CONSTRUCTING REQUIREMENTS: A QUALITATIVE STUDY OF CHALLENGES ENCOUNTERED DURING REQUIREMENTS ELICITATION FOR INFORMATION SYSTEMS

by

David L. Gibbs, B.S., M.S.

A dissertation submitted to the Graduate Council of
Texas State University in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
with a Major in Adult, Professional, and Community Education
May 2015

DEDICATION

For K, always.

ACKNOWLEDGEMENTS

I would like to thank the individuals who participated in this study by generously sharing their professional experiences so that others may benefit. I have been blessed throughout my career with the privilege of working with the most encouraging and supportive colleagues.

I also recognize my dissertation committee Chair, Dr. Ann Brooks, for her unwavering support and encouragement throughout the duration of this research undertaking. Committee members Dr. Steven Furney and Dr. Joellen Coryell provided essential guidance and encouragement at critical points. Dr. Tiankai Wang deserves special thanks for joining my committee and providing a much needed Health Information Management perspective. My friends from APCE Cohort 2006 also provided ongoing support and motivation throughout the learning adventure we shared.

Another supporter of this effort was Dr. Gene Bailey who has been an advisor, mentor, and friend since my earliest undergraduate classes in the Department of Computer Science at East Tennessee State University. Gene's enthusiasm for teaching has been a constant source of inspiration. I would also like to acknowledge Dr. Gordon Bailes and the entire ETSU Gang for contributing to those earliest, most cherished, exposures to academia.

Appreciation also goes to Dr. Doug Smith of the University of the Pacific for providing an exceptional opportunity to participate as a faculty member in the

Department of Computer Science. That short but meaningful experience energized my passion to continue a lifelong learning adventure in higher education.

And finally, this endeavor would not have been successful without inspiration and encouragement from my beautiful bride, Dr. Karen Gibbs. In so many ways, she is a precious reminder that it is good to have dreams.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ABBREVIATIONS

ACM	Association of Computing Machinery
AHIMA	American Health Information Management Association
EHR	Electronic Health Record
IEEE	Institute of Electrical and Electronics Engineers
HIMSS	Health Information and Management Systems Society
IT	Information Technology
IS	Information System
ISO	International Organization of Standards (abbreviation of French)
KAOS	Knowledge Acquisition in Automated Specification
LTC	Long-term Care
SDLC	Software (or System) Development Lifecycle
SRS	Software (or System) Requirements Specification
UML	Unified Modeling Language

ABSTRACT

This study identifies common challenges faced by practitioners while eliciting requirements for information systems. Participants representing both providers and consumers of information systems were interviewed using critical incident technique to collect rich qualitative data for grounded theory analysis. Vignettes from participants describe the challenges they faced as stakeholders on projects in the fields of health information management, insurance, and federal government systems. Grounded theory analysis revealed common challenges in three primary categories: change, communications, and knowledge. The emerging theory proposes elicitation of information systems requirements would benefit from an approach combining constructivism and social constructionism rather than traditional positivist approaches. Constructing requirements, rather than gathering requirements, reduces the risks associated with the common challenges and increases the likelihood of perceived success.

I. INTRODUCTION

Background of the Study

One of the most challenging activities of building an information system is determining what to build (Brooks, 1987; Gottesdiener, 2002). In some fields, the process of determining need is called *needs assessment* while the information systems community favors the terms *requirements analysis, requirements engineering,* or other similar phrases. Regardless of the terminology, the goal is to completely and precisely understand needs and manage expectations of the people involved in a project, collectively known as the *stakeholders*. My own experience of 30 years working with information systems is that requirements processes are full of challenges. Current literature confirms that requirements-related issues continue to pose significant challenges which inspire me to explore why this critical, fundamental activity remains so problematic after many generations of progressively advanced services. Despite decades of evolving processes to elicit and specify requirements, the information systems industry continues to suffer from extremely high levels of project failures, the majority of which can be traced to faulty requirements (Davis, Fuller, Tremblay, & Berndt, 2006; Gale, 2011; Leffingwell, 2011). While statistics vary, Sajid, Nayyar, and Moshin found "requirements elicitation counts more than 71% in a project's success or failure" (2010, p. 11). Berg summarizes the disappointing status for healthcare information systems with "It has become evident that there are many more failure stories to tell than there are success stories" (2001, p. 143). One recent high profile example was the 2013 launch of the U.S. Department of Health and Human Services' website healthcare.gov which did not initially meet expectations due to a variety of contributing factors including "evolving requirements" and "multiple definitions of success" (McKinsey, 2013, slide 4). This study explores underlying challenges that lead to these factors and others.

There is abundant research and tool development regarding the processes of specifying, managing, engineering, optimizing, and tracking requirements once they are identified. While these are important areas of the overall requirements process, my focus for this study is achieving understanding and consensus among all stakeholders regarding the appropriate set of requirements to be managed. A rough analogy would be climbing a ladder. Rather than studying the process of moving safely and effectively from one rung to the next, my research interest is to make sure the ladder is placed on the appropriate wall. Getting that earliest orientation made correctly is critical for the remaining steps to achieve their desired objectives.

Authors have established multiple levels of abstraction for requirements. For example, Gottesdiener differentiates business requirements, user requirements, and software requirements (2002). Laplante uses a different classification which distinguishes user requirements, system requirements, and design specifications (2014). For the purposes of this study, I do not distinguish among different classifications of requirements. Instead I am generalizing requirements in the spirit of Cardinal's statement that "*Perceptions* define what are required" (2014, p. 32). I am using the simple definition that a requirement is an expectation from the stakeholders.

Although simple, my definition is similar to the one used by the International Organization of Standards which defines a requirement as a "statement which translates or expresses a need and its associated constraints and conditions" (2011, p. 5).

