the research. The measures that were taken to create an organizational set of codes at the start of the research process were described in this dissertation and the methods used to develop substantive and theoretical categories during the analysis process were described in this dissertation.

Ethical Considerations

The ethical concerns were addressed throughout the research study. According to Creswell (2007) ethical research must provide answers to questions that need to be answered and generate dialogue. The purpose for the research was to better understand the use of DDD as a tool to improve software development productivity in an agile software environment. A better understanding of DDD may stimulate discussion that leads to future research in DDD.

Agreements to Gain Access. The purpose for research was to obtain knowledge and when conducting qualitative research, the researcher must remain neutral while at the same time the researcher must build rapport with the research participants (Patton, 2002).

When conducting research, the researcher must ensure that the research participants volunteer and are not coerced into participating and ensure that the privacy of the research participants is protected (Babbie, 2006). The research was conducted in an ethical manner and measures were taken to protect the confidentiality of the research participants and the data they provided. Each research participant was asked to sign a letter of informed consent, as shown in Appendix C, following approval from Walden University Institutional Review Board (IRB) to begin the research study (IRB approval #1234567).

Treatment of Human Participants. The qualitative researcher must question their underlying moral assumptions and ensure that all research participants are treated equitably (Creswell, 2007). The researcher should ensure that no harm comes to the research participants, ensure that the research participant's privacy is maintained, disclose the purpose for the research, and ensure that the research analysis and findings are reported (Babbie, 2006).

The purpose for the research was explained to the research participants and each research participant was given an informed consent form (Appendix C) as part of the research participant selection process. The research participants were given a copy of the signed informed consent form before the data collection process began. The informed consent form describes the measures that were taken to protect the research participants including the voluntary nature of the research study, the purpose for the research study, the research procedures, the risks and benefits of participation, measures to protect confidentiality, and contact information.

The research participants were told that a summary of the research findings will be shared with them when the research is ready for publication. The research participants were also asked to review the transcript of their interviews and they were given the opportunity to correct or clarify any comments from their interviews. The research participants were told that they could withdraw from the research study at any time prior to reviewing the transcript of their interview.

A coding scheme was used to ensure the research participants remain confidential and interview data is attributable to any individual. For example, each interviewee is referenced by number rather than by name and their place of employment is identified by industry rather than by name. All interviews were held in a private room, either a meeting room at the public library or a conference room selected by the research participants. Any documentation that the research participants provided is protected so that the data remains confidential.

Treatment of Data. The research participants were told that they could withdraw from the research process at any time. However, the research participants were told that if they chose to withdraw after they have been given an opportunity to review the transcript of their interview, then the data would remain part of the research study. If a research participant chose to withdraw from the research study before the interview process was complete or before they had a chance to review the transcript, their data would be eliminated from the research study.

The research data is stored on a secure laptop computer that only I can access. The interviews were tape-recorded and interviews were transcribed immediately

following each interview session. The recordings of the interviews were destroyed after the research participants had an opportunity to review the transcript of their interview.

Other Ethical Issues. The research participants were given a \$10 gift card in appreciation for their participation in the research. The monetary value of the gift card is nominal and is only intended to thank the research participants for sharing their time and expertise. No adverse effects are anticipated as a result of offering a gift card to the research participants.

Dissemination of Findings

The findings from the research study on management understanding of DDD will be shared with the research participants. The findings of the research will also be made available to the members of the Knowledge Management Association (KMA), which is a newly formed national organization focused on promoting KM best practices. Other opportunities will also be sought to disseminate the finding of the proposed research including submitting a proposal to present at the 2013 KM World conference and the APQC conference.

Summary and Transition

The research methods for a qualitative research study of software management's understanding of DDD as a tool to improve productivity within an agile software environment was discussed in this chapter. The IPA process was used for the qualitative research study. Several members of the agile software development community were interviewed to better understand the phenomenon of DDD in an agile software

environment. The results of the qualitative data analysis and data collection processes will be discussed in Chapter 4.

Chapter 4: Results

The purpose for this research study was to explore how agile coaches, project managers and software managers view data driven decision making, which includes data, analytics, and knowledge management, as a tool to improve software development productivity, to understand how agile software development organizations currently use data driven decision making to improve software development productivity, and to understand how agile software organizations may use data driven decision making in the future to improve software development productivity. The following research questions were asked in order to accomplish the goals for this research study.

1. What do software managers, project managers, and agile coaches in agile software environments think about the use of DDD to improve software development productivity?
2. How do software managers, project managers, and agile coaches in agile software environments currently use descriptive analytics, diagnostic analytics, prescriptive analytics, and predictive analytics, or knowledge creation, retention, accumulation, and transfer to improve software development productivity?
3. How do software managers, project managers, and agile coaches in agile software environments think descriptive analytics, diagnostic analytics, prescriptive analytics, and predictive analytics, or knowledge creation, retention, accumulation, and transfer could be used to improve software development productivity?

4. What obstacles do software managers, project managers, and agile coaches in agile software environments think their organizations need to overcome to improve software development productivity?

Pilot Study

The purpose for conducting a pilot study was to determine the adequacy of the research study design and to assess how long it would take to complete the research study (Bazeley, 2013). The goals were to obtain feedback from the research participants on the data gathering procedures used for this research study in order to make improvements to the research study and to execute the research data gathering and analysis procedures in order to make improvements to the research study. The same research procedures were used to conduct a pilot study as were used for the research study on data driven decision making as a tool to improve software development productivity. NVivo was used to facilitate the analysis of the pilot study data just as NVivo was used to analyze the data for the research study.

Three research participants were selected to participate in the pilot study from software teams who are currently using agile software development methods. An agile coach, a project manager, and a software manager were interviewed for the pilot study and their interviews were recorded and transcribed. The same Qualitative Research Schedule was used to conduct the interviews for the pilot study as for the research study. In addition to answering the interview questions, the pilot study participants were asked to provide feedback on the data gathering procedures used for the pilot study.

One of the pilot study participants said that the research questions were too broad and that more specific questions should be asked, such as, “How do you capture information for a retrospective?” However, this same pilot study participant said that it might be difficult for other research participants to answer more specific questions. Another pilot study participant said that the questions were fine; however, using the SWEBOK activities as a framework for the research questions could be confusing to some research participants if they assume the SWEBOK activities imply using a waterfall software development process rather than using an agile software development process. The third pilot study participant said that the questions were “pretty good;” however, some of the terminology caused them to think.

As a result of the feedback from the pilot study, the Qualitative Research Schedule, which included background information on the research study, the research study questions, and the definitions of the terminology used in the interviews was sent to the research study participants prior to each interview. In some cases, the research study participants were asked probing questions to ensure the questions were understood and that the answers were captured. The research participants were also told prior to the start of each interview that although the SWEBOK activities were used as a framework for the research study, the SWEBOK activities were considered generic software development activities and did not refer to use of waterfall software development process.

The transcriptions of the pilot study interviews were analyzed using NVIVO. Case nodes were created for the pilot study interviews and the interviews were coded based on the research study framework including analytics, the SWEBOK activities, and

the Knowledge Management types. The interviews were summarized using framework matrices. A model was created to show the relationships between Knowledge Management types, the SWEBOK activities and software development productivity and another model was created which showed the relationship between the types of analytics, the SWEBOK activities, and software development productivity. Themes began to emerge as a result of this data analysis process. The results of the pilot study indicated that the research procedures were adequate for accomplishing the goals of the research study.

Research Setting

The research interviews were conducted either face-to-face or on the telephone. The face-to-face interviews were conducted at a branch of the local library convenient to the research participants. A conference room was used for all but one of the interviews to ensure that the quality of the recording was optimized for transcription and to maintain the anonymity of the research participants. In one case, a conference room was not available at a location convenient to the research participant, and the research participant agreed to be interviewed at a table located in a quiet corner of the library. The research participants who were interviewed on the telephone selected a location that would provide the privacy they desired. No information was provided by the research participants that negatively influenced the interpretation of the data, and none of the research participants indicated that any personal or organizational issues were affecting them and their ability to answer the research questions.

Demographics

All of the research participants were agile coaches, project managers, or software managers who were using agile software development methods at the time of their interview. The research participants were asked to state their primary role. Although three of the research participants said that their primary role is an agile coach, three said their primary role is a project manager, and three of the research participants said that their primary role is a software manager, 66% of the project managers and the software managers said that they had multiple roles.

All three of the agile coaches said they have used agile software development methods for over 5 years, two of the project managers said they have used agile software development methods for over 5 years and one of the project managers said they have between one and three years of agile software development experience. Two of the software managers said they have used agile software development methods for over 5 years and one software manager said they used agile software development methods for three to five years. Table 2 shows the number of years of experience for each research participant role.

Table 2

Research Participant Roles and Years of Agile Software Development Experience

Role	#of Participants	Years of Experience
Agile Coach	3	>5 years
Project Manger	2	>5 years
Project Manger	1	1<>3 years
Software Manager	2	>5 years
Software Manager	1	3<>5 years

The research participants were asked to describe the size of their software projects, the duration of their software projects and the agile methodologies used. Three research participants described their software projects as small, three research participants described their software projects as medium, two research participants described their software projects as large, and one research participant stated that they were engaged with small, medium, and large software projects. Software projects with less than three teams were categorized as small for this research study. Software projects with less than five teams were categorized as medium for this research study and software projects with over five teams were categorized as large for this research study. While four of the research participants described the duration of their projects as less than one year, five of the research participants described the duration of their projects as over

one year and up to 5 years in duration. Table 3 shows the number of research participants by project size.

Table 3

Research Participant Project Size

Project Size	#of Participants	#of Teams
Small	3	<3
Medium	3	<5
Large	2	>5
Small, Medium & Large	1	1 - >5

The research participants stated that they used a variety of agile software development methods including Scrum, Lean\Kanban, and XP. Two of the research participants said that they used Scrum, Lean\Kanban, and XP, Two of the research participants said they used Scrum and Lean\Kanban, four of the research participants said that they used Scrum, and one of the research participants said that they used Lean\Kanban. Table 4 shows the number of research participants by research methods used.

Table 4

Research Participant Agile Software Development Methods Used

Agile Software Development Method	# of Participants
Scrum	4
Scrum, Lean\Kanban	2
Scrum, Lean\Kanban, XP	2
Lean\Kanban	1

Data Collection

The data collection methods used for this research study included semistructured face-to-face and telephone interviews. A Qualitative Research Schedule was used to ensure that the same questions were asked all of the research participants. The research participants were provided with a Qualitative Research Schedule prior to the interview and they were provided with a copy of the Qualitative Research Schedule if they did not have the document readily available at the time of the interview. The Qualitative Research Schedule was reviewed with each research participant prior to beginning the interview and each interviewee was told that they could refer to the definition of terms in the Qualitative Research Schedule at any time during the interview. The interviewees were also told that the SWEBOK activities are generic and not specific to any particular software development methodology and that the SWEBOK activities are not specific to a waterfall software development methodology.

A total of nine—one to one and a half hour interviews were conducted: five face-to-face interviews and four telephone interviews. Field notes were taken during the interviews. The research participants were asked to provide demographic data as well as answer questions on their attitude toward the need to improve software development productivity and the use of analytics and knowledge management to improve software development productivity. The interviews were recorded and transcribed.

Although the research participants were asked to provide documentation that supported their experiences with DDD in an agile software development environment, only one of the research participants provided any documentation to support or explain their experiences with descriptive, diagnostic, prescriptive, or predictive analytics or with knowledge creation, accumulation, retention, or transfer during each phase of the software development process. The documentation that was provided was analyzed and coded along with the field notes and the interviews for their relevance and contribution to the understanding of the phenomenon of DDD in software development organizations.

Data Analysis

The data analysis process began with a description of my own experience with the phenomenon under study including a description of my assumptions, viewpoint, and perspective, which Patton (2002) referred to as *epoche*. The data was systematically analyzed using the seven-step IPA process for data collection and analysis (Smith et al., 2009).

1. The first the interview transcript, the interview notes, and in one case, the examples were read and reread to understand the meaning of the whole

interview. Extraneous information was identified and unique statements that describe how the research participant's experienced the phenomenon were identified.

2. Comments were made on the interview content including comments on the linguistics and the concepts conveyed.
3. Themes within the interview were identified.
4. Patterns were identified between the emergent themes
5. Steps 1-4 were repeated for the remaining interviews
6. Patterns were identified across the interviews
7. The results of the analysis were interpreted based on the themes identified, the comments made within each interview, and the literature.

The QDA software application, NVIVO, was used to help ensure that the data was organized throughout the data collection and data analysis processes so that the accuracy of the data provided was not compromised. After the interview transcripts, the interview notes, and the examples were read and re-read, they were coded based on the conceptual framework for this research study. Framework matrices were developed to summarize each interview and themes were identified. Queries, coding matrices, and models were created to analyze the data across the emergent themes and additional codes were created to explore the emergent themes. The process was repeated for all of the qualitative data throughout the data collection, and data analysis process as recommended by Maxwell (2005) and Smith et al. (2009). Initially, codes were created based on the conceptual framework to identify the people, the project setting, the process, and

perspectives then additional codes were created to analyze what the research participants said about agile practices and productivity as shown in Table 5.

Table 5

Codes that Emerged from the Data

Category	Codes
SWEBOK Activities:	Requirements, Design, Construction, Testing, Maintenance, Configuration Management, Engineering Management, Process, Tools and Methods, Quality
Analytic Types:	Descriptive, Diagnostic, Predictive, Prescriptive
Knowledge Management Activities:	Accumulation, Creation, Retention, Transfer
Agile Practices:	Scrum, User Stories, Continuous Improvement, Burndown charts, Kanban, Meetings, XP, Retrospective, Pair Programming, Test Driven Development

Similar substantive categories emerged from the data analysis of the use of analytics to improve software development productivity and the use of KM to improve software development productivity as shown in Table 6. While the research participants discussed how communication and collaboration are used to improve software development productivity, the research participants discussed how software development

productivity is improved when analytics are used to improve security and to estimate, plan, and forecast.

Table 6

Categories that Emerged from the Data Analysis Process

Categories for Use of Analytics	Categories for Use of KM
Continuous Improvement	Continuous Improvement
NA	Communication
Decision Making	Decision Making
Estimate, plan, forecast	NA
Inspect and Adapt	Inspect and Adapt
Quality	Quality
Risk Management	Risk Management
Security	NA
Transition from Development to Release	Transition from Development to Release

The themes that emerged from the data are based on the researcher's interpretation of the research participants' responses to the interview questions. Continuous improvement is part of the agile software development process as was discussed in relationship to the use of analytics and KM.

Essentially I'm describing the retrospection process. What went wrong? What went right? All of those things feed into... Okay we're going to use all of these metrics in all the phases to improve in the future.

Continuous improvement in both the configuration management and engineering management will get better in time is long test that knowledge is transferred among the whole organization it won't prove.

Communication is used to share knowledge in agile software development; however, the communication is structured to facilitate collaboration and to minimize wasted time.

We do a daily standup in the morning. We actually have three development teams. One of them is a really small one at the moment. The two really big teams have their daily standup at 9:30 in the morning and then we have this third team which is focused on a big third-party integration and the rest of our management team do a management level [meeting].

Decision Making is improved when analytics and KM are used.

How much effort should go into maintenance [is] based [on] real metrics rather than on which salesperson speaks the loudest.

Whatever makes sense for that specific project is how you can determine what tools and methods. You get that from knowledge accumulation from knowledge retention of working on several projects so that's how that fits in.

Estimating, planning and, forecasting may be used to help software teams determine how productive they are.

That being said, I believe that descriptive analytics in terms of helping a team understand what they have done in the past is good and that I have used, both here and in my previous jobs, a predictive model that I had created. Very simple but in

terms of feeding in previous project timesheets by type of work and a few other factors... Very simple.

Inspect and Adapt is the agile process that is used to answer the question, “How are we doing?”

On a day-to-day basis I keep tabs on individual productivity and quality of work, you know, through the source control system.

Quality and productivity were closely linked by the research participants.

But since I said the word rework, I will jump ahead to quality. That’s a huge one from a quality perspective. So, you know, it’s the traditional... QA writes about a bug. The software engineer sends it back. QA opens it up again. No, it’s still broken and that iteration. So keeping track of those types of metrics greatly improve, not just the quality, but gives a window into your development team.

What’s missing there? Is it a communication issue or is this issue so complex that it’s, you know, changing...

Risk Management was talked about by the research participants when they discussed how KM is used to improve the probability that new employees will succeed.

I’m on boarding 2 new people this week. I’m going to schedule some time on the calendar for them... For the BA to talk to the existing BA and for the QA to talk to the existing QA and say, “Hey, let’s get an hour together and sit down and talk.” Walk through the environments and access rights, permissions and show them where we keep stuff in SharePoint and stuff like that.

Security is improved when analytics are used to inspect the code.

So an example would be if you look at the use of a static code testing tool like HP Fortify. What it does is inspect the code that you built and looks for particular kinds of security flaws and gives you a report back that says here's what I found, either warnings or severe errors. So the process that we use is taking that report and going back and inspecting it. From my standpoint that would be a diagnostic function that I have to perform.