My perspective is that the process of defining requirements is fundamentally an exercise in establishing and managing expectations of the group of people involved with the project, the stakeholders. For this study, stakeholder is defined to include people who both need information systems to be provided and who have the responsibility of providing the systems. Some authors distinguish between the team responsible for building the system and the stakeholders who are involved in all other ways (Cardinal, 2014). Others separate *system* stakeholders who drive the "primary" requirements from *project* stakeholders whose requirements are somehow secondary (Leffingwell, 2011, p. 120). The International Organization of Standards defines a stakeholder rather formally as an "individual or organization having a right, share, claim, or interest in a system or in its possession of characteristics that meet their needs and expectations" and goes on to note that "stakeholders include, but are not limited to, end users, end user organizations, supporters, developers, producers, trainers, maintainers, disposers, acquirers, customers, operators, supplier organizations, accreditors, and regulatory bodies" (2011, p. 6).

The purpose of this study does not require distinguishing among different classes of stakeholders, so I will use Laplante's definition of stakeholders as the "broad class of individuals who have some interest (a stake) in the success (or failure) of the system in question" (2014, p. 32).

I subscribe to the view that systems don't have requirements – *people* have requirements. Requirements are expectations and are therefore prone to be highly subjective. To highlight this subjectivity, some authors use the term *desirement* to blend the terms *desire* and *requirement* (Cardinal, 2014). While the concept of a *desirement* is consistent with my perspective, I will continue to use the traditional term *requirement* to avoid confusion.

Just as requirements are highly subjective, so are perceptions of a project's success or failure in meeting those requirements. In this study, I am adopting Berg's point of view that

> In the end, this final decision is about the attachment of the label 'success' or 'failure' (or anything in between) to a particular situation… The question whether an implementation has been successful or not is *socially negotiated*" among the various stakeholders. (2001, p. 144)

The perspectives and priorities of individual stakeholders are central to the effectiveness of the requirements process. Similarly, I am using a simple definition for success of an information system project to be that stakeholders' expectations are met. While these definitions are extremely simplified when compared to the detailed definitions available in literature, they are sufficient to explain the findings of this study.

This research is the result of my desire to understand, and raise awareness about, some of the common challenges encountered when eliciting requirements for information systems, with an ultimate goal of improving success rates of future projects. Reflecting on my own temperament, I feel empowered by knowledge and facts. Having solid knowledge about a topic provides a level of confidence required for me to speak up and take action with authority. When faced with uncertainty, I tend to gather more information rather than taking a stand on what may be shaky ground. Such delays in action can be costly in the forms of missed opportunity or being perceived as noncommittal. It is my intent that this research will begin to establish facts that can be used by others to accelerate their own confidence in order to initiate timely action and to inspire further related research.

During this qualitative grounded theory study, I interviewed a variety of fellow practitioners to collect rich data that might yield a theory about the challenges faced during requirements analysis and perhaps uncover ways to address those challenges. I expected this study to reveal a wide assortment of challenges, many of which I could not anticipate. The data collected from interviews were complemented with data from relevant literature as well as my own observations as a practitioner/researcher involved with pertinent information systems projects.

There were two sensitizing concepts that provided inspiration for this study, but my interest was not constrained to only these two areas. Charmaz explains that sensitizing concepts "give you initial ideas to pursue and sensitize you to ask particular kinds of questions about your topic. Grounded theorists often begin their studies with certain guiding empirical interests to study" (2006, p. 16). The first sensitizing concept for my study was the use of positivist approaches to address an activity that I believe is inherently constructivist. Given the highly subjective nature of personal priorities and expectations, I was curious to see what challenges emerged from the collected data related to conflicts between positivism and constructivism. Despite the subjective nature of information systems requirements, the terms commonly used in industry and literature to describe the process include *gathering, discovering,* and *capturing* requirements. These expressions reflect the traditional, positivist view that a set of finite, objective requirements for the desired system exists and simply needs to be collected and documented. Thomas and Hunt state clearly "The fundamental problem here is that folks believe that underlying every project there’s some absolute, discoverable set of requirements” (2004, p. 13). This view appears again a decade later as “It is not realistic to think that requirements are just out there somewhere, and the only thing you need to do is get explanations from stakeholders” (Cardinal, 2014, p. 13). The traditional positivist approach may hint at the root cause for some of the challenges faced by stakeholders during requirements process. During data analysis, I watched for statements that may indicate positivist and constructivist influences.

The second sensitizing concept was the impact of stakeholder turnover. Since requirements are expectations held by stakeholders, when a stakeholder joins or leaves the group there is likely change to the set of requirements. These changes would be revealed only through collaboration among the stakeholders. I analyzed the collected data for real world challenges related to stakeholder turnover during projects.

Much of the effort to date directed at improving information systems requirements has been focused on symbolic notations and structured text methods to record and concisely communicate the details gathered (Heaven & Finkelstein, 2004; Maiden, 2005). While these valuable efforts address the very real challenges of effectively communicating precise details among people with different perspectives and levels of understanding, these efforts do not address the need to ensure the details accurately represent the expectations of the stakeholders. There remains a gap in understanding between those who have needs and those who are charged with addressing those needs. I believe the gap would be at least partially closed by a more constructivist approach to requirements. I looked for statements related to this influence as I analyzed the collected data.

My viewpoint is that requirements are *constructed* by the collaboration that occurs among stakeholders during successful requirements workshops. Effective discussions enable each stakeholder to better understand the perspectives of the other stakeholders which lead to compromise and negotiated consensus. I expected this study to reveal specific challenges, some of which would be related to positivist vs. constructivist approaches to requirements. Once this study identifies the challenges faced during requirements analysis, subsequent research and practices may address those challenges to improve the rate of success for future information systems projects.