Transition from Development to Release was discussed by the research participants as a phase in the software development process that caused conflict which negatively impacted productivity.

In my opinion, the pain points are... the transition from testing to deployment and integration was painful but it got better with additional measures so I guess that's a place to start. How did we improve that? Well, we added more measures and defined objects, which were the actual test cases themselves and their related bugs and once we define those objects, we could measure them.

The themes that emerged from the analysis of the data are documented in Tables 7 and 8.

Table 7

KM Themes that Emerged from the Data by Category

KM Category	KM Themes
Continuous Improvement	For continuous improvement To drive change across the organization
Communication	To store and find content To improve communication between stakeholders To communicate project status to stakeholders To share knowledge including agile expertise To capture knowledge just-in-time To integrate knowledge into the code
Decision Making	To know what to build and how to build it To improve team decision making To select the best tool for the job
Monitor and Adapt	To monitor and adapt To improve knowledge about project status
Quality	To improve code quality
Risk Management	To manage risk To minimize the negative effects of employee transitions
Transition from Development to Release	To transition from development to release

Table 8

Analytic Themes that Emerged from the Data by Category

Analytic Category	Analytic Theme
Continuous Improvement	For continuous improvement To improve coding productivity
Decision Making	To determine what products and features should be built To determine maintenance priorities To determine how to design and build it
Estimate, Plan, Forecast	Groom the backlog Time to design versus time to construct To estimate, plan, and forecast
Monitor and Adapt	To monitor and adapt To measure cost for delays To measure change in scope over time
Quality	To improve quality
Risk Management	To manage risk To determine the root cause of issues
Security	To improve security
Transition from Development to Release	For continuous integration and automated testing To improve the transition from development to release

Although almost all of the research participants discussed some themes, few research participants discussed other themes. For example, while 32% of the research participants discussed how analytics are currently used to estimate, plan, and forecast iterations and releases; only 2% discussed how analytics are used to improve security and while 33% of the research participants said that KM is used to store and find content only 4% of the research participants said that KM is used to onboard new employees. However, none of the themes identified were eliminated from the results of this research study because this research study did not evaluate how well software development

productivity is or could be improved by each of the activities discussed by the research participants. This research study did not attempt to correlate the percentage of research participants who discussed a theme and the effectiveness of an activity on software development productivity. For example, although only 2% of the research participants discussed how analytics could be used to improve security, in some cases, software development productivity may be greatly improved if analytics are used to improve security and software development productivity may be only moderately improved if KM is used to store and find content although 33% of the research participants discussed this theme.

Evidence of Trustworthiness

Credibility

I remained neutral throughout the process to improve the believability of the research findings. The research reported both confirming and disconfirming evidence. For example, while I reported that the agile coaches, project managers, and software managers said that software development productivity needs to improve, I also reported that one of the agile coaches and two of the software managers qualified their responses when they said, a balance needs to be maintained between productivity and quality.

Transferability

Data was collected from three different groups within the agile software development community: agile coaches, project managers, and software managers, which improved the transferability of the research findings from the context under study to a

congruent context. However, in general, qualitative study outcomes are not transferable. If some elements are transferable, these are beyond the scope of this study.

Dependability

I conducted a pilot study and I accounted for the changes that occurred within the context of the research study to improve the repeatability of the research. For example, I stated in the research proposal that Scrum coaches would be interviewed; instead, I interviewed agile coaches when it was discovered that Scrum coaches and project managers have similar roles. Because one of the pilot study participants was confused by the terminology used during the interview, I provided the research participants with the operational definitions for the terms used in the research questions during the interview process.

Confirmability

The data collection and analysis processes for the research study were described in detail to improve the confirmability of the research study findings. The research questions were included in Appendix A. QDA software was used to analyze the data, and the IPA process that was used to systematically analyze the data was described.

Research Results

The research participants were asked a series of questions as shown in Appendix A in order to answer the research questions. The responses to the interview questions were analyzed and the results for each research question are discussed in this section of the dissertation. The interview questions were also analyzed to determine the similarities and differences between responses based on the research participant demographics.

The purpose of this qualitative phenomenological research study was to explore how agile software managers view DDD as a tool to improve software development productivity and to understand how agile software development organizations may use DDD now and in the future to improve software development productivity. Agile coaches, project managers, and software managers were interviewed and the transcripts of the interviews were analyzed. An analysis of the data revealed the answers to the four research questions.

Research Question 1

The research participants were asked, “What do software managers, project managers, and agile coaches in agile software environments think about the use of DDD to improve software development productivity?” Although most of the agile coaches (3), project managers (3), and software managers (2) stated that software development productivity needs to improve, one of the agile coaches cautioned that a balance needs to be maintained between productivity and quality as did one of the software managers. One of the software managers stated that productivity is a side effect of effective software development management.

I don't want to waste time. I want to control scope. I want to understand risk and I want to manage it and I kind of tend to think of it that way. So although productivity is important I tend to think of it more as a side effect of managing all those things effectively.

The agile coaches (3), project managers (3), and software managers (3) claimed that DDD is needed to improve software development productivity. One of the agile

coaches stated that customers expect to be given dates when work will be completed; however, an on-going dialogue is needed throughout the software development process to establish priorities and to set expectations.

We have to give people that we think it's going to be this big and this long. So we have to produce some types of estimates because of the nature of the work that we do, right? But our customers hear those words as commitments and we are afraid of making those commitments. It is this double-edged sword, but from my perspective, if you tell your customers, "This is what we think it is. Let me tell you about why my estimate is probably incorrect, and what the word estimate really means, right?" Then we are having an open conversation with perhaps another adult but we are afraid to give those things. It is just a psychology problem more than anything else I think.

Another agile coach described how productivity can be improved when KM and analytics are used to inspect and adapt to ensure that the right software product is built.

Knowledge management, that's one of the things with agile and Scrum per se is that you inspect and adapt the process as you move forward and by doing that you garner the knowledge of what's working, what's successful, how you get something to the end state of done and shippable to production. The analytics deals with exactly what it is that the project needs to do. What's the project goal and defining what that project goal is then you have the ability to look at, okay, here's the first iteration, so, in the first iteration, the first Sprint, what is the Sprint goal? One of the things I always like to do is challenge my team members to look

at the Sprint goal and question the product owner, does that Sprint goal satisfy the project goal? It's that inspect and adapt... Constantly looking and analyzing it as it goes that when you get done with the project you'll have something that they desire to have built.

One of the agile coaches cautioned that although analytics and KM are valuable, data collection should be part of the process and if data collection is not part of the software development process and is only done to meet programmatic or organizational goals then it should not be done. One of the project managers also warned that software development productivity is not improved if the wrong thing is measured.

The key thing of any analytics is measuring the right thing.

According to one of the software managers, analytics need to measure the work in progress and they need to be actionable; however, the action needs to be considered carefully since numbers do not always reflect the productivity of an individual engineer according to another software manager.

Research Question 2

The research participants were asked, "How do software managers, project managers, and agile coaches in agile software environments currently use descriptive analytics, diagnostic analytics, prescriptive analytics, and predictive analytics, or knowledge creation, retention, accumulation, and transfer to improve software development productivity?" The data was analyzed to determine what the research participants said about their experiences with the people, process, and tools and methods related to the current use of analytics and KM to improve software development

productivity. Although the research participants were asked to describe how they currently use analytics and KM in each of the activities defined by the SWEBOK, the research participants did not always structure their responses based on the SWEBOK activities; therefore, I interpreted the responses.

Current Use of Analytics – People. Although the agile coaches, project managers, and software managers said that people know how to use analytics to estimate, plan and forecast (34%), inspect and adapt (21%), and to transition from development to release (21%), the agile coaches, project managers, and software managers said that people have difficulty using analytics because they do not know how to use analytics (44%). People currently know how to use descriptive analytics to answer the question, “Where are we at?” and people know how to view charts that show software project status. Although people know how to measure how long a typical installation takes, people do not have a common understanding of velocity and how to measure velocity.

Config[uration] Management and Engineering Management... It’s hard to get from an analytics perspective. You can use tools in TFS to say, “How was the build cycle? Was it successful?” ...those kinds of things but we tend not to do a lot of this. It’s advanced thinking for a lot of people. ...some projects we will use them on.

Current Use of Analytics – Process. The themes that emerged from the data were analyzed to determine how descriptive, diagnostic, predictive, or predictive analytics are currently used. The themes were analyzed to identify the differences and similarities between what each of the research participant groups said about the current

use of analytics and the themes were analyzed to determine the SWEBOK activities in which analytics are currently used. All themes identified by the research participants were included in the results, although some themes were identified by only one research participant.

Current Use of Analytics by Analytic Type. The research participants primarily use descriptive analytics (57%) and diagnostic analytics (28%) to improve software development productivity rather than predictive (15%) or prescriptive (0%) analytics as shown in Table 9. Descriptive, diagnostic and predictive analytics are used to estimate, plan, and forecast (33%).

I believe that descriptive analytics in terms of helping a team understand what they have done in the past is good and that I have used, both here and in my previous jobs, a predictive model that I had created. Very simple but in terms of feeding in previous project timesheets by type of work and a few other factors...

Descriptive, diagnostic, predictive analytics are used to monitor and adapt (22%)

I think the tendency is to look at the information from both the predictive analytics and from the descriptive analytics and then the diagnostic stuff to make a determination about what you should do. In other words if I'm looking at a burn down chart I can change things around for instance I can remove an item, remove scope. Generally then am I going to fit within the team's capacity at that point? We do things like that.

Descriptive and diagnostic analytics are used to transition from development to release (20%) to improve quality (11%) and for continuous improvement (7%). Diagnostic and

predictive analytics are used to improve security and descriptive analytics are used to manage risk and to improve decision making.

Table 9

Current Use of Analytics by Analytic Type

Category	Descriptive	Diagnostic	Predictive	Prescriptive	Total
Estimate, Plan, Forecast	13%	9%	11%	0%	33%
Monitor and Adapt	13%	7%	2%	0%	22%
Transition from Development to Release	13%	7%	0%	0%	20%
Quality	9%	2%	0%	0%	11%
Continuous Improvement	4%	2%	0%	0%	7%
Security	0%	2%	2%	0%	4%
Decision Making	2%	0%	0%	0%	2%
Risk Management	2%	0%	0%	0%	2%
Grand Total	57%	28%	15%	0%	100%

Current Use of Analytics – Category by Role. The current use of analytics was compared by role. The agile coaches (48%) discussed how they use analytics to improve software development productivity more than the project managers (26%) or the software managers (26%). The agile coaches, project managers and software managers discussed how they use analytics to estimate, plan, and forecast and to inspect and adapt (33%). The agile coaches, project managers and software managers discussed how they use analytics to transition from development to release (20%), to monitor and adapt (20%) and to improve quality (11%). The agile coaches and the project managers discussed how analytics are used for continuous improvement. Only one agile coach talked about

the use of analytics to improve security, only one software manager talked about the use of analytics for decision making and only one project manager talked about the use of analytics for risk management. The analytic themes identified by the agile coaches are shown in Table 10. The analytic themes identified by the project managers are shown in Table 11 and the analytic themes identified by the software managers are shown in Table 12.

Table 10

Current Use of Analytics - Category by Agile Coaches

Category	Descriptive	Diagnostic	Predictive	Prescriptive	Total
Estimate, Plan, Forecast	4%	4%	7%	0%	15%
Monitor and Adapt	4%	4%	2%	0%	11%
Transition from Development to Release	4%	4%	0%	0%	9%
Continuous Improvement	2%	2%	0%	0%	4%
Quality	2%	2%	0%	0%	4%
Security	0%	2%	2%	0%	4%
Decision Making	0%	0%	0%	0%	0%
Risk	0%	0%	0%	0%	0%
Grand Total	17%	20%	9%	2%	48%

Table 11

Current Use of Analytics - Category by Project Managers

Category	Descriptive	Diagnostic	Predictive	Prescriptive	Total
Estimate, Plan, Forecast	4%	2%	2%	0%	9%
Transition from Development to Release	4%	2%	0%	0%	7%
Quality	4%	0%	0%	0%	4%
Continuous Improvement	2%	0%	0%	0%	2%
Monitor and Adapt	2%	0%	0%	0%	2%
Risk	2%	0%	0%	0%	2%
Security	0%	0%	0%	0%	2%
Decision Making	0%	0%	0%	0%	0%
Grand Total	20%	4%	4%	0%	26%

Table 12

Current Use of Analytics - Category by Software Managers

Row Labels	Descriptive	Diagnostic	Predictive	Prescriptive	Total
Estimate, Plan, Forecast	4%	2%	2%	0%	9%
Monitor and Adapt	7%	2%	0%	0%	9%
Transition from Development to Release	4%	0%	0%	0%	4%
Decision Making	2%	0%	0%	0%	2%
Quality	2%	0%	0%	0%	2%
Continuous Improvement	0%	0%	0%	0%	0%
Risk	0%	0%	0%	0%	0%
Security	0%	0%	0%	0%	0%
Grand Total	19%	4%	2%	2%	26%

Current Use of Analytics - Themes by Category. The themes emerged from an analysis of the data on how agile coaches, project managers, and software managers currently use analytics. Although the research participants primarily discussed how they use analytics to estimate, plan, and forecast all of the themes that emerged from an analysis of the data are described in this dissertation. For example, although only one

software manager talked about the use of analytics to improve security, the theme was included in the results.

Estimate, Plan, Forecast. Agile coaches, project managers, and software managers use descriptive, diagnostic, and predictive analytics to estimate, plan, and forecast iterations and releases. One agile coach and one software manager talked about the use of descriptive analytics to determine how much time is spent on design versus how much time is spent constructing the software and one project manager talked about using descriptive analytics to groom the backlog.

An agile practitioner or Scrum practitioner would look at things like velocity and estimating accuracy and those pieces of information and those are useful and actually things that a well-functioning agile team of developers find useful. In other words I want to be able to estimate so that I can accurately predict how long it will take me. And I want to improve my velocity and make sure that I'm working at as high a level as I can. There are some good tools out there for helping that. Are you familiar with tools like VersionOne or Rally?

Monitor and Adapt. Agile coaches, project managers, and software managers use descriptive, diagnostic, and predictive analytics to monitor progress and to adapt the software development plan for each software development iteration.

I think the tendency is to look at the information from both the predictive analytics and from the descriptive analytics and then the diagnostic stuff to make a determination about what you should do. In other words if I'm looking at a burn down chart I can change things around for instance I can remove an item,

remove scope. Generally then am I going to fit within the team's capacity at that point? We do things like that. That does play into it.

Transition from Development to Release. Agile coaches, project managers, and software managers talked about the use of descriptive and diagnostic analytics to transition from development to release and the project managers talked about the use of diagnostic and predictive analytics to transition from development to release.

Would you consider configuration management... We do use... And again this is probably only descriptive analytics. We do use descriptive analytics and monitoring tools to take a look at and understand what our system performance requirements are. If we identify that we have to add new servers and those sorts of things, we are keeping an active daily monitor of speeds and response times and things like that and system bandwidth usage. Things like that. We actively use a lot of those.

Agile coaches, project managers, and software managers use descriptive analytics for continuous integration and automated testing. Two agile coaches also talked about the use of diagnostic analytics for continuous integration and automated testing and one project manager talked about the use of predictive analytics for continuous integration and automated testing.

We do use descriptive analytics and monitoring tools to take a look at and understand what our system performance requirements are. If we identify that we have to add new servers and those sorts of things, we are keeping an active daily

monitor of speeds and response times and things like that and system bandwidth usage. Things like that. We actively use a lot of those.

Quality. Agile coaches, project managers, and software managers use descriptive analytics to improve quality and one agile coach talked about the use of diagnostic analytics to improve quality.

From a quality perspective it easy to use tools to measure where was the defect was found, how often? We will use TFS to do that. That's pretty simple for a descriptive diagnostic perspective.

Continuous Improvement. Agile Coaches and Project Managers use descriptive and diagnostic analytics for continuous improvement.

Essentially I'm describing the retrospection process. What went wrong? What went right? All of those things feed into... Okay we're going to use all of these metrics in all the phases to improve in the future. In that respect agile does have a good process for having a traditional postmortem but shorter cycles than a waterfall. So, you can kind of prevent those mistakes from happening again.

Security, Decision Making, and Risk Management. Although few research participants talked about the use of analytics to improve productivity related to security, decision making or risk management, productivity may be improved if more agile software teams focused on security, decision making and risk management. One agile coach uses diagnostic and predictive analytics to improve security. One software manager discussed how descriptive analytics are used to determine maintenance

priorities. One project manager use descriptive analytics to manage risk by determining the root cause of issues.

Current use of Analytics - Tools and Methods. Agile coaches, project managers, and software managers discussed how they use qualitative and quantitative tools and methods to measure progress on software development projects (67%). Agile coaches, project managers, and software managers said that they use tools like Team Foundation Server (TFS), Jira, Trello, VersionOne, Rally, ScrumworksPro, and Excel to manage the backlog, to determine the software development team velocity and to view burndown charts. The backlog is groomed for each software development iteration and release, which enables the software development team to focus on the highest priority items for each software development iteration and release.

Agile software teams collect epics, which are linked to the organizational goals. The epics are decomposed into user stories and the user stories are combined with existing backlog items where they are prioritized and sized. The backlog items are linked to tasks and tests. Agile coaches, project managers and software coaches said that they measure progress by analyzing burndown charts and comparing the burndown to the team capacity.

So the tools that we use and the way we use the tools allow us to do that. So we use it both as a way to describe what is happening so for descriptive analytics. We have new ones in, how many do we have, the state that they are in, whether they been allocated to a release, whether they are currently being worked on in an iteration, and whether or not they are successfully completed. Are all the tasks

completed or have they been left out of the requirements? Which tests have been executed successfully against it as well?

Although agile coaches, project managers, and software managers said that they currently use tools and methods for configuration management (44%), they said that there is a conflict between the agile software development goal to release software frequently and the configuration management goal to maintain system stability. Consequently, agile software development teams need to provide information to the configuration management and maintenance teams that enable the configuration management and maintenance teams to plan long term.

So, the agile teams need a way to put information in a place where they can see more roadmap oriented type of data. Whether that is transforming the objects that we call epics into something that they can use in the future... I think that a tool could benefit this by making sure that the maintenance and configuration group have visibility because they are challenged.

One of the agile coaches talked about how regression testing has just begun to be used to improve security and the agile coaches and project managers talked about how tools and methods are used to improve quality (33%).

We look at quality is kind of a two-phase thing. Quality is built into the product. So that processes actually built in all the way through. Again going back to what we talked about in release planning side identifying what tests will be executed on it. What standards have to be applied and getting agreement and buy-in by the team for a definition of done for the release planning before you move into the

construction phase and then the other side of quality is the inspection or audit side of it.

Current Use of Analytics – SWEBOK Activity by Analytic Type. The research participants primarily discussed how analytics are used during the tools and methods activities (20%), the process activities (12%), the requirements activities (12%), the engineering management activities (11%), and the quality activities (11%) as shown in Table 13. What the research participants said about the use of analytics during the Tools and Methods activities was discussed previously in this dissertation. Few the research participants talked about how they currently use analytics during the testing activities, the design activities or construction activities.

Table 13

Current Use of Analytics – SWEBOK Activity by Analytic Type

SWEBOK Activities	Descriptive	Diagnostic	Predictive	Prescriptive	Total
Tools and Methods	9%	7%	3%	1%	20%
Process	7%	2%	3%	0%	12%
Requirements	7%	3%	2%	0%	12%
Engineering Management	5%	2%	2%	1%	11%
Quality	7%	3%	1%	0%	11%
Configuration Management	4%	4%	0%	0%	9%
Testing	7%	2%	0%	0%	9%
Construction	5%	1%	0%	0%	7%
Design	4%	1%	0%	0%	5%
Maintenance	4%	1%	0%	0%	5%
Grand Total	59%	27%	12%	2%	100%

Current Use of Knowledge Management – People. The agile coaches, project managers, and software managers described how they currently know how to use KM (87%) to improve software development productivity. People know how to use KM

processes to retain and transfer knowledge and they know how to collaborate to reach decisions and resolve issues. However, the project managers stated that people currently have difficulty using KM effectively because they do not have the time to accumulate and transfer knowledge (22%). Software teams may accumulate knowledge when they work in a culture of knowledge sharing (22%).

So, knowledge management and accumulation and creation... The whole concept... I see the value of it. It continually goes against what software development teams want to be doing. Is there an easy way for me to accumulate [knowledge] without impacting software development?

Current Use of Knowledge Management – Process. The themes that emerged from the data were analyzed to determine how knowledge accumulation, creation, retention, and transfer are currently used. The themes were analyzed to identify the differences and similarities between what each of the research participant groups said about the current use of KM. The data was analyzed to determine what the research participants said about the use of KM in each of the SWEBOK activities.

Current Use of KM by KM Activity. Agile coaches, project managers, and software managers use knowledge accumulation (19%), creation (24%), retention (26%), and transfer (31%) to improve software development productivity as shown in Table 14. The research participants primarily use knowledge management to improve communication (36%), to improve decision making (17%), for risk management (17%), and to improve quality (11%). A few of the agile coaches, project managers, and

software managers use knowledge management for continuous improvement, to transition from development to release, and to monitor and adapt.

Table 14

Current Use of Knowledge Management by KM Process

Category	Accumulation	Creation	Retention	Transfer	Total
Communication	9%	7%	10%	10%	36%
Decision Making	3%	4%	4%	6%	17%
Risk	3%	4%	4%	6%	17%
Quality	3%	3%	3%	3%	11%
Continuous Improvement	0%	4%	1%	4%	10%
Transition from Development to Release	1%	1%	1%	3%	7%
Monitor and Adapt	0%	0%	1%	0%	1%
Grand Total	19%	24%	26%	31%	100%

Current Use of KM – Category by Role. The current use of KM was compared by role. The agile coaches (43%) and the software managers (36%) discussed how they use KM to improve software development productivity more than the project managers (21%). The agile coaches, project managers and software managers discussed how they use KM to improve communication, to improve decision-making and for risk management. While the agile coaches and project managers discussed how they use KM improve quality and for continuous improvement, the software managers did not discuss how KM is used to improve quality and for continuous improvement. The project managers and software managers talked about how they use KM to transition from development to release and one software manager talked about how KM is used to monitor and adapt. The KM themes identified by the agile coaches are shown in Table

15. The KM themes identified by the project managers are shown in Table 16 and the KM themes identified by the software managers are shown in Table 17.

Table 15

Current Use of KM - Category by Agile Coaches

Category	Accumulation	Creation	Retention	Transfer	Total
Communication	3%	4%	3%	4%	14%
Decision Making	3%	3%	3%	3%	11%
Continuous Improvement	0%	3%	1%	3%	7%
Quality	1%	1%	1%	1%	6%
Risk	0%	1%	0%	3%	4%
Monitor and Adapt	0%	0%	0%	0%	0%
Transition from Development to Release	0%	0%	0%	0%	0%
Grand Total	7%	13%	9%	14%	43%

Table 16

Current Use of KM - Category by Project Managers

Category	Accumulation	Creation	Retention	Transfer	Total
Communication	1%	0%	3%	1%	6%
Quality	1%	1%	1%	1%	6%
Continuous Improvement	0%	1%	0%	1%	3%
Decision Making	0%	0%	1%	1%	3%
Risk	0%	0%	1%	1%	3%
Transition from Development to Release	0%	0%	0%	1%	1%
Monitor and Adapt	0%	0%	0%	0%	0%
Grand Total	3%	3%	7%	9%	21%

Table 17

Current Use of KM - Category by Software Managers

Category	Accumulation	Creation	Retention	Transfer	Total
Communication	4%	3%	4%	4%	16%
Risk	3%	3%	3%	1%	10%
Transition from Development to Release	1%	1%	1%	1%	6%
Decision Making	0%	1%	0%	1%	3%
Monitor and Adapt	0%	0%	1%	0%	1%
Continuous Improvement	0%	0%	0%	0%	0%
Quality	0%	0%	0%	0%	0%
Grand Total	9%	9%	10%	9%	36%

Current Use of KM - Themes by Category. The themes emerged from an analysis of the data on how agile coaches, project managers, and software managers currently use KM. Although the research participants primarily discussed how they use KM for communication, all of the themes that emerged from an analysis of the data are described in this dissertation. For example, although only one software manager talked about the use of KM to monitor and adapt; the theme was included in the results.

Communication. The agile coaches and software managers said that knowledge accumulation, creation, retention and transfer are used to store and find content and one project manager said that knowledge retention and transfer are used to store and find content.

We are a Microsoft shop so of course it is in SharePoint. We tuned the search engines and all that kind of stuff. We are also associating metadata tags to the deliverables before they are stored with the artifacts. So as part and parcel of that, the accumulation portion is beginning again.

While the agile coaches and software managers discussed how they currently use knowledge accumulation, creation, retention and transfer to communicate project status to stakeholders, one project manager talked about how they use knowledge retention to communicate project status to stakeholders. One agile coach said that knowledge accumulation, creation, retention and transfer are used to share agile expertise and one project manager said that knowledge retention and transfer are used to share agile expertise. One project manager discussed how knowledge transfer is used to share knowledge.

For example, when I talk to people in my department about anything pertinent to what we're going to do... Any risks need to be recorded there and shared through our project management processes. Some of our processes are... So we have regular project reviews with stakeholders and the leadership teams. Typically, most of those folks... And we post our report so we have... Our reports are stored in SharePoint so they are produced and then put into SharePoint. I can tell you that many people don't go there to look at them.

The agile coaches and the software managers said that knowledge accumulation, creation, retention, and transfer are used to improve communication between stakeholders. One project manager said that knowledge accumulation is used to improve communication between stakeholders.

We use standard Scrum meetings for knowledge transfer with the usual Scrum dictates of what I did I do since the last meeting, what am I going to do, what impediments do I have, and what Scrum topics do I need to talk to the rest of the

team about? That can cover anything from a requirements conversation, a programmatic design, construction and any of those activities or most all of them.

Decision Making. One agile coach discussed how knowledge accumulation, creation, retention, and transfer are used to know what to build and how to build it. An agile coach and a software manager said that knowledge creation and transfer are used to know what to build and how to build it.

The best communication is a whiteboard, markers, and face-to-face conversation. Getting back to the three C's: the card, the conversation, and the confirmation. That's how you get at knowing what is you need to build and how to build it, through those conversations.

One agile coach said that knowledge accumulation, retention, and transfer are used to select the best tool for the job and one project manager said that knowledge retention and transfer are used to improve team decision making.

For me, sort of with our agile mindset when we were doing Scrum we had kind of that an agile practices guide for our department. This is our story points, kind of metric table. How to decide if it is a three or a five or and eight. How we should be using Mercury or how we should be using Jira. So we have some best practices documentation there that people can refer to.

Risk Management. One software manager said that knowledge accumulation, creation, retention and transfer are used to manage risk while one software manager said that knowledge accumulation, creation, and retention are used to manage risk by minimizing the negative impact of acquiring or losing employees. One of the project

managers discussed how knowledge retention and transfer are used to manager risk by retaining basic templates and information for use by new employees.

We had a key employee leave around September. She had been here for about 6 years and she was the product development manager for about 2 or 3 of those years so she had a lot of knowledge of the system and how things worked and how things should work and how they are supposed to work and because the culture around here has been about documenting everything in the wiki then most of her knowledge was captured and we were able to pull forward.

One of the agile coaches talked about how pair programming is used to reduce risk by improving the skills and knowledge of all of the software developers.

For construction, pair programming. Transfer that knowledge. Tear down the silos. If all you do is backend DB work... I've got a team I'm working with at [organization] now doing that. Transferring the knowledge of the backend development to the front end, etc. it's going to be a time on that team when anyone on the team can do any task, which is much better than the silo effect.

Quality. One agile coach and one project manager talked about the use of knowledge accumulation, creation, retention, and transfer to improve code quality.

Continuous Improvement. One of the agile coaches said that knowledge creation, retention, and transfer are used for continuous improvement and another agile coach and one of the project managers said that knowledge creation and transfer are used for continuous improvement. One agile coach said that knowledge creation and transfer are used to drive change across the organization.

We're very strong proponents of retrospectives. Each iteration ends with a retrospective where the team can look at what they did and look at improving the process but also a structure that identifies what is in their purview to change. There are some things they can fully change how they are doing it and there are other things where they can't just deviate completely from the architecture that the rest of the product might be following for example. So that kind of a retrospective as well as using production support or operations and maintenance retrospectives to look at how the software is working. It is a deployed set of software and they are looking at what kinds of changes, what kind of non-functional changes might to be driven out of the maintenance group back into the backlog as nonfunctional requirements.

Transition from Development to Release. One software manager said that knowledge accumulation, creation, retention, and transfer are used to transition from development to release and one project manager said that knowledge transfer is used to transition from development to release.

Monitor and Adapt. One software manager said that knowledge retention is used to monitor and adapt.

Current Use of Knowledge Management - Tools and Methods. Although agile coaches, project managers, and software managers discussed how KM tools and methods are used to qualitatively measure progress on software development projects, they did not discuss how KM tools and methods are currently used to quantitatively measure progress on software development projects. One of the agile coaches described how face-to-face

communication is used to transfer knowledge about project status during the daily standup meeting.

Told the team that I had just three questions: what did you do yesterday, what are you going to do today and the only thing I want to hear as the Scrum master is you have anything impeding your progress? I want to know that and we'll talk afterwards and you aren't reporting to me. You are just having a conversation with your team members of what it is you're doing and that is the key importance of it.

One of the software managers discussed how KM tools and methods are currently used to accumulate, create, retain, and transfer knowledge that minimize wasted time as long as the software development team follows the process.

We have a wiki and members of the team will write up summaries of how things work when they reach one of those points. So part of the discipline around here is if you don't have the time or it's not part of the current scope of a project to kind of rework some of those things that don't make sense then document them as code, if you will. Then you have to go describe it somewhere so that then the next person to run into it has a reference to it. That's been going phenomenally well for us, especially for some things that are not worth rebuilding because maybe they get revisited once a year.

Current Use of KM – SWEBOK Activity by KM Activity. The research participants primarily discussed how KM is currently used during the tools and methods activities (21%) and the process activities (14%) as shown in Table 18. What the

research participants said about the use of KM during the Tools and Methods activities was discussed previously in this dissertation. A few research participants discussed how KM is currently used during testing activities (13%) and the requirements activities (12%) and even fewer research participants discussed how KM is currently used during the configuration management, design, quality, construction, and maintenance activities. For example, one of the software managers discussed how the Scrum process includes KM activities.

In terms of the Scrum process itself obviously we have the normal, you know, Sprint reviews and all the normal ceremonies that go on in Scrum. The main place that we store information... We also use Jira. Half of our organization uses Jira because they haven't moved to Team Foundation Server but either way between SharePoint and between the actual tool itself almost everything related to those projects is really stored in those two areas and they are shared through meetings basically.

Table 18

Current Use of KM – SWEBOK Activity by KM Activity

SWEBOK Activity	Accumulation	Creation	Retention	Transfer	Total
Tools and Methods	6%	4%	6%	6%	21%
Process	2%	4%	3%	4%	14%
Testing	2%	4%	3%	4%	13%
Requirements	3%	3%	3%	4%	12%
Configuration Management	1%	3%	1%	3%	9%
Design	1%	3%	1%	3%	9%
Quality	2%	2%	2%	2%	9%
Construction	1%	2%	1%	2%	6%
Maintenance	1%	2%	1%	2%	6%
Engineering Management	0%	1%	0%	1%	3%
Grand Total	19%	29%	21%	31%	100%

Research Question 3

The research participants were asked, “How do software managers, project managers, and agile coaches in agile software environments think descriptive analytics, diagnostic analytics, prescriptive analytics, and predictive analytics, or knowledge creation, retention, accumulation, and transfer could be used to improve software development productivity?” Although the research participants were asked to describe how they could use analytics and KM in the future in each of the activities defined by the SWEBOK, the research participants did not always structure their responses based on the SWEBOK activities; therefore, I interpreted the responses.

Future Use of Analytics – People. The agile coaches, project managers, and software managers said that people needed to know how to use analytics to improve software development productivity (44%). People need to know more about what to measure and people need to know how to analyze the user feedback to improve the

software. People need to know how to measure productivity in an agile software development environment, which is delivering value to the customer rather than measuring lines of code or counting function points.

....And that's something I don't necessarily have the answer for, what are the ideal analytics or what are the right things that will help the productivity. For us right now, having a measure of delivering value is kind of for me the paramount thing.

Future Use of Analytics – Process. The themes that emerged from the data were analyzed to determine how descriptive, diagnostic, predictive, or prescriptive analytics could be used in the future. The themes were analyzed to identify the differences and similarities between what each of the research participant groups said about how analytics could be used in the future and the themes were analyzed to determine the SWEBOK activities in which analytics could be used in the future.

Future Use of Analytics by Analytic Type. The research participants said that descriptive (37%), diagnostic (26%), and predictive (21%) and prescriptive (16%) analytics could be used in the future to improve software development productivity as shown in Table 19. Descriptive, diagnostic, predictive, and prescriptive analytics could be used in the future to improve decision making (26%). Descriptive, diagnostic, and predictive analytics could be used in the future to improve quality (26%). Descriptive, predictive, and prescriptive analytics could be used in the future to estimate, plan, and forecast (16%). Descriptive and diagnostic analytics could be used in the future to for continuous improvement. Diagnostic and predictive analytics could be used in the future

to improve the transition from development to release. Diagnostic analytics could be used in the future to improve security and descriptive analytics could be used in the future to monitor and adapt.