Research Question and Significance

The research question addressed by this study is: What challenges are encountered by stakeholders when identifying requirements for information systems? The scope of my interest includes all types of challenges experienced by individuals as they participate in collaborative workshops that may restrict how their expectations are respected by the group. I anticipated challenges related to priority conflicts, personality conflicts, responsibility conflicts, lack of understanding, lack of time, and lack of interest as well as my sensitizing concepts of constructivism vs. positivism and stakeholder turnover. I was especially interested in whatever unanticipated challenges surfaced. By identifying the challenges and analyzing any patterns that emerge across participants, this research is intended to increase awareness of those challenges so that they may be addressed and mitigated. The long-term significance of this study is to improve the success rate of information systems projects.

The success rate of information systems projects is surprisingly and disappointingly low. A 2010 survey of 10,000 information systems projects from around the globe revealed that only 37% of the projects were considered successful. The survey was mentioned in an article highlighting the improvement in success rates over similar surveys from earlier years (Gale, 2011). In 2008 and 2004 the success rates were 32% and 28%, respectively. Even with the improvement, these success rates hardly seem worth celebrating. Since a large percentage of information system project failures are attributed to faulty requirements (Davis et al., 2006; Leffingwell, 2011), this is a problem worthy of further investigation with a fresh perspective.

Regardless of what specific challenges are identified by this qualitative study, the results may provide valuable insight to practitioners, educators, and researchers regarding issues to be addressed during future requirements workshops to maximize effectiveness, improve requirements, and ultimately improve the overall success rate of information systems projects. A secondary benefit may exist for at least some participants of the study if their participation inspires them to have a fresh perspective during their next requirements workshop.

Researcher Perspectives

As both researcher and practitioner, I have an inside perspective on requirements analysis and see great need and opportunity for improvement of current processes. The literature review for this study aligns with my personal experience that significant challenges exist to hinder elicitation of information system requirements, despite decades of evolution to improve requirements processes. My academic exposure to constructivism and social construction of knowledge caused me to ponder whether at least some of the challenges with current processes are related to a positivist approach being applied to an inherently constructive activity. I believe that a constructivist approach to requirements would improve perceived success rates of information systems projects by managing expectations of stakeholders. This research study is intended as a first step to understand the challenges that stakeholders perceive and to begin to determine whether the positivist versus constructivist dilemma deserves further study. While this study is not restricted to challenges related to constructivism versus positivism, it is important that I acknowledge this influence from my preconceptions (Mason, 2002).

Unlike projects to build physical systems such automobiles or buildings, information systems are abstract and inherently socially constructed. While physical systems are governed by laws of nature such as physics and chemistry, information systems are governed largely by the imagination, capabilities, and expectations of the people involved. The requirements process is “all about perceptions” (Davis, 2005, p. 44) which means it’s all about the people, not the technology. Although the common expression is that *systems* have requirements, I believe it is more appropriate to say the *people* have requirements in the form of expectations. A project perceived as successful is one that meets the expectations of the people who matter, the *stakeholders*. A project may be perceived as unsuccessful if even one vocal stakeholder is sufficiently disappointed by the results. It is not uncommon for stakeholders to have conflicting expectations which, if left unresolved, may doom a project to a perception of failure from the start. Conflicts must be identified and reconciled among the stakeholders early through interactive collaboration, resulting in socially constructed requirements.

Since information system requirements are indeed stakeholder expectations, I consider requirements to be a form of socially constructed knowledge and therefore prime candidates for application of constructivist principles. Requirements cannot be simply gathered, discovered, or collected. Requirements must be constructed through interactive collaboration among the stakeholders. I am conducting this qualitative study of real-world stakeholders in order to gather data that may reveal interesting information about challenges faced by stakeholders, including the effects of positivist or constructivist approaches to requirements along with whatever other challenges emerge.

This qualitative research study is based on a few premises that are rather elementary but have not yet been widely combined and applied in the context of requirements elicitation for information systems projects. These concepts are: 1) any group of people will have diverse perspectives and priorities (Keirsey, 1998; Knowles, 1980; Lewin, 1946), 2) People, not technology, are the sources and judges of knowledge about requirements (Davis, 1993; Gottesdiener, 2002; Orr, 2004; Thomas & Hunt, 2004;), and 3) knowledge is constructed individually and socially (Berger & Luckman, 1967; Piaget, 1970; von Glaserfeld, 1989). Figure 1 graphically depicts the relationships among the concepts of the study.