Table 19

Future Use of Analytics ALL

Category	Descriptive	Diagnostic	Predictive	Prescriptive	Total
Decision Making	5%	5%	5%	11%	26%
Quality	16%	5%	5%	0%	26%
Estimate, Plan, Forecast	5%	0%	5%	5%	16%
Continuous Improvement	5%	5%	0%	0%	11%
Transition from Development to Release	0%	5%	5%	0%	11%
Monitor and Adapt	5%	0%	0%	0%	5%
Security	0%	5%	0%	0%	5%
Grand Total	37%	26%	21%	16%	100%

Future Use of Analytics – Category by Role. The future use of analytics was compared by role. The software managers (53%) and the agile coaches (37%) discussed how analytics could be used in the future to improve software development productivity more than the project managers (11%). The agile coaches, project managers, and software managers discussed how analytics could be used in the future to improve decision-making (33%). The project managers and software managers discussed how analytics could be used to improve quality. The agile coaches said that analytics could be used in the future to estimate, plan, and forecast and to improve security. The software managers said that analytics could be used to in the future for continuous improvement, to transition from development to release, and to monitor and adapt. The analytic themes identified by the agile coaches are shown in Table 20. The analytic themes identified by

the project managers are shown in Table 21 and the analytic themes identified by the software managers are shown in Table 22.

Table 20

Future Use of Analytics - Category by Agile Coaches

Category	Descriptive	Diagnostic	Predictive	Prescriptive	Total
Decision Making	0%	5%	5%	5%	16%
Estimate, Plan, Forecast	5%	0%	5%	5%	16%
Security	0%	5%	0%	0%	5%
Continuous Improvement	0%	0%	0%	0%	0%
Monitor and Adapt	0%	0%	0%	0%	0%
Quality	0%	0%	0%	0%	0%
Transition from Development to Release	0%	0%	0%	0%	0%
Grand Total	5%	11%	11%	11%	37%

Table 21

Future Use of Analytics - Category by Project Managers

Category	Descriptive	Diagnostic	Predictive	Prescriptive	Total
Decision Making	5%	0%	0%	0%	5%
Quality	5%	0%	0%	0%	5%
Continuous Improvement	0%	0%	0%	0%	0%
Estimate, Plan, Forecast	0%	0%	0%	0%	0%
Monitor and Adapt	0%	0%	0%	0%	0%
Security	0%	0%	0%	0%	0%
Transition from Development to Release	0%	0%	0%	0%	0%
Grand Total	11%	0%	0%	0%	11%

Table 22

Future Use of Analytics - Category by Software Managers

Category	Descriptive	Diagnostic	Predictive	Prescriptive	Total
Quality	11%	5%	5%	0%	21%
Continuous Improvement	5%	5%	0%	0%	11%
Transition from Development to Release	0%	5%	5%	0%	11%
Decision Making	0%	0%	0%	5%	5%
Monitor and Adapt	5%	0%	0%	0%	5%
Estimate, Plan, Forecast	0%	0%	0%	0%	0%
Security	0%	0%	0%	0%	0%
Grand Total	21%	16%	11%	5%	53%

Future Use of Analytics - Themes by Category. The themes emerged from an analysis of the data on how agile coaches, project managers, and software managers could use analytics in the future. Although the research participants primarily discussed how they could use analytics to improve decision making and quality all of the themes that emerged from an analysis of the data are described in this dissertation. For example, although only one agile coach talked about the use of analytics to improve security, the theme was included in the results.

Decision Making. A project manager talked about how descriptive analytics could be used in the future to determine what features should be developed. An agile coach and a software manager said that prescriptive analytics could be used in the future to determine how to design and build the software product.

That would also be helpful from a design standpoint. So we were thinking about moving from... I don't know, what is a good example? Moving from, who knows what it is...one particular architecture to a second one... Kind of predictive like

simulators that would validate...” Is that the right design approach to take to this code problem or are you making things worse or better?” Those sorts of things. An agile coach said that diagnostic and predictive analytics could be used to determine what products and features should be built.

Predictive analytics would probably come and maybe on the process of the whole thing if you are looking at the project. Predicting that what it is you are developing, is it going to be useful? Are people going to use it? Are they going to need it? How is it going to go in the marketplace? By utilizing predictive analytics would give you an idea of maybe what share of the market can you hope to obtain by coming out with this new product, new software to help people live easier.

Quality. A software manager said that descriptive, diagnostic, and predictive analytics could be used in the future to improve quality and a project manager and a software manager said that descriptive analytics could be used in the future to improve quality.

I tend to find that really good software developers tend to have very little effect on improvements in their code in development. Where I see the bigger effect is in assisting younger or more junior developers coming up to learn things quicker as to where certain problem areas are. Part of my staff is very experienced in they tend to know what is or tends to be difficult and to focus on that first whereas the more junior developers have no intuition and that and so the metrics would definitely help them understand where the difficulties are going to be and allow

them to focus more on that. So, I think the improvements there would be more on quality than anywhere.

Estimate, Plan, Forecast. One agile coach said that descriptive, predictive, and prescriptive analytics could be used in the future to forecast.

This is really where I'm trying to take my company from predictive to prescriptive. This is where you're getting into forecasting models. It could be perhaps. From a prescriptive perspective you're really trying to predict what is your future timeline for multiple releases across the project, right? What do we anticipate our burn to be?

Continuous Improvement. One software manager said that descriptive analytics could be used in the future to improve coding productivity and another software manager said that diagnostic analytics could be used in the future to improve coding productivity.

...a commit tool that would go through your code and identify memory leaks and things like that. Those tools are definitely hugely beneficial...Kind of self-testing diagnostic tools.

Transition from Development to Release. A software manager said that diagnostic and predictive analytics could be used in the future to improve the transition from development to release.

Monitor and Adapt. One software manager said that descriptive analytics could be used to measure the change in scope over time and to measure the cost for delays.

Security. An agile coach said that diagnostic analytics could be used in the future to improve security.

Future Use of Analytics - Tools and Methods. Agile coaches, project managers, and software managers talked about how qualitative and quantitative tools and methods could be used in the future to measure progress on software development projects (66%). One of the project managers discussed the need for automated user interface testing and one of the software managers stated that tools and methods are needed to measure the progress of each user story through the Kanban process when agile Lean software development methods are used. Although the agile coaches, project managers, and software managers stated that software development organizations may better understand the scope of work to be completed if better forecasting tools and methods were available, they doubted that better forecasting tools and methods would be used in an agile software development environment and one of the agile coaches stated that although continuous integration and automated test tools and methods are currently available, software development productivity will not improve until more software development teams use the tools.

Configuration management and engineering management: those two go hand-in-hand. Once again I can't say it enough, continuous integration and automated test. The more these come together and in sync will greatly improve software quality and improve it. By utilizing... the tools...

Future Use of Analytics – SWEBOK Activity by Theme. The research participants primarily discussed how analytics could be used in the future during the engineering management activities (16%), the requirements activities (16%), and the tools and methods activities (16%) as shown in Table 23. What the research participants

said about the use of analytics during the tools and methods activities was discussed previously in this dissertation.

The research participants also talked about how analytics could be used in the future during the quality activities (12%), testing activities (12%), and the process activities (9%). For example, a software manager talked about measuring the time a software developer spends problem solving versus writing code as a way to optimize development time in the future.

Then on the process side even more, helping... I'll give you a great example.

You have a developer who gets stuck on a problem and they chase that problem for way too long before they realize that they are losing ground in terms of actually getting things done productivity wise.

A few research participants also discussed how analytics could be used in the future during the configuration management (7%), design (5%), maintenance (5%) and construction (2%) activities.

Table 23

Future Use of Analytics – SWEBOK Activity by Theme

SWEBOK Activities	Descriptive	Diagnostic	Predictive	Prescriptive	Total
Engineering Management	7%	2%	5%	2%	16%
Requirements	7%	5%	5%	0%	16%
Tools and Methods	5%	5%	5%	2%	16%
Quality	7%	2%	2%	0%	12%
Testing	7%	5%	0%	0%	12%
Process	5%	2%	2%	0%	9%
Configuration Management	0%	2%	5%	0%	7%
Design	0%	0%	0%	5%	5%
Maintenance	0%	2%	2%	0%	5%
Construction	2%	0%	0%	0%	2%
Grand Total	40%	26%	26%	9%	100%

Future Use of Knowledge Management – People. The agile coaches, project managers, and software managers said that people needed to know how to use KM effectively to improve software development productivity (56%). People need to know how to do their jobs. People need knowledge they can understand and people need knowledge when they need it. Most importantly, people need to know how to communicate and collaborate.

Collaboration is key on any process whether you use lean or Kanban or Scrum or XP. It's that collaboration that is key. It's also one of the things that points out why software projects fail; there is no collaboration. Management doesn't support it. Stuff like that. You've got a get involved to make it successful.

Future Use of Knowledge Management – Process. The themes that emerged from the data were analyzed to determine how knowledge accumulation, creation, retention, and transfer could be used in the future to improve software development

productivity. The themes were analyzed to identify the differences and similarities between what each of the research participant groups said about the use of KM in the future. The data was analyzed to determine what the research participants said about the use of KM in the future in each of the SWEBOK activities.

Future Use of KM by KM Activity. The research participants talked more about the use of knowledge transfer (50%) and knowledge creation (31%) than knowledge retention (13%) or knowledge accumulation (6%) when they talked about the use of KM to improve software development productivity. The research participants talked about how knowledge transfer could be used in the future to improve decision making (19%). Table 24 shows the KM themes by knowledge process.

Table 24

Future Use of Knowledge Management

Categories	Accumulation	Creation	Retention	Transfer	Total
Decision Making	6%	6%	6%	19%	38%
Communication	0%	6%	6%	3%	16%
Monitor and Adapt	0%	6%	0%	6%	13%
Quality	0%	6%	0%	6%	13%
Risk	0%	0%	0%	6%	6%
Transition from Development to Release	0%	0%	0%	6%	6%
Grand Total	6%	31%	13%	50%	100%

Future Use of KM – Category by Role. The future use of KM was compared by role. The agile coaches (38%) and the project managers (38%) talked more about how KM could be used in the future to improve software development productivity than the software managers (25%). Agile coaches, project managers, and software managers said

that KM could be used in the future to improve decision making (38%). The software managers said that KM could be used in the future to improve communication (16%). The project managers said that KM could be used in the future to monitor and adapt (13%), to improve quality (13%), and to transition from development to release (6%). The KM themes identified by the agile coaches are shown in Table 25. The analytic themes identified by the project managers are shown in Table 26 and the analytic themes identified by the software managers are shown in Table 27.

Table 25

Future Use of KM - Category by Agile Coaches

Categories	Accumulation	Creation	Retention	Transfer	Total
Decision Making	6%	6%	6%	13%	31%
Risk	0%	0%	0%	6%	6%
Communication	0%	0%	0%	0%	0%
Monitor and Adapt	0%	0%	0%	0%	0%
Quality	0%	0%	0%	0%	0%
Transition from Development to Release	0%	0%	0%	0%	0%
Grand Total	6%	6%	6%	19%	38%

Table 26

Future Use of KM - Category by Project Managers

Categories	Accumulation	Creation	Retention	Transfer	Total
Monitor and Adapt	0%	6%	0%	6%	13%
Quality	0%	6%	0%	6%	13%
Decision Making	0%	0%	0%	6%	6%
Transition from Development to Release	0%	0%	0%	6%	6%
Communication	0%	0%	0%	0%	0%
Risk	0%	0%	0%	0%	0%
Grand Total	0%	13%	0%	25%	38%

Table 27

Future Use of KM - Category by Software Managers

Categories	Accumulation	Creation	Retention	Transfer	Total
Communication	0%	6%	6%	6%	19%
Decision Making	0%	6%	0%	0%	6%
Monitor and Adapt	0%	0%	0%	0%	0%
Quality	0%	0%	0%	0%	0%
Risk	0%	0%	0%	0%	0%
Transition from Development to Release	0%	0%	0%	0%	0%
Grand Total	0%	13%	6%	6%	25%

Future Use of KM - Themes by Category. The themes emerged from an analysis of the data on how agile coaches, project managers, and software managers could use KM in the future. Although the research participants primarily discussed how they use KM could be used to improve decision making and communication, all of the themes that emerged from an analysis of the data are described in this dissertation. For example, although only one project manager talked about how KM could be used in the future to transition from development to release, the theme was included in the results.

Decision Making. An agile coach and a project manager said that knowledge creation and transfer could be used in the future to determine what gets built and another agile coach said that knowledge accumulation, retention, and transfer could be used in the future to determine what gets built.

In the work I see in the work I've been involved with there is definitely a need for improvement in the requirements and user stories. Typically, in a user story "As a, I want to, so that", that is a user story and that goes in the product backlog and that's handed to the team but there are several things that can help strengthen that user story to make it able for a team to understand exactly what it is that needs to be done to satisfy that user story. Those are some of the other techniques of including acceptance criteria or even behavior driven development. ...Additions to the user story. There's ways to improve it even just beyond the shell so to speak.

Communication. A software manager said that knowledge transfer could be used in the future to improve communication between stakeholders and to capture knowledge

just in time while another software manager said that knowledge creation and retention could be used in the future to improve communication between stakeholders and to integrate knowledge into the code.

So it's really presenting information in a way that people can see and will take the time to look at and so for example, it would be pretty interesting if you could shoot back information to somebody's phone. Everybody's tethered to their phones these days and they are used to looking at it. When you're sitting in a meeting it's kind of hard to get people to look up from their phones. So I think there a lot of areas where delivering information to stakeholders in ways they can understand and will look at is important.

Other Categories. Few research participants talked about the use of KM to monitor and adapt, to improve quality, to manage risk or to transition from development to release. A project manager said that knowledge creation and transfer could be used in the future to improve knowledge about project status. A project manager said that knowledge creation and transfer could be used in the future to improve quality. An agile coach said that knowledge transfer could be used in the future to manager risk and a project manager said that knowledge transfer could be used to transition from development to release.

Future Use of Knowledge Management - Tools and Methods. The agile coaches, project managers, and software managers did not discuss how quantitative methods could be used in the future to measure software development progress; however, the project managers and software managers claimed that KM tools and methods could

be used to improve software development productivity if the knowledge was understandable to all stakeholders, up to date, and synchronized with the software development.

Right. I think one of the biggest challenges of any of those software things is that separation of code from kind of human understandable language. When it is happening, it seems that they naturally become separated. Okay, we have our code so all of the engineers can look at the code and know what's going on and know what should happen and then describing back to the users... If someone says, "How does the payment processing work?" I can't give them the payment processing class and there is a function in here called process payment and that describes it, right? So how do we build in or find translation tools to make those pieces of the communication and knowledge management, you know, more holistic?...How do we make the requirement and the documentation of the system and the code of the system, one?

One of the project managers recommended that KM tools and methods could be used to improve software development productivity if software teams actively trained the configuration and maintenance teams for a seamless transition from development to release.

So that knowledge management of the system at a high level, deploying it, configuring it for release, those types of things as well as all of the diagnostic tools of the software. You know, for instance, logging and performance monitoring. Those types of things. Knowledge management is critical there and

that's where typically if you are transitioning to a support team you're not just going to dump. "Here's the wiki, have fun." Figure it out for yourself. There has to be some official knowledge transfer process they kind of falls outside of anything agile gives you guidance for.

Future Use of KM – SWEBOK Activity by KM Activity. The research participants primarily discussed how KM could be used in the future during the process activities (50%) and the research participants discussed how KM could be used in the future during the requirements activities (25%) and the tools and methods activities (25%) as shown in Table 28. What the research participants said about the use of KM during the Tools and Methods activities is discussed later in this dissertation. One of the software managers discussed how KM could be used in the future to improve software development productivity by delivering information to the stakeholders' phones.

So it's really presenting information in a way that people can see and will take the time to look at and so for example, it would be pretty interesting if you could shoot back information to somebody's phone. Everybody's tethered to their phones these days and they are used to looking at it. When you're sitting in a meeting it's kind of hard to get people to look up from their phones. So I think there a lot of areas where delivering information to stakeholders in ways they can understand and will look at is important.

Table 28

Future Use of KM – SWEBOK Activity by KM Activity

SWEBOK Activities	Accumulation	Creation	Retention	Transfer	Total
Requirements	3%	10%	7%	10%	31%
Process	0%	7%	3%	10%	21%
Tools and Methods	0%	7%	3%	7%	17%
Configuration Management	0%	3%	0%	7%	10%
Construction	0%	3%	0%	3%	7%
Maintenance	0%	3%	0%	3%	7%
Quality	0%	3%	0%	3%	7%
Design	0%	0%	0%	0%	0%
Engineering Management	0%	0%	0%	0%	0%
Testing	0%	0%	0%	0%	0%
Grand Total	3%	38%	14%	45%	100%

Comparison of Themes by Research Participant Demographics

The research participant responses were analyzed to determine the similarities and differences of their responses to the research questions based on demographics. The responses were compared by number of years of agile software development experience. The responses were compared based on the agile software development methods used by the research participants and the responses were compared based on the size of the software development projects described by the research participants.