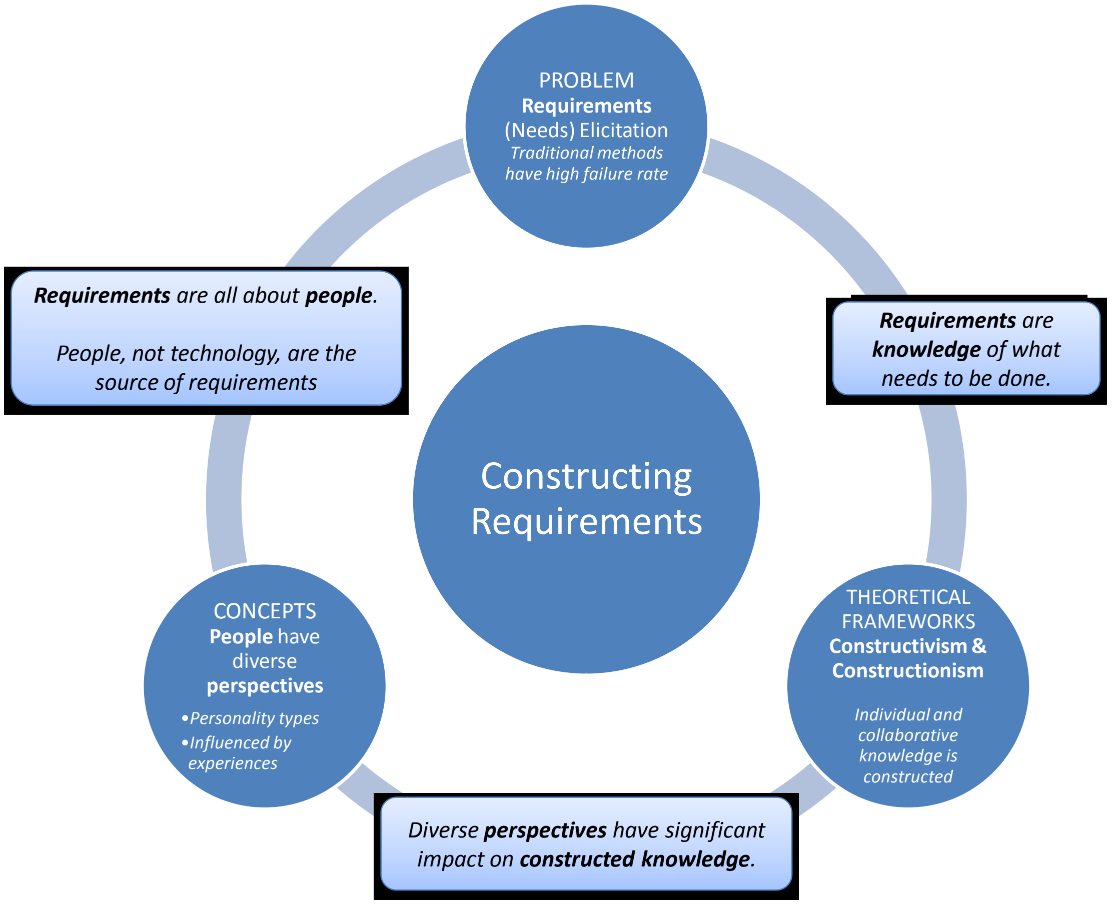

PROBLEM
Requirements
(Needs) Elicitation
Traditional methods have high failure rate
***Requirements** are all about **people**.*
People, not technology, are the source of requirements
***Requirements** are **knowledge** of what needs to be done.*
Constructing Requirements
CONCEPTS
People have diverse **perspectives**
•*Personality types*
•*Influenced by experiences*
THEORETICAL FRAMEWORKS
Constructivism & Constructionism
Individual and collaborative knowledge is constructed
*Diverse **perspectives** have significant impact on **constructed knowledge**.*

Figure 1. Relationships Among the Concepts of Constructing Requirements

Taking these concepts together, I was inspired by the sensitizing concept that information systems requirements elicitation is actually an exercise in constructing knowledge and should be approached as such in order to produce accurate and complete requirements that will yield a project deemed successful by stakeholders. I asked the research questions to gather data about what challenges are encountered by stakeholders to further illuminate this concept. I then analyzed the data looking for either a positivist or constructivist approach, and patterns that indicated if the approach has impact on the accuracy or completeness of the requirements.

I hope this concept will inspire others to explore the value of a constructivist approach to requirements and potentially improve the quality of requirements and consequently the success rates of information systems projects.

Summary

In this chapter, I presented a background for this study to introduce the persistence and significance of challenges associated with eliciting requirements for information systems. Studies show requirements elicitation playing a significant role in a project's success or failure, yet after generations of progress, projects continue to experience low success rates. This chapter also includes the research question and a summary of my perspective as the researcher. The next chapter presents a review of relevant literature.

Definition of Terms

1. **Abduction**: A type of reasoning that begins with the researcher examining inductive data and observing a surprising or puzzling finding that cannot be explained with conventional theoretical accounts. After scrutinizing these data, the researcher entertains all possible theoretical explanations for the observed data, and then forms hypotheses and tests them to confirm or disconfirm each explanation until he or she arrives at the most plausible theoretical interpretation of the observed data. Hence, abduction begins but does not end with induction. Rather, the search for a theoretical explanation involves an imaginative leap to achieve a plausible theoretical explanation. At this point, the researcher may create a new theory or put extant theories together in a novel way. Thus, abduction brings creativity into inquiry and

takes the iterative process of grounded theory further into theory construction. (Charmaz, 2014, p. 341)

2. **Categorization**: The analytical step in grounded theory of selecting certain codes as having overriding significance or abstracting common themes and patterns in several codes into an analytic concept. As the researcher categorizes, he or she raises the conceptual level of the analysis from description to a more abstract, theoretical level. The researcher then tries to define the properties of the category, the conditions under which it is operative, the conditions under which it changes, and its relation to other categories. Grounded theorists make their most significant theoretical categories into the concepts of their theory. (Charmaz, 2014, p. 341)
3. **Code**: The short label the grounded theorist constructs to depict what is happening in a piece of data. Codes sort, synthesize, and most significantly *analyze* the data. Codes connect raw data with the grounded theorist's conceptualization of them. The best codes are short, simple, precise, and analytic. These codes account for the data in theoretical yet accessible terms. Codes vary in their level of abstraction depending on the data, the researcher's theoretical acumen, and the point in the research process. Recoding earlier codes can result in substantially more abstract analytic codes. (Charmaz, 2014, pp. 341-342)
4. **Coding**: The process of taking data apart, defining, and labeling what these data are about. Unlike quantitative researchers, who apply preconceived categories or codes to the data, a grounded theorist creates qualitative codes by defining what he or she sees in the data. Thus, grounded theory codes are emergent. Researchers develop codes as they study and interact with their data. The coding process may take a researcher to

unforeseen areas and research questions. Grounded theory proponents follow such leads; they do not pursue previously designed research problems that lead to dead-ends. (Charmaz, 2014, p. 342)