Analytics and Agile Experience. Approximately 78% of the research participants had over 5 years of agile software development experience. Unlike the research participants with fewer than 5 years of experience (22%), the research participants with more than 5 years of experience said that analytics are used to improve security. The research participants with fewer than 5 years of agile software development experience identified three unique themes as show in Table 29.

Table 29

Analytic Themes Unique To Research Participants With <5 Years of Experience

Category	Theme
Estimate, Plan, Forecast	Analytics are used to groom the backlog
Decision Making	Analytics are used to determine maintenance priorities
Risk Management	Analytics are used to determine the root cause of issues

Knowledge Management and Agile Experience. Most of the research participants had more than five years of experience using agile software development methods (78%). The research participants with less than 5 years of agile software development experience included one project manager and one software manager. They did not identify any unique KM themes compared to the KM themes identified by the research participants with over 5 years of agile software development experience.

Analytics and Project Description. The themes that emerged from the data were compared based on the project description provided by the research participants. The projects were categorized as small, medium or large based on the project description provided by the research participants and the estimated number of employees at the research participant's organization. Projects were categorized as small if the number of employees was 500 or less. The project was categorized a medium if the number of employees was greater than 500 but less than 50k. The project was categorized as large if

the number of employees was greater than 50k. One research participant described their projects as small, medium, and large.

Four research participants described their projects as small, four research participants described their projects as medium, and three research participants described their projects as large. One research participant described their project as small, medium, and large. The themes were analyzed to identify the unique themes for each project size. Only one research participant who described their project as large discussed the use of analytics to improve security and only one research participant who described their project as large talked about the use of analytics to manage risk. The remaining themes were not unique to the category. Table 30 shows the analytic themes discussed by the research participants unique to each project size.

Table 30

Analytic Themes Discussed by Research Participants – Unique for Project Size

Category	Theme	Size
Decision Making	Analytics are used to determine maintenance priorities	Small
Estimate, Plan, Forecast	Analytics are used to groom the backlog	Large
	Analytics could be used in the future to forecast	Small
Inspect and Adapt	Analytics could be used in the future to measure change in scope over time	Medium
	Analytics could be used in the future to measure cost for delays	Medium
Risk	Analytics are used to determine the root cause of issues	Large
Security	Analytics are used to improve security	Large
	Analytics could be used to improve security	Large
Transition from Development to Release	Analytics could be used to improve the transition from development to release	Small

Knowledge Management and Project Description. The research participants who described their projects as small, medium, and large did not discuss categories related to the use of KM to improve software development productivity that were unique. Instead, the research participants discussed themes that supported the categories identified by the other research participants. For example, the research participants who described their projects as small discussed how KM is used to inspect and adapt as did other the research participants. Table 31 shows the unique KM themes discussed by the research participants for each project size.

Table 31

KM Themes Discussed by Research Participants – Unique for Project Size

Category	Theme	Project Size
Communication	KM could be used in the future to integrate knowledge into the code	Small
	KM could be used in the future to capture knowledge just-in-time	Medium
	KM could be used in the future to improve communication between stakeholders	Medium
Continuous Improvement	KM is used to drive change across the organization	Large
Decision Making	KM is used as a competitive advantage	Medium
Monitor and Adapt	KM could be used in the future to improve knowledge about project status	Small
Risk	KM could be used in the future to manage risk	Small

Analytics and Agile Practices. The research participants stated that they used Scrum (44%), Scrum and Lean\Kanban (22%), Scrum, Lean\Kanban, XP (22%), and Lean\Kanban (11%). No unique categories related to how analytics are used to improve software development productivity emerged from the data provided by the 11% of the research study participants who stated that they did not use Scrum methods for software development. Although one unique analytic theme was discussed by the research participants who stated that they used XP (22%) in addition to Scrum and other agile software development methods; this theme was not unique within the category of

estimate, plan, forecast since research participants who do not use XP discussed themes related to estimate, plan, forecast. No unique Analytic categories emerged from the data provided by the research participants who claimed to use Scrum (89%). Table 32 shows the theme unique to the research participants who use XP.

Table 32

Analytic Themes Discussed by Research Participants Using XP

Category	Theme
Estimate, Plan, Forecast	Analytics could be used in the future to forecast

KM and Agile Practices. No unique themes related to how KM is used to improve software development productivity emerged from the data provided by the 11% of the research study participants who stated that they used Lean\Kanban rather than Scrum methods for software development. No unique KM categories emerged from the data provided by the research participants who claimed to use Scrum (89%).

Research Question 4

The research participants were asked, “What obstacles do software managers, project managers, and agile coaches in agile software environments think their organization need to overcome to improve software development productivity?”

Obstacles - People. The agile coaches, project managers, and software managers said that the stakeholders do not understand agile software development practices well enough to make informed decisions. For example, the Software Managers said that senior management interrupts the software development teams during an iteration to add or change the work because the management does not understand agile well enough. The

project managers said that confusion and conflict results when the stakeholders and the software development team do not have the same understanding of agile software development practices. Although productivity could be improved if the stakeholders had a better understanding of agile software development practices, the agile coaches stated that the stakeholders do not always want to know more about agile software development.

So for software managers and project managers it's explaining how they do what they do lightly enough to people who don't have an interest deeply enough so that they understand the implications of the decisions that they can make from their role within an organization.

The agile coaches also said that there is a lack of project management skills and the software teams do not have a thorough understanding of agile software development practices; consequently, only some agile practices are used which limits the benefits derived from using agile software development methodology. According to one software manager, productivity could be improved if software teams had alternative ways to learn new processes, tools, and technology.

The project managers recommended that software development teams improve the communication and collaboration with the infrastructure team to remove the obstacles to implementation, which cause confusion and conflict.

There are groups outside of the dome of the engineering team which aren't quite there yet and that is where the obstacles still exist or the challenges still exist. We talked about it in several different ways. The way that I see that they could be

improved which is visibility and a common description of those objects, if you will, that are shared between them.

Obstacles - Process. The agile coaches said that the organizations' command-and-control structures conflict with the use of agile software development practices. According to the project managers, there is a culture clash between the iterative process of agile software development, which is not focused on long range planning and the business need for long range planning. The software managers said that the corporate culture does not support agile software development.

For software managers and agile coaches I think the biggest hurdle is corporate culture and that's true of really anything but even more so in software. It hugely defines whether someone can be agile and innovative or whether, you know, you are just coasting and putting things out because someone up the chain decided, "Hey, we should do this."

The project managers and the software managers said that the software developed does not solve the right problem because there is insufficient understanding of the business needs and because the requirements are not clearly defined.

One of our biggest challenges is really nailing a requirement and we have seen this time and time again.

Productivity is negatively affected when there is no software product roadmap according to one software manager and aligning the software development plan with the organizational goals could improve productivity.

Really making sure that this is what we think we need to do, this is what we can do, and then this is what we're going to do.

One of the project managers said that when organizations try to impose a single software development process on all software development teams, productivity is negatively impacted.

We had a guy a year or so ago tried to write the software development lifecycle manual and we realized in the process of doing that, because we have so many different teams that have different end-user business needs and they operate in different languages. The applications are different sizes. This one small little lightweight app they could probably even deploy every week. The other one that is our e-commerce site, it takes a week to regression test. So, you can't be apples to apples on different teams forcing them to do things the same way because they are just different scales.

Obstacles – Tools and Methods. The agile coaches said that tools like Excel are not sufficient for agile software development because they do not scale and do not enable the team to forecast and the software managers said that the software development and management tools are not mature and that the software teams do not know how to use the agile tools.

Summary

The results of this phenomenological research study were discussed in this chapter. The research questions were answered and evidence was provided to support the findings. The agile coaches, project managers, and software managers who were

interviewed said that DDD is needed to improve software development productivity; however, productivity should not be improved at the expense of quality. The research participants discussed the need to include DDD in the software development process and to measure the right thing. The research participants recommended that DDD be used to determine what should be built because productivity is negatively impacted when the wrong thing is built and the research participants recommended that DDD be used to ensure the stakeholders have a common understanding of agile software development methods.

In Chapter 5, the research findings discussed in this chapter are interpreted based on the conceptual framework: people, process, and tools and methods. Recommendations for additional research are presented and the limitations of this research study are presented. Finally, the implications for social change are discussed and the conclusions are discussed.

Chapter 5: Discussion, Conclusions, and Recommendations

Software project success needs to improve (Ambler, 2012; Emam & Koru, 2008; Mieritz, 2012; the Standish Group (n. d.) and agile software development methods were developed to improve software project success (Rao, Naidu, & Chakka, 2011). Although DDD can improve organizational output and productivity, organizations need to define DDD within the context of the problem (Ferrand et al., 2010; Herschel et al. 2010; Yeoh & Koronios, 2010). Based on a review of the literature on DDD and agile software development, the research on DDD as a tool to improve software development productivity is in the initial stages; therefore, this qualitative research study was intended to fill this gap in the literature by exploring the meaning of DDD within an agile software development environment.

The purpose of this phenomenological research study was to understand the lived experiences of agile coaches', project managers', and software managers' use of DDD in agile software organizations as a tool to improve software development productivity. The purpose was to identify impediments to DDD use in software development organizations and to make recommendations for improving DDD use in software development organizations based on the findings from this research study and a review of the literature.

The IPA approach was used to iteratively analyze the data for this research study. Approximately 19 themes in eight categories emerged from an analysis of the data on the current and future use of analytics to improve software development productivity and

approximately 26 themes in seven categories emerged from an analysis of the data on the current and future use of KM to improve software development productivity.

Software development productivity may be improved if organizations use DDD to determine what to build and how to build it. Software development productivity may be improved if organizations use DDD to transition from a command and control culture to a culture that supports agile software development. Based on the results of this research, software development productivity may be improved if organizations use DDD to ensure the stakeholders have a common understanding of agile software development methods.

Interpretation of the Findings

Software development productivity needs to improve (Ambler, 2012; Emam & Koru, 2008; Mieritz, 2012; the Standish Group (n. d.) and agile software development methods were developed to improve software development productivity (Schwaber, 1995). Most of the research participants said that analytics and KM could be used to improve software development productivity; however, productivity should be improved by focusing on building the right thing, rather than focusing on increasing the number of lines of code written per hour. Although a few of the research participants cautioned that a balance needs to be maintained between productivity and quality, other research participants equated improved quality with improved productivity. If quality is improved, less rework is needed to meet customer expectations and to maintain the software.

People

Organizations need to adapt a culture of sharing for successful agile software development and organizations need to adapt a culture of sharing to successfully create and transfer active knowledge (Sholla & Nazari, 2011). Organizations need to explore ways to use analytics. Although Brynjolfsson et al. (2011) found that DDD improved organizational productivity, organizations need to brainstorm ways to use DDD to improve software development productivity (Adrian & Genovese, 2011). Agile software organizations need to explore ways to use DDD because people need to know more about what to measure and how to measure productivity in an agile software development environment.

Agile software development teams need to be trained to use the agile software development tools and agile software development teams and their stakeholders need alternative ways to learn new processes, tools, and methodologies. For example, one research participant recommended that the agile software development teams consider using mobile technology to provide knowledge to stakeholders when and where they need it. The knowledge about agile software development practices needs to be tailored to the needs of the stakeholders. For example, some of the research participants said that the stakeholders do not always want to know more about agile software development. The stakeholders need to know enough about agile software development practices to make informed decisions. The agile software development teams need to know the benefits and disadvantages of each of the agile software development methods in order to

select the appropriate agile methods for each project. Roa et al. (2011) were able to identify the pros and cons of XP, DSDM, and Scrum.

In addition to training the software development team, the stakeholders need to understand agile software development practices to avoid negative impacts to productivity such as interruptions. Productivity can be improved by reducing interruptions and by improving the quality of the software produced. Software developers can use historical data to improve software estimation and to reduce defects (Elminir et al., 2009).

Process

Communication. Most of the research participants talked about how knowledge transfer is used to store and find content and to improve communication between stakeholders just as Ceschi et al. (2005) found, productivity was improved when agile software communication methods and knowledge transfer methods were used instead of traditional software development methods. Although the research participants did not discuss the differences between using agile software development methods on small projects versus large projects and although Qumer and Hendersen-Sellers (2008) claimed that agile methods could be used in large and small software projects, managers should be aware of the exponential increase in communication channels as team size increases (Lalsing et al., 2008; Pikkarainen et al., 2008; Rao et al., 2011).

One of the research participants promoted the benefits of face-to-face communication on agile software development projects; however, organizations that choose to use automated methods to develop and display story cards may not benefit

from the social benefits of face-to-face communication (Sharp et al., 2009). A few of the research participants said that knowledge accumulation, creation, retention, and transfer are used to share agile expertise and one of the research participants said that KM could be used in the future to capture knowledge just-in-time and to integrate knowledge into the code.

Continuous Improvement. The research participants said that descriptive and diagnostic analytics are currently used for continuous improvement and that analytics could be used in the future to improve coding productivity. A few of the research participants talked about how KM is used during the agile retrospective process which was one of the six KM activities Levy and Hazzan (2009) recommended to improve agile software development. Organizations should integrate KM activities into the agile software development continuous improvement processes (Levy & Hazzan, 2009). KM is used to drive change across the organization according to one research participant.

Decision Making. Descriptive analytics are currently used to determine maintenance priorities according to one of the research participants. The research participants recommended that organizations use advanced analytics to determine what should be built and how to build the software products just as Laney (2012) recommended that organizations take advantage of advanced analytics to evolve from insight to foresight and to embrace complexity and changing conditions. Therefore, organizations will need to take advantage of big data and explore ways to use advanced analytics.

Several of the research participants talked about how knowledge transfer is currently used as a tool to determine what should be built and how it should be built and how knowledge transfer could be used in the future to determine what should be built and how it should be built. However, agile software development teams need to aware that increased knowledge transfer may or may not lead to the correct solution. If stakeholders are like CTOs, they may use heuristics to make decisions when time, knowledge, and computational power are limited. Additional research may be needed to determine how decision makers consider, weigh, and integrate data for decision-making (Ow & Morris, 2010).

The research participants also discussed how KM is used to improve team decision making, to select the right tool for the job, and as a competitive advantage. However, agile software development teams need to aware that knowledge needs to be transformed so that it is actionable (Linden et al., 2007; Lingling et al., 2009). More data does not lead to better decision-making and improved productivity unless the data is analyzed, formatted, and presented in a way that enables the decision makers to make better decisions. Although the research participants discussed how KM is used to improve team decision making, agile software development teams need to balance team cohesion and team empowerment to avoid dysfunctional consensus in which all the group members either silently disagree with a solution or all of the members agree because of one person who is perceived as an influencer (McAvoy & Butler, 2009).

Estimate, Plan, Forecast. A few of the research participants mentioned that descriptive analytics are currently used to determine how much time has been spent on

the software design versus how much time has been spent on software construction and to groom the backlog. Several of the research participants said that currently descriptive, diagnostic, and predictive analytics are used to estimate, plan, and forecast software development iterations and releases; however, a few of the research participants said that more advanced analytics could be used in the future to estimate, plan, and forecast. For example, one of the research participants talked about the use of advanced analytics to do long-term planning:

This is really where I'm trying to take my company from a prescriptive perspective. This is where you're getting into forecasting models. It could be perhaps. From a prescriptive perspective you're really trying to predict what is your future timeline for multiple releases across the project, right? What do we anticipate our burn to be?

Although Abouelela and Benedicenti (2010) were able to successfully estimate the completion date and the defect rate of an agile software development project using a Bayesian network, their research was limited to two case studies and agile XP methods and although Zare and Akhavan (2009) found that their fuzzy cyclic network algorithm was more accurate than the schedule based on the critical path method (CPM), their research was limited to estimating software development in a traditional software development environment. Additional research is needed to determine how advanced analytics could be used to estimate, plan, and forecast agile software development projects. Contrary to what the project managers said about the need for long term estimating, planning, and forecasting in order to meet the business needs, Pelrine (2011)

proposed that agile software development organizations use the inspect and adapt model to estimate by establishing system boundaries and then adapting as more is learned about the evolving system.

Monitor and Adapt. Several of the research participants talked about how they currently use descriptive and diagnostic analytics to monitor progress and adapt the software development plan or “inspect and adapt.” Only one software manager recommended that descriptive analytics be used in the future to measure the change in scope over time and to measure the cost for delays in order to improve software development productivity. A few of the research participants talked about how knowledge creation and retention could be used to improve knowledge about project status and to monitor progress and adapt the software development plan.

Quality. Several of the research participants said that descriptive and diagnostic analytics are currently used to improve quality and one of the research participants said that predictive analytics could be used in the future to improve quality. Several of the research participants discussed how KM is currently used to improve code quality and how KM could be used in the future to improve code quality. A research study seems to support this claim. When software engineers were provided with historical data, although they were not able to improve productivity, they were able to improve the quality of their work (Elminir et al, 2009).