5. **Concepts**: The analytic conceptualizations that form the components of the developed grounded theory. Concepts are abstract ideas that account for the data and have specifiable properties and boundaries. Grounded theorists construct fresh concepts from inductive data and check and develop them through abduction. For objectivist grounded theorists, concepts are variables abstract of time, place, and people. For constructivists, concepts provide abstract understanding of the studied phenomenon and are situated in the conditions of their production in time, place, people and the circumstances of the research process. Although most scholars view theories as demonstrating relationships between concepts, many grounded theorists focus on developing one concept. (Charmaz, 2014, p. 342)
6. **Concept-indicator model**: A method of theory construction in which the researcher constructs concepts that account for relationships defined in the empirical data and each concept rests on empirical indicators. Thus, the concept is ‘grounded’ in data. (Charmaz, 2014, p. 342)
7. **Constant comparison method**: A method of analysis that generates successively more abstract concepts and theories through inductive processes of comparing data with data, data with code, code with code, code with category, category with category, and category with concept. In the last stages of analysis, researchers compare their major categories with those in relevant scholarly literatures. Comparisons then constitute each state of analytical development. Grounded theorists

use this method to reveal the properties and range of the emergent categories and to raise the level of abstraction of their developing analyses. (Charmaz, 2014, p. 342)

8. **Constructivism**: A social scientific perspective addressing how realities are made. This perspective brings subjectivity into view and assumes that people, including researchers, construct the realities in which they participate. Constructivist inquiry starts with the experience and asks how members construct it. To the best of their ability, constructivists enter the phenomenon, gain multiple views of it, and locate it in its web of connections and constraints. Constructivists acknowledge that their interpretation of the studied phenomenon is itself a construction. (Charmaz, 2014, p. 342)
9. **Constructivist grounded theory**: A contemporary version of grounded theory that adopts methodological strategies such as coding, memo-writing, and theoretical sampling of the original statement of the method but shifts its epistemological foundations and takes into account methodological developments in qualitative inquiry occurring over the past fifty years. Thus, constructivist grounded theorists attend to the production, quality, and use of data, research relationships, the research situation, and the subjectivity and social locations of the researcher. Constructivist grounded theorists aim for abstract understanding of studied life and view their analyses as located in time, place, and the situation of inquiry. (Charmaz, 2014, p. 342)
10. **Customer**: A type of stakeholder who has needs to be met by an information system that will be delivered by a provider.

11. **Focused coding**: A sequel to initial coding in which researchers concentrate on the most frequent and/or significant codes among their initial codes and test these codes against large batches of data. Researchers can then take those codes demonstrating analytic strength and raise them to tentative categories to develop. When the researcher's initial codes are concrete, the researcher can code them by asking what analytic story these codes indicate, and thus arrive at a set of focused codes. (Charmaz, 2014, p. 343)
12. **Faulty Requirement**: An expectation for an information system that has been documented as being required, but has not been accepted by all stakeholders. A potential source of conflict and a high risk for perceptions of project failure.
13. **Induction**: A type of reasoning that begins with study of a range of individual cases and extrapolates patterns from them to form a conceptual category. (Charmaz, 2014, p. 343)
14. **Initial coding**: The early process of engaging with and defining data. Initial coding forms the link between collecting data and developing an emergent theory to understand and account for these data. Through coding, you *define* what is happening in the data and begin to grapple with what it means. (Charmaz, 2014, p. 343)
15. ***In vivo* Codes**: Codes that researchers adopt directly from the data, such as telling statements they discover in interviews, documents, and the everyday language used in a studied site. (Charmaz, 2014, p. 343)
16. **Information System**: A collection of computer software and hardware technologies assembled to address a problem or opportunity.

17. **Line-by-line Coding**: A form of initial coding in which the researcher assesses what is happening in each line of data and what theoretical ideas it suggests. Line-by-line coding is a *heuristic device* to encourage researchers to think analytically about their data and to generate fresh ideas about them. This type of coding encourages active engagement with data and enables researchers to see their data from new standpoints. Researchers conduct line-by-line coding only until they have generated some codes to pursue. Line-by-line coding also serves as an excellent antidote for analytic and writing blocks. (Charmaz, 2014, p. 342)
18. **Memo-writing**: The pivotal intermediate step in grounded theory between data collection and writing drafts of papers. When grounded theorists write memos, they stop and analyze their ideas about their codes and emerging categories in whatever way that occurs to them (see also Glaser, 1998). Memo-writing is a crucial method in grounded theory because it prompts researchers to analyzer their data and develop their codes into categories early in the research process. Writing successive memos keeps researchers involved in the analysis and helps them to increase the level of abstraction of their ideas. (Charmaz, 2014, p. 343)
19. **Positivism**: An epistemology that subscribes to a unitary scientific method consisting of objective systematic observation and experimentation in an external world. Positivism relies on empiricism as the source of generalizations. The goal of positivistic inquiry is to discover and to establish generalizations that explain the studied phenomena and from which predictions can be made. Subsequently, experimentation and prediction can lead to scientific control over the studied phenomena. (Charmaz, 2014, p. 344)

20. **Properties**: The defining characteristics or attributes of a category or concept as ascertained from the researcher's study and analysis of his or her data and codes. (Charmaz, 2014, p. 344)
21. **Provider**: A type of stakeholder who will meet customer needs by providing an information system as a solution.
22. **Requirement**: An expectation about the features of an information system. "A requirement defines what a system is supposed to do, without defining how it is to do it" (Davis, 1993).
23. **Sensitizing concepts**: Give researchers initial but tentative ideas to pursue and questions to raise about their topics. Grounded theorists use sensitizing concepts as tentative tools for developing their ideas about processes that they define in their data. (Charmaz, 2014, p. 30)
24. **Social constructionism**: A theoretical perspective that assumes that people create social reality or realities through individual and collective actions. Rather than seeing the world as given, constructionists ask how it is accomplished. Thus, instead of assuming realities in an external world – including global structures and local cultures – social constructionists study what people at a particular time and place take as real., how they construct their views and actions, when different constructions arise, whose constructions become taken as definitive, and how that process ensues. Symbolic interactionism is a constructionist perspective because it assumes that meanings and obdurate realities are the product of collective processes. (Charmaz, 2014, p. 344)
25. **Stakeholder**: A broad class of individuals who have some interest (a stake) in the success (or failure) of the system in question. (Laplante, 2014, p. 32).