Quality may also be improved when agile software development teams use the XP practice of paired programming. Paired programming promotes knowledge transfer and as Balijepally et al. (2009) found, while paired programming methods did not improve

performance, paired programming did improve software quality. However, Lee and Xia (2010) recommended that agile software managers balance software team autonomy and diversity to successfully deliver the functionality that meets the customer expectations for quality, cost, and schedule because agile software requirement changes can have both positive and negative effects on on-time completion and on-budget completion.

Risk. One of the research participants discussed how descriptive analytics are currently used to manage risk by determining the root cause of issues. Although Laney (2012) recommended that analytics be used to automate risk reporting, no other articles were found in the literature that discussed analytics as a tool to manage risk in an agile software development environment. Most of the research participants talked about the use of KM to manage risk by minimizing the negative impact of personnel changes. However, agile software development teams need to find ways to train new employees rather than have more experienced team members take the time to train less expert team members which can negatively impact productivity (Neves et al., 2011).

Security. Only one of the research participants discussed how diagnostic analytics are currently used to improve security during software development and that diagnostic analytics could be used in the future to improve security.

Transition from Development to Release. The majority of the research participants said that descriptive and diagnostic analytics are currently used to transition from software development to release by using automated test and continuous integration methods. The research participants also said that descriptive and diagnostic analytics could be used in the future to transition from software development to release while one

research participant said that predictive analytics could be used to transition from development to release. Although Smith (2011) recommended using automated regression testing and continuous integration to release software frequently for cloud computing, no empirical research was found to validate this recommendation. A few of the research participants said that KM is used to improve the transition from development to release and that KM could be used in the future to improve the transition from development to release.

Process. According to Linden et al. (2007) organizations could improve software development by designing software systems on Churchman's inquiry system design characteristics, which would provide a framework that is generalizable and repeatable. Data would be needed to determine the differences between the user's behavior patterns and data would be needed to estimate how well the user's behavior met the overall system goals. Data would also be needed to communicate the goals to the software development team so that the information system design reflected the goals and data would be needed to ensure the integrity of the whole system was maintained.

Tools and Methods

Some of the software managers said that software development and management tools needed to mature. Software productivity may improve when the software tools mature because software development productivity is dependent upon people, process, and tools (Wadhwa & Mitra, n.d.). Some of the research participants said that agile software development teams need to use tools that scale and software estimating,

scheduling, and management tools need to be improved and the techniques need to be improved (Emam & Koru, 2008; Patil, Nageswara, & Yogi, 2011).

Organizations may benefit by implementing KM tools that focus on skill management and people to minimize entry cost and increase visibility to KM. Organizations could also benefit by tailoring KM tools to provide the knowledge needed by team members other than management (Sholla & Nazari, 2011). Several of the research participants talked about the use of KM to improve communication within the software development team and between stakeholders. Although Rayner (2011) and Chandler (2011) recommended that organizations use CDM to improve decision making, communication and collaboration, the research participants did not mention using CDM platforms for decision making nor did they recommend using CDM platforms in the future. Agile software development teams may need to aware of the benefits and limitations of CDM as well as the tools that enable CDM. Agile software teams need to know that CDM platforms are best used for “nonroutine, complex decisions that require iterative human interactions” (Schlegel et al., 2009, p. 1).

Limitations of the Study

Although the research participants were provided with definitions for descriptive, diagnostic, predictive, and prescriptive analytics, and knowledge accumulation, creation, retention, and transfer, the responses to the research questions were limited to the research participants’ understanding of the use of analytics and KM in an agile software development. This research study was limited to interpreting the lived experiences of software managers, project managers, and agile coaches in the United States who use

agile software development methods and did not attempt to provide empirical evidence that analytics or KM improve software development productivity. The research questions were answered by research participants, who agreed that software development productivity needed to improve, and that analytics and KM could help improve software development productivity. The responses to the research questions may differ for those who do not agree that software development productivity needs to improve or that analytics and KM could help improve software development productivity.

Recommendations

This qualitative research study was intended to explore the use of DDD, which includes data, analytics, and KM, as a tool to improve software development productivity in an agile software development environment. This research study was limited to understanding the lived experiences of nine individuals who work for organizations in the United States and this research study was limited to understanding the lived experiences of individuals with knowledge creation, accumulation, retention, and transfer and descriptive, diagnostic, predictive, and prescriptive analytics. Additional research outside the United States or with different operational definitions for KM and analytics may reveal new insights into how DDD could be used to improve software development productivity.

This research study was limited to understanding the lived experiences of agile coaches, software managers, and projects managers. Additional research that included software developers, business analysts, and other agile software development team members could reveal different results. Although some of the research participants

described their projects as small, others described their projects as medium or large.

Additional research that purposefully selected research participants based on project size could add to the knowledge on how DDD improves software development productivity in an agile software development environment.

Qualitative research is exploratory in nature. Additional research is needed to empirically determine the correlation between DDD and software development productivity in an agile software development environment. Quantitative research methods could be used to determine how much productivity is or is not improved when DDD is used and mixed methods research methods could be used to better understand agile software development environments and to measure their effectiveness on the use of DDD to improve software development productivity. Additional research could be conducted to evaluate the effectiveness of KM tools that incorporate analytics to improve software development productivity.

Implications

Organizations in the United States and elsewhere have spent time and money developing software products that have failed to meet customer expectations and have failed to take advantage of advances in hardware capabilities. Organizations may be able to use DDD to improve software development productivity, which would result in improved customer satisfaction, opportunities to develop new products and features, and the potential to spend time and money on additional projects. Positive social change could result from organizations that are better able to compete in a global economy and from organizations that are better able to create products and jobs.

Organizations should consider transforming from a command and control culture to a culture that supports agile software development and organizations should encourage stakeholders at all levels of the organization to learn enough about agile software development methods to make informed decisions. Organization should explore ways to use DDD to determine what should be built and software development organizations should explore ways to use DDD to improve software development productivity.

Conclusion

This phenomenological qualitative research study explored the lived experiences of agile coaches, project managers, and software managers with DDD as a tool to improve software development productivity in an agile software development environment. Just as DDD was found to improve organizational productivity (Brynjolfsson et al., 2011), the research participants agreed that DDD has the potential to improve software development productivity in agile software development organizations. Although agile software development methods were developed to improve software development productivity (Schwaber, 1995), the research participants talked about the need for organizations to consider the unique characteristics of each project and ensure that the people, process, tools, and methods are aligned to meet the goals of the organization.

This qualitative research study explored the use of DDD, which consists of data, analytics, and KM to improve software development productivity. The cost for software project failure is high and the benefits for improved software development productivity are significant. Therefore, based on the results of this research study, organizations

should explore ways to use more advanced analytics to better ensure the right product is built and to work collaboratively with agile software development organizations throughout the software development process and organizations should find ways to use KM to improve communication and collaboration with agile software development teams and to make knowledge actionable.

References

- Abdullah, R., Shah, Z. M., & Talib, A. M. (2011a). A framework of tools for managing software architecture knowledge. *Computer & Information Science*, 4(2), 2-16. Retrieved from <http://journal.ccsenet.org/index.php/cis/index>
- Abdullah, R., Talib, A. M., & Misran, E. K. (2011b). Agent technology application strategies in personal and team software process environment. *American Journal of Economics & Business Administration*, 3(2), 347-351. Retrieved from <http://thescipub.com/ajeba.toc>
- Abouelela, M., & Benedicenti, L. (2010). Bayesian network based XP process modeling. *International Journal of Software Engineering & Applications*, 1(3), 1-15. doi:10.5121/ijsea.2010.1301
- Adrian, M., & Genovese, Y. (2011, May 2). Analytics and learning technology: CIOs, CTOs should rethink art of the possible. 1-7. Retrieved from <http://www.gartner.com/technology/home.jsp>
- AlAli, A. I., & Issa, A. A. (2011). Towards well documented and designed agile software development. *World Academy Of Science, Engineering & Technology*, 73, 73126-73131. Retrieved from <http://www.waset.org/>
- Ambler, S. W. (2012). Feature driven development and agile modeling. Retrieved from <http://www.agilemodeling.com/essays/fdd.htm>
- Ambler, S. W. (2010). 2010 IT project success rates. Retrieved from <http://www.drdoobs.com/architecture-and-design/2010-it-project-success-rates/226500046?pgno=2>

- Amescua, A. A., Bermón, L. L., García, J. J., & Sánchez-Segura, M. I. (2010). Knowledge repository to improve agile development processes learning. *IET Software*, 4 (6), 434-444. doi:10.1049/iet-sen.2010.0067
- Babbie, E. R. (2006). *The practice of social research* (11th ed.). Belmont, CA: Wadsworth.
- Balijepally, V., Mahapatra, R., Nerur, S., & Price, K. H. (2009). Are two heads better than one for software development? The productivity paradox of pair programming. *MIS Quarterly*, 33(1), 91-118. Retrieved from <http://www.misq.org/>
- Ballou, M. (2008). Key disruptive trends driving agile adoption. Retrieved from <http://www.qsma.com/pdfs/AgileMetricsQSMASStudy.pdf>
- Bartes, F. (2011). Action plan - basis of competitive intelligence activities. *Economics & Management*, 16664-16669. Retrieved from <http://www.ktu.lt/lt/mokslas/zurnalai/ekovad/16/1822-6515-2011-0664.pdf>
- Bassi, L. (2011). Raging debates in HR analytics. *People & Strategy*, 34(2), 14-18. Retrieved from <http://www.hrps.org/>
- Bazeley, P. (2013). *Qualitative data analysis: Practical strategies*. Thousand Oaks, CA: Sage.
- Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., . . . Thomas, D. (2001). The agile manifesto. Retrieved from www.agilemanifesto.org
- Bjornson, F., & Dingsoyr, T. (March 11, 2008). Knowledge management in software engineering: A systematic review of studied concepts, findings and research

- methods. *Information and Software Technology*, 14. Retrieved from <http://alarcos.esi.uclm.es/doc/metotecinfinf/articulos/bjornson.pdf>
- Boden, A., Avram, G., Bannon, L., & Wulf, V. (2009). *Knowledge Management in distributed software development teams: Does culture matter?* Paper presented at the IEEE 4th International Conference on Global Software Engineering (ICGSE'09), Limerick, Ireland.
- Boehm, B., Lane, J., Koolmanojwong, S., & Turner, R. (2010). Architected agile solutions for software-reliant systems. In T. Dingsøyr, T. Dybå & N. B. Moe (Eds.), *Agile Software Development* (pp. 165-184): Berlin: Springer.
- Brynjolfsson, E., Hitt, L. M., & Kim, H. H. (2011). Strength in numbers: How does data-driven decision making affect firm performance? *SSRN eLibrary*. Retrieved from <http://ssrn.com/paper=1819486>
- Cappelli, W., & Kowall, J. (2011). Magic quadrant for application performance. 46. Retrieved from <http://www.gartner.com/technology/core/home.jsp>
- Ceschi, M., Sillitti, A., Succi, G., & De Panfilis, S. (2005). Project management in plan-based and agile companies. *Software, IEEE*, 22(3), 21-27.
doi:10.1109/ms.2005.75
- Chan, F. K. Y., & Thong, J. Y. L. (2009). Acceptance of agile methodologies: A critical review and conceptual framework. *Decision Support Systems*, 46(4), 803-814.
doi:10.1016/j.dss.2008.11.009
- Chandler, N. (2011). Hype cycle for performance management. 64. Retrieved from <http://www.gartner.com/technology/core/home.jsp>

- Chandler, N., Hostmann, W., Rayner, N., & Herschel, G. (2011). Gartner's business analytics framework. 18. Retrieved from <http://www.gartner.com/technology/core/home.jsp>
- Chen, S. (1998). *Mastering reasearch: A guide to the methods of social and behavioral sciences*. Chicago, IL: Nelson-Hall.
- Clark, W., & King, M. (2011). Magic quadrant for mobile consumer application platforms. 39. Retrieved from <http://www.gartner.com/technology/core/home.jsp>
- Clutterbuck, P., Rowlands, T., & Seamons, O. (2009). A case study of SME web application development effectiveness via agile methods. *Electronic Journal of Information Systems Evaluation*, 12(1), 13-26. Retrieved from <http://www.ejise.com/main.html>
- CMMI Product Team (2010). CMMI for development, Version 1.3. Retrieved from <http://www.sei.cmu.edu/library/abstracts/reports/10tr033.cfm>
- Cohn, M.. (n.d., a). What is agile and Scrum? In Mountain goat software. Retrieved June 3, 2012, from [http://www.mountaingoatsoftware.com/.\(definition\)](http://www.mountaingoatsoftware.com/.(definition))
- Cohn, M.. (n.d., b). What is Scrum? In Mountain goat software. Retrieved June 3, 2012, from [http://www.mountaingoatsoftware.com/.\(definition\)](http://www.mountaingoatsoftware.com/.(definition))
- Cohn, M.. (n.d., c). A reusable Scrum presentation. In Mountain goat software. Retrieved June 3, 2012, from [http://www.mountaingoatsoftware.com/.\(figure\)](http://www.mountaingoatsoftware.com/.(figure))
- Creswell, J. W. (2007). *Qualitative inquiry & research design: Choosing among five approaches* (2nd ed.). Thousand Oaks, CA: Sage.

- Dubey, N. (2011). A paper presentation on software development automation by computer aided software engineering (CASE). *International Journal Of Computer Science Issues (IJCSI)*, 8(1), 182-184. Retrieved from www.ijcsi.org
- Earl, M. (2001). Knowledge management strategies: Toward a taxonomy. *Journal of Management Information Systems* 18(1), 215–233. Retrieved from <http://www.mesharpe.com/results1.asp?ACR=mis>
- Eccles, M., Smith, J., Tanner, M., Van Belle, J., & van der Watt, S. (2010). The impact of collocation on the effectiveness of agile is development teams. *Communications of The IBIMA*, 1-11. Retrieved from <http://www.ibimapublishing.com/journals/CIBIMA/cibima.html>
- Elminir, H. K., Khereba, E. A., Elsoud, M., & El-Hennawy, I. (2009). Application and evaluation of the personal software process. *International Journal of Basic & Applied Sciences*, 9(10), 55-78. Retrieved from <http://www.sciencepubco.com/index.php/ijbas>
- Emam, K., & Koru, A. (2008). A replicated survey of IT software project failures. *IEEE Software*, 25(5), 84-90. doi:1534417671
- Ferrand, D., Amyot, D., & Corrales, C. (2010). Towards a business intelligence framework for healthcare safety. *Journal of Internet Banking & Commerce*, 15(3), 1-9. Retrieved from <http://www.arraydev.com/commerce/jibc/>
- Fitzgerald, B. (2012). Software crisis 2.0. *IEEE Computer*, 45(4), 89-91. doi:10.1109/MC.2012.147

- Ganesh, N. N., & Thangasamy, S. S. (2012). Lessons learned in transforming from traditional to agile development. *Journal of Computer Science*, 8(3), 389-392. Retrieved from <http://thescipub.com/jcs.toc>
- Gassman, B., Salam, R. L., Bitterer, A., Hagerty, J., & Chandler, N. (2010). Predicts 2011: New relationships will change BI and analytics. Retrieved from <http://www.gartner.com/technology/core/home.jsp>
- Glazer, H., Dalton, J., Anderson, D., Konrad, M., & Shrum, S. (2008). CMMI or agile: Why not embrace both! Retrieved from <http://www.sei.cmu.edu/reports/08tn003.pdf>
- Gold, A. H., Malhotra, A., & Segars, A. H. (2001). Knowledge Management: An organizational capabilities perspective. *Journal Of Management Information Systems*, 18(1), 185-214. Retrieved from <http://www.mesharpe.com/>
- Hedgebeth, D. (2007). Data-driven decision making for the enterprise: an overview of business intelligence applications. *VINE*, 37(4), 414-420. doi:10.1108/03055720710838498
- Herschel, G. (2011). Hype cycle for analytic applications, 2011. Retrieved from <http://www.gartner.com/technology/home.jsp>
- Herschel, G., Hostmann, B., Rayner, N., & Bitterer, A. (2010). Clarifying the meanings of "analytics". 1-7. Retrieved from <http://www.gartner.com/technology/core/home.jsp>

- Hewagamage, C., & Hewagamage, K. P. (2011). Redesigned framework and approach for IT project management. *International Journal of Software Engineering & Its Applications*, 5(3), 89-106. Retrieved from <http://www.sersc.org/journals/IJSEIA/>
- Hullett, K., Nagappan, N., Schuh, E., & Hopson, J. (2011). Data analytics for game development. *ICSE: International Conference On Software Engineering*, 940-943. doi:10.1145/1985793.1985952
- IEEE Computer Society (2004). Guide to the software engineering body of knowledge (SWEBOK). Los Alamitos, CA: Angela Burgess.
- Ionel, N. (2009). Agile software development methodologies: an overview of the current state of research. *Annals of The University Of Oradea, Economic Science Series*, 18(4), 381-385. Retrieved from <http://www.doaj.org/doaj?func=openurl&issn=1222569X&genre=journal>
- Ivancenco, V., Boldeanu, D., & Mocanu, M. (2010). Efficient information management: Essential factor for high-performance management. *Metalurgia International*, 1551-1554. Retrieved from http://www.metalurgia.ro/metalurgia_int.html
- Jangping, W., Qingjing, L., Dejie, L., & Hongbo, X. (2010). Research on knowledge transfer influencing factors in software process improvement. *Journal of Software Engineering & Applications*, 3(2), 134-140. doi:10.4236/jsea.2010.32017
- Jiang, L., Eberlein, A., & Far, B. H. (2008). A case study validation of a knowledge-based approach for the selection of requirements engineering techniques. . *Requirements Engineering*, 13(2), 117-146. doi:10.1007/s00766-007-0060-2