26. **Substantive theory**: A theoretical interpretation or explanation of a delimited problem in a particular areas, such as family relationships, formal organizations, or education. (Charmaz, 2014, p. 344)
27. **Theoretical codes**: Codes that researchers draw on from prior theories or analytic schemes and use to integrate the categories of their analyses. Glaser (2005) offers a loosely organized series of 'coding families' that researchers can use to articulate how their categories, codes and data are related. (Charmaz, 2014, p. 345)
28. **Theoretical sampling**: A type of grounded theory sampling in which the researcher aims to develop the properties of his or her developing categories or theory, not to sample randomly selected populations or to sample representative distributions of a particular population. To engage in theoretical sampling, the researcher must have already developed a tentative theoretical category from the data. When engaging in theoretical sampling, the researcher seeks people, events, or information to illuminate and define the properties, boundaries, and relevance of this category or set of categories. Because the purpose of theoretical sampling is to sample in order to develop the theoretical categories, conducting it can take the researcher across substantive areas. (Charmaz, 2014, p. 345)
29. **Theoretical saturation**: Refers to the point at which gathering more data about a theoretical category reveals no new properties nor yields any further theoretical insights about the emerging grounded theory. (Charmaz, 2014, p. 345)

II. REVIEW OF THE LITERATURE

Rationale for Topic

While there are applications of purely fictional creativity, such as video games and artistic products, it is often the case that a software abstraction or information system is intended to represent real-world scenarios that involve stakeholder expectations and human-imposed constraints. Information systems professionals must develop and document thorough knowledge of the expectations and constraints in order to provide systems that accurately represent the stakeholders' understanding of the scenario. The process of creating knowledge of constraints and operational capabilities of an information system is traditionally known within the computer science field as *requirements analysis* and a few similar names (Neill & Laplante, 2003). The requirements are often recorded in the Software Requirements Specification (SRS) or simply the requirements document. The term *requirements engineering* is sometimes used interchangeably with *requirements analysis*. Requirements *research* using qualitative methods is relatively new in the lexicon (Maiden, 2005). Terms commonly used include requirements *gathering* and also *capturing* requirements, which are both strong indicators that the traditional approach assumes that finite requirements exist and simply need to be collected and coherently documented. There is evidence that *gathering* requirements is no longer considered sufficient and that a more proactive approach is required that involves "constant probing, drawing on numerous sources and perspectives, and truly understanding the business overall and the needs" of the stakeholders (HIMSS, 2008, p. 48).

Formal models to describe the software (or system) development lifecycle (SDLC) evolved over the years to standardize processes and attempt to make outcomes more predictable. The earliest SDLC practices have come to be known as the *waterfall* model, signifying the one-directional cascade of events. Royce reflected on the practices common at the time, pointing out their weaknesses and how failure was invited (Royce, 1970). The waterfall model consists of these sequential phases:

1) Requirements
2) Analysis
3) Design
4) Coding
5) Testing
6) Operations

Royce made several recommendation to improve the waterfall model, including introducing iteration between phases as well as having at least two iterations of the complete lifecycle where the first pass results in a smaller scale simulation of the actual project. In practice, the small scale simulation completed first would be either a prototype or a pilot phase which would generate lessons learned and improved understanding for the team. Royce summarized:

> Without this simulation the project manager is at the mercy of human judgment. With the simulation he can at least perform experimental tests of some key hypothesis and scope down what remains for human judgment, which in the area of computer program design (as in the estimation of takeoff gross weight, costs to

complete, or the daily double) is invariably and seriously optimistic. (Royce, 1970, p7)

Despite the well documented shortcomings and warnings, the waterfall model continues to be followed by some practitioners today, but is regarded by many as ineffective, especially on non-trivial projects (Leffingwell, 2011). One reason the waterfall model is considered ineffective is lack of feedback or iteration. In a pure waterfall model, there is no mechanism to deal with change that may occur following the conclusion of the requirements phase. In other words, the model assumes the requirements established up front are accurate and no new knowledge will be gained during subsequent phases that would change the requirements. This is a very positivist approach. Considering requirements to be a form of knowledge, locking in requirements at the beginning of a project implies that no learning will occur during the later phases.

Modifications and alternatives to the waterfall model later emerged that introduced procedural mechanisms to deal with change. A modified waterfall model "uses the same phases as the pure waterfall, but is not based on a discontinuous basis. This enables the phases to overlap when needed" to enable more flexibility (Munassar & Govardhan, 2010, p. 97). The *spiral* model was introduced by Boehm in 1988. This model kept the requirements phase early in the lifecycle, but did include an element of constant feedback to allow for change. Additional models emerged that added more attention to change that occurs throughout the information system project. Rapid Application Development (RAD) was popularized in the 1990s by James Martin. Another model is the Rational Unified Process (RUP) from the late 1990s. RUP recognizes the overlap across the lifecycle phases of inception, elaboration, construction, and transition.

Requirements are addressed throughout the lifecycle, not only during the initial phase (Leffingwell, 2011).

One of the recent models for an information system lifecycle began in 2001 with *The Agile Manifesto* which has become a very popular set of guiding principles known simply as *Agile* or sometimes *agile*, without capitalization. The text of the manifesto is brief and included here as well as in Appendix A:

> We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:
>
> Individuals and interactions over processes and tools
>
> Working software over comprehensive documentation
>
> Customer collaboration over contract negotiation
>
> Responding to change over following a plan
>
> That is, while there is value in the items on the right, we value the items on the left more. (Agile Alliance, 2001)

These brief guidelines have inspired many in the field to change attitudes and processes, yet information systems continue to fail at high rates over a decade later. There is more improvement to be done.