- Jiangping, W., Hui, Z., Dan, W., & Deyi, H. (2010). Research on knowledge creation in software requirement development. *Journal of Software Engineering & Applications*, 3(5), 487-494. doi:10.4236/jsea.2010.35055
- Khan, U. A., Al-Bidewi, I. A., & Gupta, K. (2011). Object-oriented software methodologies: Roadmap to the future. *International Journal of Computer Science Issues (IJCSI)*, 8(5), 392-396. Retrieved from <http://www.ijcsi.org/>
- King, N., & Horrocks, C. (2010). *Interviews in qualitative research*. Thousand Oaks: CA: Sage.
- Lalsing, V., Kishnah, S., & Pudaruth, S. (2012). People factors in agile software development and project management. *International Journal of Software Engineering & Applications*, 3(1), 117-137. doi:10.5121/ijsea.2012.3109
- Laney, D. (2012). Ten reasons to reach beyond basic business intelligence. 7. Retrieved from <http://www.gartner.com/technology/core/home.jsp>
- Layman, L., Williams, L., & Cunningham, W. (2006). Motivations and measurements in an agile case study. *Journal of Systems Architecture: the EUROMICRO Journal - Special issue: AGILE methodologies for software production*, 52(11), 654 - 667. doi:10.1016/j.sysarc.2006.06.009
- Lee, G., & Xia, W. (2010). Toward agile: An integrated analysis of quantitative and qualitative field data on software development agility. *MIS Quarterly*, 34(1), 87-114. Retrieved from <http://www.misq.org/>
- Levy, M., & Hazzan, O. (2009). *Knowledge management in practice: The case of agile software development*. Paper presented at the Workshop on Cooperative and

Human Aspects on Software Engineering, 2009. CHASE '09. ICSE Vancouver, BC. <http://iee.org>

Linden, L. P. (2010). *A method for developing churchmanian knowledge management systems*. University of Central Florida). *ProQuest Dissertations and Theses*, 155. Retrieved from <http://search.proquest.com/docview/733931756?accountid=14872>. (733931756).

Linden, L., Kuhn Jr., J., Parrish Jr., J., Richardson, S., Adams, L., & Elgarah, W. (2007). Churchman's inquiring systems: Kernel theories for knowledge management. *Communications of AIS*, 2007(20), 836-871.

Lingling, Z., Jun, L., Yong, S., & Xiaohui, L. (2009). Foundations of intelligent knowledge management. . *Human Systems Management*, 28(4), 145-161. doi:10.3233/HSM-2009-0706

Machi, L. A., & McAvoy, J. (2009). *The literature review*. Thousand Oaks, CA: SAGE.

Mansour, E., Alhawari, S., Talet, A., & Al-Jarrah, M. (2011). Development of conceptual framework for knowledge management process. *Journal of Modern Accounting & Auditing*, 7(8), 864-877. Retrieved from <http://www.davidpublishing.com/davidpublishing/journals/J2/acc2011/accountant2011/414.html>

Maxwell, J. A. (2005). *Qualitative research design: An interactive approach* (2nd ed. Vol. 41). Thousand Oaks, CA: SAGE.

- McAvoy, J., & Butler, T. (2009). The role of project management in ineffective decision making within agile software development projects. *European Journal of Information Systems, 18*, 372-383. doi:10.1057/ejis.2009.22
- Mieritz, L. (2012). Gartner surveys show why projects fail. Retrieved from <http://thisiswhatgoodlookslike.com/2012/06/10/gartner-survey-shows-why-projects-fail/>
- Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook* (2nd ed.). Thousand Oaks, CA: Sage.
- Mishra, D., & Mishra, A. (2011). Research trends in management issues of global software development: Evaluating the past to envision the future. *Journal of Global Information Technology Management, 14*(4), 48-69. Retrieved from <http://www.uncg.edu/bae/jgitm/>
- Molaei, M. (2011). Knowledge management model for managing knowledge among related organizations. *World Academy Of Science, Engineering & Technology, 7*4426-429. Retrieved from <http://www.waset.org/>
- Moses, J. W., & Knutsen, T. L. (2007). *Ways of knowing: Competing methodologies in social and political research*. New York, NY: Palgrave/Macmillan.
- Moustakas, C. (1994). *Phenomenological research methods*. Thousand Oaks, CA: Sage.
- Nerur, S., & Balijepally, V. (2007). Theoretical reflections on agile development methodologies. *Communications of the ACM, 50*(3), 79–83. Retrieved from <http://cacm.acm.org/>

- Neves, F., Rosa, V., Correia, A., & Neto, M. (2011). *Knowledge creation and sharing in software development teams using agile methodologies: key insights affecting their adoption.* . Paper presented at the CISTI (Iberian Conference On Information Systems & Technologies / Conferência Ibérica De Sistemas E Tecnologias De Informação).
- Nonaka, I., & Takeuchi, H. (1995). *The knowledge-creating company.* Oxford, UK: Oxford University Press.
- Northern, C., Mayfield, K. M., Benito, R., & Casagni, M. (2010). Handbook for implementing agile in department of defense information technology acquisition. Retrieved from http://www.mitre.org/work/tech_papers/2011/11_0401/11_0401.pdf
- Omar, M., Syed-Abdullah, S., & Yasin, A. (2011). The impact of agile approach on software engineering teams. *American Journal of Economics & Business Administration*, 3(1), 12-17. Retrieved from <http://thescipub.com/ajeba.toc>
- Ow, T. T., & Morris, J. G. (2010). An experimental study of executive decision-making with implications for decision support. *Journal Of Organizational Computing & Electronic Commerce*, 20(4), 370-397. doi:10.1080/10919392.2010.516642
- Patil, M. V., & Nageswara Yogi, A. M. (2011). Importance of data collection and validation for systematic software development process. *International Journal of Computer Science & Information Technology*, 3(2), 260-278. doi:10.5121/ijcsit.2011.3220

- Patton, M. Q. (2002). *Qualitative research & evaluation methods* (3rd ed.). Thousand Oaks, CA: Sage.
- Pelrine, J. (2011). On understanding software agility: A social complexity point of view. *Emergence: Complexity & Organization*, 13(1/2), 26-37. Retrieved from http://emergentpublications.com/ECO/about_eco.aspx
- Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering*, 13(3), 303-337. doi:10.1007/s10664-008-9065-9
- Qumer, A., & Henderson-Sellers, B. (2008). A framework to support the evaluation, adoption and improvement of agile methods in practice. *The Journal of Systems and Software*, 81, 1899-1919. Retrieved from <http://www.journals.elsevier.com/journal-of-systems-and-software/>
- Rajteric, I. (2010). Overview of business intelligence maturity models. *Management: Journal of Contemporary Management Issues*, 15(1), 47-67. Retrieved from <http://www.efst.hr/management/>
- Ramesh, B., Lan, C., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: An empirical study. *Information Systems Journal*, 20(5), 449-480. doi:10.1111/j.1365-2575.2007.00259.x
- Rao, K. N., Naidu, G. K., & Chakka, P. (2011). A study of the agile software development methods, applicability and implications in industry. *International Journal of Software Engineering & Its Applications*, 5(2), 35-45. Retrieved from <http://www.sersc.org/journals/IJSEIA/>

- Rao, M. (2005). *Knowledge management tools and techniques*. Burlington, MA: Elsevier.
- Rayner, N. (2011). Maverick research: Judgement day, or why we should let machines automate decision making. 18. Retrieved from <http://www.gartner.com/technology/core/home.jsp>
- Rinko-Gay, B. (2009). Test reporting on an agile project. *Journal of the Quality Assurance Institute*, 23(1), 5. Retrieved from <http://www.qaiglobalinstitute.com/Innerpages/Default.asp>
- Rodger, J. A., Pankaj, P., & Nahouraii, A. (2011). Knowledge management of software productivity and development time. *Journal of Software Engineering & Applications*, 4(11), 609-618. doi:10.4236/jsea.2011.411072
- Rubin, E., & Rubin, H. (2011). Supporting agile software development through active documentation. *Requirements Engineering*, 16(2), 117-132. doi:10.1007/s00766-010-0113-9
- Ryan, S., & O'Connor, R. V. (2009). Development of a team measure for tacit knowledge in software development teams. *Journal of Systems and Software*, 82, 229-240. doi:10.1016/j.jss.2008.05.037
- Salam, R., & Clearley, D. (2012). Advanced analytics: predictive, collaborative, pervasive. 16. Retrieved from <http://www.gartner.com/technology/core/home.jsp>
- Salo, O. O., & Abrahamsson, P. P. (2008). Agile methods in European embedded software development organizations: a survey on the actual use and usefulness of Extreme Programming and Scrum. *IET Software*, 2(1), 58-64. doi:10.1049/iet-sen:20070038

- Schlegel, K., Salam, R., Austin, T., & Roswell, C. (2009). The rise of collaborative decision making. 7. Retrieved from <http://www.gartner.com/technology/core/home.jsp>
- Schwaber, K. (1995). *SCRUM development process*. Paper presented at the 10th Annual ACM Conference on Object Oriented Programming Systems, Languages and Applications (OOPLSA).
- Sharp, H., Robinson, H., & Petre, M. (2009). The role of physical artefacts in agile software development: Two complementary perspectives. *Interacting with Computers*, 21(1-2), 108-116. doi:10.1016/j.intcom.2008.10.006
- Sholla, A., & Nazari, E. (2011). Knowledge Management and factors that influence the success of codification strategies in medium-sized companies that develop software: The model, strategies and tools. *Journal of Information Technology & Economic Development*, 2(1), 54-63. Retrieved from <http://www.informingscience.us/icarus/journals/jiito/>
- Shull, F., Melnik, G., Turhan, B., Layman, L., Diep, M., & Erdogmus, H. (2010). What do we know about Test-Driven Development? *IEEE Software*, 27(6), 16-19. doi:10.1109/MS.2010.152
- Siwen, Y., & Jun, A. (2010). Software test data generation based on multi-agent. *International Journal of Software Engineering & Its Applications*, 4(3), 69-76. Retrieved from <http://www.sersc.org/journals/IJSEIA/>

- Slaughter, S., & Kirsch, L. (2006). The effectiveness of knowledge transfer portfolios in software process improvement: A field study. *Information Systems Research*, 17(3), 301-320. doi:10.1287/isre.1060.0098
- Smith, D. (2011). Hype cycle for cloud computing. 82. Retrieved from <http://www.gartner.com/technology/core/home.jsp>
- Smith, J. A., Flowers, P., & Larkin, M. (2009). *Interpretive phenomenological analysis: Theory, method and research*. Thousand Oaks, CA: Sage.
- Smith, R. G., & Farquahar, A. (2000). The year ahead for knowledge management: An AI perspective. *AI Magazine*, 21(4), 17-44. Retrieved from <http://www.aaai.org/Magazine/magazine.php>
- Sudhakar, G., Farooqb, A., & Patnaikc, S. (2012). Measuring productivity of software development teams. *Serbian Journal Of Management*, 7(1), 65-75. doi: 10.5937/sjm1201065S
- Sullivan, T. J. (2001). *Methods of social research*. Orlando, FL: Harcourt Inc.
- Sutherland, J., Jakobsen, C. R., & Johnson, K. (2007). Scrum and CMMI level 5: The magic potion for code warriors. doi:10.1109/AGILE.2007.52
- Tessem, J., & Maurer, F. (2007). *Job satisfaction and motivation in a large agile team*. Paper presented at the Agile Processes in Software Engineering and Extreme Programming, Berlin.
- The Standish Group. (n.d.). *Chaos summary for 2010*. Retrieved from <http://insyght.com.au/special/2010CHAOSSummary.pdf>

- Trochim, W., & Donnelley, J. P. (2006). *The research methods knowledge base* (3rd ed.). Cincinnati, OH: Atomic Dog.
- Turban, E., Sharda, R., & Delen, D. (2005). *Decision support systems and intelligent systems* (9th ed.). Upper Saddle River, NJ: Pearson/Prentice Hall.
- Wadhwa, A., & Mitra, N. (n.d.). People, process and tools: What is more important – the order or the mechanics? Retrieved from <http://jindal.utdallas.edu/files/11WadhwaMitra-Paper.pdf>
- Yeoh, W., & Koronios, A. (2010). Critical success factors for business intelligence systems. *Journal of Computer Information Systems*, 50(3), 23-32. Retrieved from <http://www.iacis.org/jcis/jcis.php>
- Zare, H., & Akhavan, A. A. (2009). Developing one heuristic algorithm for software production schedule in fuzzy condition. *Australian Journal Of Basic & Applied Sciences*, 3(3), 1685-1695. Retrieved from <http://www.ajbasweb.com/>
- Zhang, Y., & Patel, S. (n.d.). Agile Model-Driven Development in Practice. IEEE Software. *IEEE Software*, 28(2), 84-91. doi:10.1109/MS.2010.85

Appendix A: Qualitative Research Schedule

Research Interview Schedule

1. Please review the materials provided prior to the scheduled interview. (Research participants will be provided with the following data one week prior to the scheduled interview:

- a. Table 2 which shows the software failure rate from the 1994 – 2009 Standish Group reports as summarized by Hewagamage and Hewagamage (2011)

Table 2. Summary of findings from the 1994 – 2009 Standish Group reports on software failure.

	1994	1996	1998	2000	2002	2004	2006	2009
Successful	16%	27%	26%	28%	34%	29%	35%	32%
Challenged	53%	33%	46%	49%	51%	53%	46%	44%
Failed	31%	40%	28%	23%	15%	18%	19%	24%

(Hewagamage and Hewagamage, 2011, p. 90)

- b. Although the software failure rate has decreased since 2009, software development productivity has not kept pace with advancements in hardware; consequently, there is a new crisis in software development called the software crisis 2.0 (Fitzgerald, 2012). For example, there were

35 billion devices tied to the Internet in 2010 and the number of devices tied to the Internet is expected to increase to 100 billion by 2020. Efforts have been made to resolve the crisis; however, the efforts have been disjointed and are not likely to enable software organizations to take advantages of the available data.

- c. A list of definitions for data, analytics and KM from the literature.
 - i. Data and analytics

Descriptive analytics: Answer the questions what happened and what is happening and are used to measure and manage performance. Examples include reports, dashboards, and scorecards (Salam & Cearley, 2012). Descriptive analytics may be used to identify alternative solutions but may not provide an optimal solution (Turban et al., 2005).

Diagnostic analytics: Answers the questions why did it happen and what are the key relationships. Diagnostic analytics are used to understand outliers and variance, to create profiles, and to classify data. Examples include machine learning, interactive visualization, data mining and modeling, and content analytics (Salam & Cearley, 2012). Diagnostic analytics may be used to identify the underlying causes for irregularities (Turban et al., 2005).

Knowledge accumulation: the process of acquiring, capturing or obtaining knowledge (Gold, Malhotra, & Segars, 2001).

Knowledge creation: the process in which explicit and tacit knowledge is shared between individuals and groups within an organization through socialization, externalization, combination, and internalization (Nonaka & Takeuchi, 1995).

Knowledge retention: the process of organizing and preserving or storing knowledge (Mansour, Alhawari, Talet & Al-Jarrah (2011).

Knowledge transfer: the process of distributing knowledge to people other than those who generated, produced, or created the knowledge (Mansour et al., 2011).

Predictive analytics: Answers the questions what will happen, how risky is it, and what if it happened. Predictive analytics are used to forecast and test hypothesis and to model risk. Examples include forecasting applications, predictive models, and content analytics (Salam & Cearley, 2012).

Prescriptive analytics: Answers the questions what is the best option, how can an optimal solution be reached, and what should happen. Prescriptive analytics are used for risk management, business optimization, and recommending the best action. Examples include modeling, simulation, optimization, and visualization (Salam & Cearley, 2012).

Interview Questions:

1. What is your current role?

Current Role:	Software Manager
	Project Manager
	Agile Coach

2. How long have you used agile software development methods?

Experience	Mark one answer.
<1 year	
>1 year and <=3years	
>3 years and <=5 years	
>5 years	

3. How do agile software managers, project managers, and agile coaches describe their projects?
 - What is the project size?
 - What is the project duration?
 - What agile methodologies are used?
4. What do software managers, project managers, and agile coaches in agile software environments think about the need to improve software development productivity?
5. What do software managers, project managers, and agile coaches in agile software environments think about the use of analytics and KM to improve software development productivity?
6. How do software managers, project managers, and agile coaches in agile software environments currently use descriptive, diagnostic, prescriptive or predictive analytics to improve software development productivity in each of the following software activities?