The spirit of Agile manifests in *Scrum,* which is a project management framework popular among software professionals (Leffingwell, 2011). Although popularized by software developers, Scrum is a flexible framework that can be applied to other types of complex problems or projects, such as deploying an information system. Scrum has been used since the early 1990s as a process framework to manage complex projects (Schwaber & Sutherland, 2013). According to Laplante,

> Scrum, which is named after a particularly contentious point in a rugby match, enables self-organizing teams by encouraging verbal communications across all team members and across all stakeholders….Scrum encourages self-organization by fostering high-quality communications among all stakeholders. In this case, it is implicit that the problem cannot be fully understood or defined (it may be a wicked problem). And the focus in Scrum is on maximizing the team's ability to respond in an Agile manner to emerging challenges. (Laplante, 2014, p. 171)

The Scrum framework has specific terminology to articulate the processes, artifacts, and people involved. Since terms from Scrum are used later in this report, *The Scrum Guide* is provided as Appendix B as an overview and reference.

Whereas the waterfall model above addressed requirements only at the beginning of the project, Agile approaches address requirements continually (Laplante, 2014). In Scrum, requirements are documented in the product backlog which is managed by the product owner. "The product backlog is an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product" (Schwaber & Sutherland, 2013, p. 12). The findings of this study are relevant to Scrum product owners who are responsible for ensuring the product backlog contains the optimal set of requirements for the involved stakeholders.

From my experience, information systems projects often include multiple sponsors and other stakeholders, each having a different set of expectations and perspectives based on individual experiences. Furthermore, requirements evolve over time as new constraints emerge in response to change. It is often true that "by the time an information system is deployed, the targets at which it was aimed have moved so much so as to render it useless in the majority of cases" (Galal & Paul, 1999, p. 92). Personnel changes among the stakeholders have especially high impact on requirements because each new stakeholder brings a lifetime of knowledge, a new perspective, and new priorities to be assimilated into the collective understanding. One of my sensitizing concepts for this research project is the impact of stakeholder turnover, considering the team of stakeholders as an organic rather than mechanistic entity (Burns & Stalker, 1961).

Pondering these characteristics led me to speculate that requirements are actually *constructed*, following the way knowledge is constructed according to Piaget (1970). If true, then information systems professionals would benefit from applying a constructivist approach to requirements. Such an approach will inherently improve the consensus perception of accuracy of the requirements by reducing the gaps between the expectations of sponsors and visions of the system developers. This pondering becomes a sensitizing concept for my research which seeks to understand the specific challenges that cause the gaps to occur.

Galal and Paul (1999) do not speak to constructivism directly, but they do present a qualitative approach to requirements engineering based on grounded theory. Their motivation for this approach is that requirements are inherently dynamic and contextual combined with the fact that information systems projects take time. They point out that requirements tend to change, are predictive, are sensitive to context, and may prove to be invalid predictions. Rather than think of requirements as fixed, Galal and Paul envision requirements as scenarios of dynamically linked concepts and categories developed by qualitative data analysis. Data are gathered, or knowledge constructed, using semi-structured protocols, observation, documentary analysis, and similar investigative techniques.

Hazzan and Dubinsky do specifically relate constructivism to software engineering. They clearly, and correctly, state that “software development is a learning process” and devote a chapter to explaining the relationships between Agile software development and constructivism (2008, p. 139). For example, they explain that the short duration iteration of Agile “guides the customer, as well as the team members, in a gradual process of knowledge construction” (2008, p. 142). The findings from this research contribute additional data that strengthen the concept that knowledge about information systems requirements is indeed constructed.

While requirements engineering certainly benefits from engineering principles, the strong learning aspect of elicitation indicates the need to leverage principles from adult education, or andragogy. Davey and Cope point out

> The process of requirements elicitation is often seen as a process of mutual education of consultant and client…. A strong theme of education theory is that learning takes place by interaction between people. This view leads to investigations of a technique called collaborative elaboration, in which two or more people interact in a conversation in a structured way. (2008, pp. 547-548)

Knowles wrote that adults need to understand why they need to learn something (1984). Collaboration is important both to provide motivation as well as to exchange information in order to construct knowledge. Simply gathering requirements without collaboration and knowledge construction lacks inspiration, similar to memorizing multiplication tables. Adults do not readily learn by simply being told. They must be involved as collaborative inquirers (Knowles, 1980). This is true for learning and constructing requirements.

My motivation for the study includes knowing first-hand that many practicing and graduating information systems professionals are weak in requirements processes. Although “every system of consequence needs good requirements” (Orr, 2004, p. 71), requirements are often neglected, especially by information systems practitioners claiming to follow newer methods of Agile development. “Rather than understanding the user’s business and information needs, some Agile approaches attempt to give users what they say they want as quickly as possible” (Orr, 2004, p. 71). Information systems developers are so enthusiastic about applying their creativity to build something that they fail to take the time to thoroughly understand the requirements of what they are being invited to build. “The requirements engineer’s job, then, is much like that of a map maker trying to create a good map from all those individual viewpoints” (Orr, 2004, p. 72).

The result of ignored or insufficient requirements construction is wasted time, effort, and money as well as disappointment by stakeholders involved. There must be a better way. Educating information systems developers on the challenges faced during requirements construction may help them realize they are indeed beginning to build something during the requirements phase, they are building knowledge of the problem and the candidate solutions. The remainder of this chapter will provide additional background from the literature on the challenges involved with information systems requirements.

Importance of Understanding and Communication

Information systems development is “arguably the most intensive of all engineering disciplines as regards human communications” (Lang & Duggan, 2001, p. 161). Ian Graham explains that “requirements engineering is about communications” (2003, p. 100). Ultimately, “the success or failure of software projects is sometimes attributed to people communication issues (Hazzan & Dubinsky, 2008, p. 61). A requirements engineer “must be able to understand and structure problems, construct and test models of these problems and their possible solutions, and communicate” (Graham, 2003, p. 99). Although Graham chose *communicate*, the stronger *educate* is more appropriate. The requirements engineer must facilitate construction of knowledge with both the business problem experts and technology solution experts in order to fill knowledge gaps, raise awareness of different perspectives, and bring all parties toward a common understanding of both problems and solutions. Many information systems practitioners tend to be uncomfortable with or incapable of communicating effectively with non-technical stakeholders. “Above all, requirements must be communicable” (Lang & Duggan, 2001, p. 163). Graham also points out “it’s amazing how many times I have seen development teams impose their own thinking on a problem and totally fail to understand what the customer is telling them – or more often – implying” (2003, p. 100).