- Requirements
 - Design
 - Construction
 - Testing
 - Maintenance
 - Configuration Management
 - Engineering Management
 - Process
 - Tools and Methods
 - Quality
7. How do software managers, project managers, and agile coaches in agile software environments currently use KM (knowledge creation, knowledge accumulation, knowledge retention, and knowledge transfer) to improve software development productivity in each of the following software activities?
- Requirements
 - Design
 - Construction
 - Testing
 - Maintenance
 - Configuration Management
 - Engineering Management

- Process
 - Tools and Methods
 - Quality
8. How do software managers, project managers, and agile coaches in agile software environments think descriptive, diagnostic, prescriptive, or predictive analytics could be used in the future to improve software development productivity in each of the following software activities?
- Requirements
 - Design
 - Construction
 - Testing
 - Maintenance
 - Configuration Management
 - Engineering Management
 - Process
 - Tools and Methods
 - Quality
9. How do software managers, project managers, and agile coaches in agile software environments think KM (knowledge creation, knowledge accumulation, knowledge retention, and knowledge transfer) could be used in the future to

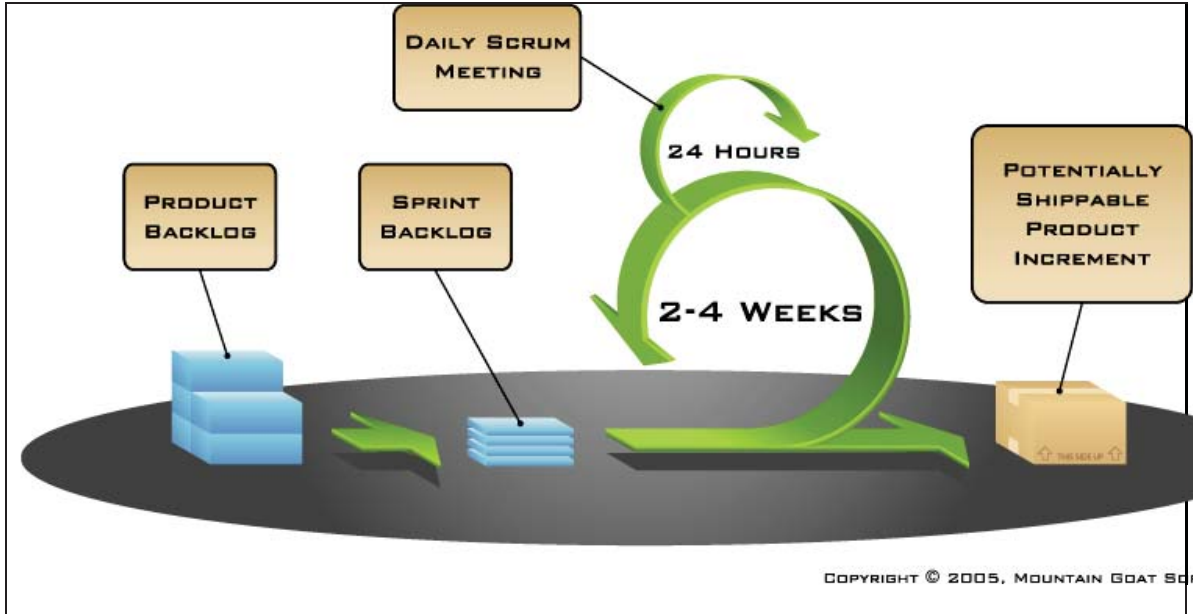
improve software development productivity in each of the following software activities?

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

10. What obstacles do software managers, project managers, and agile coaches in agile software environments think their organization should overcome to improve software development productivity? Consider obstacles that involve people, process and tools.

|

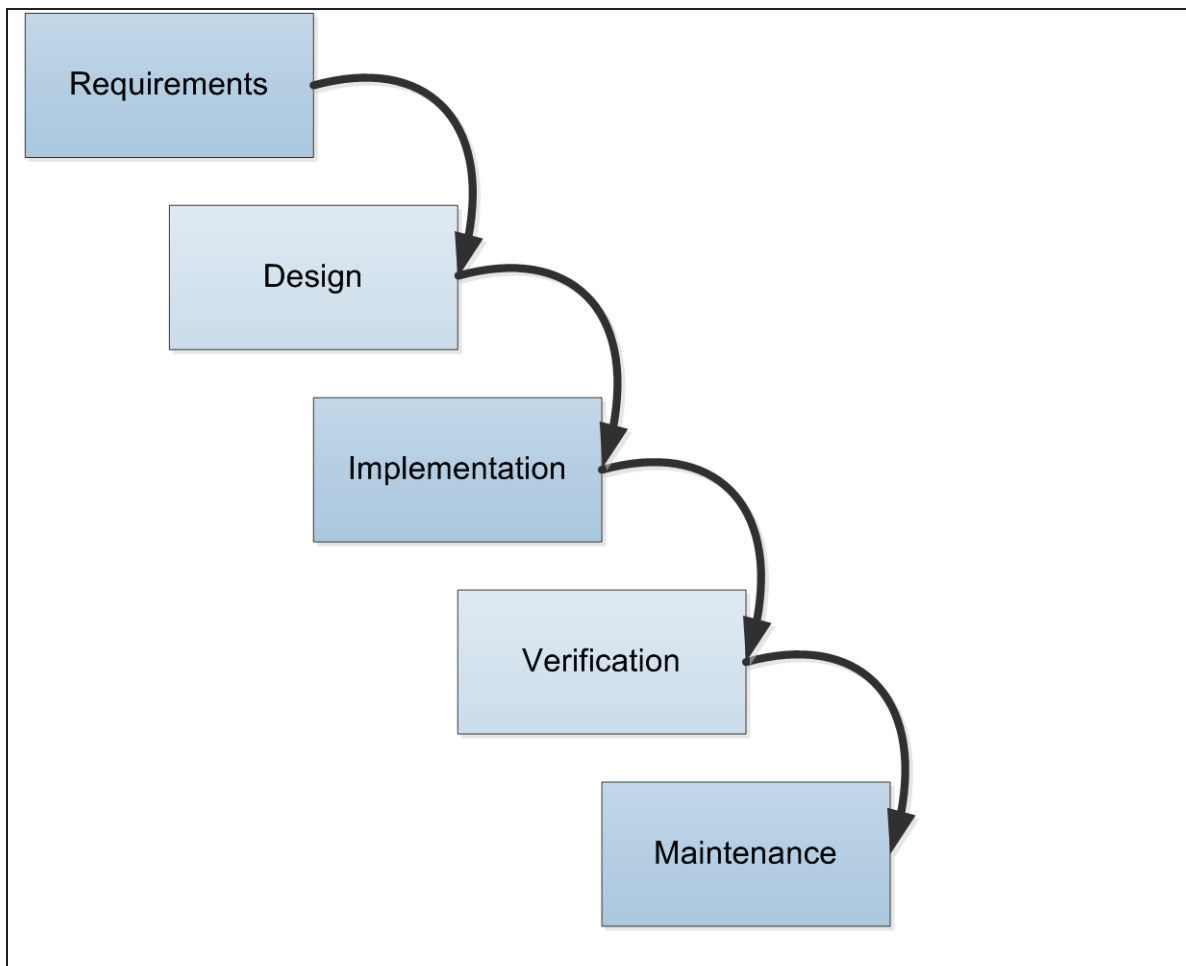
Appendix B: Agile Scrum Process Versus the Traditional Waterfall Process



(Cohn, n.d. c, "A reusable Scrum presentation")

"Traditional	Agile
Sequential	Iterative
Defined	Empirical
Plan-driven	Result-driven
Big-bang	Incremental
Specialized teams	Cross-functional teams
Test at the end	Test-first"

(Peltine, 2011, p. 28)



("waterfall model", n.d.)

Appendix C: Informed Consent Form

You are invited to take part in a research study of management's understanding of data driven decision making as a tool to improve productivity in an agile software development environment. You were chosen for the study because of your level expertise and your familiarity with agile software development methods. The purpose for this informed consent form is to allow you to understand this study before deciding whether to take part.

Organizations face global competition and increasing volumes of data that must be managed. If organizations can find ways to improve software development productivity, they may increase their opportunities and their ability to thrive and survive in a competitive world. This research study is being conducted by a researcher named Molly Brown, who is a doctoral student at Walden University. Research gathered in this study will be used to explore the lived experiences of software managers, project managers, and agile coaches who have managed agile software development projects.

Research Study Purpose Statement:

The purpose for this research study is to explore how agile software managers view data driven decision making, which includes data, analytics, and knowledge management, as a tool to improve software development productivity, to understand how agile software development organizations currently use data driven decision making to improve software development productivity, and to understand how agile software organizations may use data driven decision making in the future to improve software development productivity.

Procedures:

Participate in a 1-2 hour individual interview regarding the use of data driven decision-making in an agile software development environment. Provide documentation that supports your experiences with data driven decision making in an agile software development environment. The interview will be audio taped for analysis by the researcher. You will be provided with a copy of your transcribed interview for your review. At the conclusion of the research study, your interview transcript along with any documentation provided will be transferred to DVD, and both the DVD and audio tape will be stored in a secure location for the required 5 years. Both the audio tape and DVD will be destroyed at the end of the 5 years.

Voluntary Nature of the Study:

Your participation in this study is voluntary. This means that everyone will respect your decision of whether or not you want to be in the study. No one will treat you differently if you decide not to be in the study. If you decide to join the study now, you can still change your mind during the study. If you feel stressed during the study you may stop at any time. You may skip any questions that you feel are too personal.

Risks and Benefits of Being in the Study:

The interview will take approximately 1-2 hours to complete and will involve a detailed discussion of your lived experiences regarding the use of data driven decision making as a tool to improve software development productivity in an agile software development environment. This study could potentially benefit the agile community by providing a description of how software development productivity is currently improved and how

software development productivity may be improved in the future. The risks of participation in this research study are minimal as participants will not be subject to any stress or risk greater than would normally be encountered in everyday life.

Compensation:

Although participation is voluntary, you will receive a \$10 Amazon.com gift certificate as a thank you for your participation and you will be given a summary of the research findings.

Confidentiality:

Any information provided will be entirely confidential. The researcher will not use your information for any purposes outside of this research project. Also, the researcher will not include your name or anything else that could identify you in any reports of the study.

Contacts and Questions:

You may ask any questions you have now. Or if you have questions later, you may contact the researcher via telephone (602-721-4568) or email (mary.brown2@waldenu.edu). If you want to talk privately about your rights as a participant, you can call Dr. Leilani Endicott. She is the Walden University representative who can discuss this with you. Her phone number is 1-800-925-3368, extension 1210. Walden University's approval number for this study is [nnnn] and it expires on [M/D/Y]. The researcher will give you a copy of this form to keep.

Statement of Consent:

I have read the above information and I feel I understand the study well enough to make a decision about my involvement. Please reply to this e-mail with the words "I consent" if

you agree to participate in this research study. Your reply to this e-mail with the words “I consent” serves as your consent to participate in this research study.

Appendix D: Research E-mail Invitation

From: Molly Brown

Email address: azmollybrown@cox.net

Date: [Date]

Dear [Research Participant Name]:

You are invited to take part in a research study of data driven decision making (DDD), which includes the use of analytics and knowledge management, as a tool to improve software development productivity in an agile software development environment.

You were selected based on your experience with agile software development methods including Scrum methods. The research will be conducted by Molly Brown, Certified Scrum Master (CSM) and a doctoral candidate at Walden University. The purpose for this research is to understand how software development productivity may be improved through the use of analytics and knowledge management. The results of this research study may benefit the software development community by improving the understanding of the tools and techniques that can be used to improve software development productivity.

Please read the attached Letter of Informed Consent and reply to this e-mail with the words “I consent” if you agree to participate in this research study. Your reply to this e-mail with the words “I consent” serves as your acknowledgement that you are eighteen years of age or older and that you consent to participate in this research study.

Sincerely,

Molly Brown

Attachment

Appendix E: Curriculum Vitae

Mary Erin Brown, MA, MS, PhD
mollybrownaz@gmail.com

EDUCATION:

PhD Management – Management.....	2013
Walden University, Minneapolis, MN	
Dissertation Topic: Data driven decision making as a tool to improve software development productivity	
Dissertation Advisor: Dr. David Gould	
Master of Science Information Management.....	1998
Arizona State University	
Master of Arts – Educational Technology	1976
Western Michigan University	
Bachelor of Arts – Education.....	1970
Western Michigan University	

TEACHING EXPERIENCE:

Instructor of Information Technology	1999-2009
University of Phoenix, Phoenix, AZ	
<ul style="list-style-type: none"> • Taught face-to-face courses to undergraduate students in Bachelor Degree program. Course topics include project management, critical thinking, and business systems development. Received average course evaluation of 3.9/4.0. 	
Instructor of Technical Project Management.....	2003-2004
ITT Technical Institute	
<ul style="list-style-type: none"> • Taught face-to-face courses to undergraduate students in Associates Degree program. Course topics include project management and E-commerce. 	

COURSES TAUGHT:

CMGT 410	Project Planning and Implementation
CMGT 575	Project Management
WEB 350	The Internet Concepts and Applications
BSA 375	Fundamentals of Business Systems Development

CIS 319	Computers and Information Processing
CSS 330	Critical Thinking and Computer Logic
CMGT 325	Organizational Communications
EC 321	Introduction to E-Commerce
EC 312	Project Management Techniques

OTHER EXPERIENCE:

Embedded Software Training Lead	2004-2011
Boeing, Mesa AZ	
<ul style="list-style-type: none"> • Led the Warfighter Machine Interface (WMI) effort to coordinate with the Future Combat Systems (FCS) / Brigade Combat Modernization (BCTM) program training management, their suppliers and their subcontractors to design and develop embedded training and to embed training systems into the operational software. • Advised BCTM Training suppliers on cost, schedule, and technical issues impacting the embedded training development • Managed separate Future Combat Systems, FCS Contract Line Item (CLIN) that included managing WMI team and suppliers to develop embedded training User Interfaces, and WMI Supplier to develop WMI Soldier training • Developed embedded training standards and guidelines in coordination with FCS Training IPT. 	
Instructional Design Lead	2003-2004
ComForce, Mesa, AZ contractor to The Boeing Company, Boeing, Mesa AZ	
<ul style="list-style-type: none"> • Member of the Boeing Defense Systems, Mission System Armament IPT process team implementing CMMI processes for the MSA IPT. • Led the effort to implement process improvements within MSA Common Test Environment (CTE) • Led Apache Longbow engineers and a team of training developers to design, develop, and deliver over 75 days of technical training to Fuji Heavy Industries covering Apache Longbow hardware and software. 	
Project Manager	1999-2001
MarchFIRST, Phoenix, AZ	
<ul style="list-style-type: none"> • Managed airline industry Internet and Intranet projects including development of jetBlue infrastructure and websites for National Airlines and TWA 	
IT Project Manager	1998-1999
Apollo Group, Phoenix, AZ	

- Implemented infrastructure, curriculum, marketing, and accounting system at 12 UOP campuses to support delivery of technology based certification programs including MCSE and A+.

Program Manager..... 1993-1996
Intel, Chandler, AZ

- Managed cross-site project to assess the network, hardware, and software infrastructure improvements needed to enable delivery of Intel computer-based-training needed for Intranet delivery of computer-based-training.
- Managed implementation of Components Training Department's Alternative Training Partnered with IT and Intel University, and other customers to build and improve corporate systems for alternative training design & delivery, increasing Component's Training Alternative Training Delivery from 400 seats over 4,000 seats.
- Awarded Division Recognition for multisite database Implementation
- Managed instructor base and materials for corporate wide Gas Systems Training Curriculum
- Led project to develop a Virtual Reality simulation of the Anelva 1052

Manager, Pilot Training Technology 1991-1993
United Airlines Flight Training Center, Denver, CO

- Managed United Airlines supplier to develop database to support UAL Advanced Qualification Program
- Developed United Airlines pilot training task analysis for the Advanced Qualification Program in coordination with Boeing for the initial Boeing 777 pilot training program.

Instructional Design Lead 1989-1991
McDonnell Douglas Aircraft, Aurora, CO

- Led instructional design team for the Tanker Transport Training System, TTTS program
- Led team of 60 instructional designers for the USAF Special Operations Forces ATS program

- Instructional Design Lead..... 1987-1989
Infotec Development, Inc., Colorado Springs, CO
- Managed team to develop computer-based-training for the USAF Consolidated Space Operations Center

CERTIFICATIONS:

SCRUM Master certified (SCM)
Project Management Professional, PMP certified,

PROFESSIONAL AFFILIATIONS:

Project Management Institute member
American Society for Training and Development member
Knowledge Management Association member

REFERENCES:

John Rosenberger
Email: dospalomas06@yahoo.com
Phone: 915-261-8026

Dr. David Gould
Email: david.gould@waldenu.edu
Phone: 206-409-4021

Mark Busby
Email: marksbusby1@gmail.com
Phone: 602-920-3648