Knowledge creation and sharing between information systems professionals and stakeholders is not the only communication issue. Information systems projects often involve multiple professionals who must collaborate with each other. Sinha, Sengupta, and Chandra (2006) utilized literature sources as well as qualitative research methods to determine the challenges faced by distributed teams. “Communicating and managing requirements in a distributed setting was one of the concerns the practitioners expressed most often" (Sinha et al., 2006, pp. 52-53). Teams often rely on communication tools to facilitate collaboration. Their research shows that collaboration tools should offer: informal ad hoc collaboration services, formal structured collaboration services, notifications for changes and approvals, and knowledge management features. "Ad hoc management processes that suffice for requirements management in collocated projects don't scale when development is distributed" (Sinha et al., 2006, p. 61). The study is largely a case study of a tool, *EGRET*, created by the authors for distributed requirements management. "Collaborative technologies such as EGRET can make a true impact only when a healthy culture of collaboration exists" (Sinha et al., 2006, p. 61). This point is an important reminder that simply buying a tool will not necessarily inspire cultural change. Leadership, including professional development educators, must invest time and effort to nurture a culture of collaboration and open communication. People must want to communicate before the technology can help.

Modeling Tools

Requirements researchers have developed techniques to formalize the requirements process. Knowledge Acquisition in Automated Specification (*KAOS*) and *i** are two goal modeling techniques (Maiden, 2005). KAOS is described as "one of the most important approaches to requirements engineering of the last ten years" (Heaven & Finkelstein, 2004, p. 10). "Goal modeling research has increasingly led analysts to specify requirements in terms of a system's actors and goals rather than its processes and objects…. We now model systems in terms of their wider organizational goals and link software solutions to business needs more effectively than in the past" (Maiden, 2005, p. 104).

Heaven and Finkelstein (2004) present a profile in Unified Modeling Language (UML) to support the Knowledge Acquisition in Automated Specification (KAOS) approach to goal-oriented requirements acquisition. Their intent is to enable the use of industry standard UML-based tools and graphical notations to aid the implementation of KAOS, rather than relying on non-standard tools and notations. The authors go into great detail with ample examples showing how to represent KAOS concepts in the UML. However, I believe using KAOS in a requirements document is counterproductive as it is not friendly to non-technical readers. Graham goes so far as to say "communicating with users using UML diagrams—forcing them to learn your language—is staggeringly arrogant" (Graham, 2003, p. 100).

While these tools may help requirements experts communicate with precision among their peer group, they do not address the challenges of eliciting the right set of requirements from stakeholders who are unfamiliar with these complex tools and notations.

Symbols vs. Natural Language

Written communication is important when dealing with requirements construction. Power and Moynihan describe the requirements document as "a means of communication between the developers and the other stakeholders" (Power & Moynihan, 2003, p. 86). This flow of information is clearly critical to ensure the developed system meets the intended needs. In fact, one of the principal roles of the requirements document is to establish the basis of agreement between the parties involved. Yet, after decades of evolution of information system development, there remains no general agreement on the contents of a requirements document (HIMSS, 2008). Lang and Duggan (2001) point out that communication of requirements is particularly problematic due to the high risk of project failure associated with erroneous requirements. An effective requirement is *unambiguous*, meaning there is only one possible interpretation. A requirement must also be *understandable* by all readers with a minimum of explanation. Lang and Duggan (2001) believe the requirements document, or system requirements specification (SRS), must contain natural language, or plain English, descriptions as well as supplemental structured notations and diagrams to be effective. "Natural language descriptions is the only technique that is generally understandable by all potential users of the SRS" (Lang & Duggan, 2001, p. 163). However natural language is susceptible to ambiguity and inconsistency, so more structured notations should also be used. "To get great requirements, you don't have to be a genius or a magician. You just have to know your users' outputs and constraints and then document those needs in a way they understand" (Orr, 2004, p. 73).

One formal guide related to requirements elicitation is The International Organization of Standards publication *Systems and Software Engineering – Life Cycle Processes – Requirements Engineering* (2011). This document presents a formal approach to requirements engineering and is written for a technical audience. The standard prescribes a sequence of steps, with some iteration, to elicit requirements. The second step prescribed and its accompanying note is:

Elicit stakeholder requirements from the identified stakeholders.

NOTE Stakeholder requirements describe the needs, wants, desires, expectations and perceived constraints of identified stakeholders. They are expressed in terms of a model that may be textual or formal, that concentrates on system purpose and behaviour, and that is described in the context of the operational environment and conditions. A product quality model and quality requirements, such as found in ISO/IEC 9126-1 and ISO/IEC 25030, may be useful for aiding this activity. Stakeholder requirements include the needs and requirements imposed by society, the constraints imposed by an acquiring organization and the capabilities and operational characteristics of users and operator staff. It is useful to cite sources, including solicitation documents or agreements, and, where possible, their justification and rationale, and the assumptions of stakeholders and the value they place on the satisfaction of their requirements. For key stakeholder needs, the measures of effectiveness are defined so that operational performance can be measured and assessed. If significant risks are likely to arise from issues (i.e. needs, wants, constraints, limits, concerns, barriers, factors or considerations) relating to people (users and other stakeholders) and their involvement in or interaction with a system at any time in the life cycle of that system, recommendations for identifying and treating human-system issues can be found in ISO PAS 18152, A specification for the process assessment of human-system issues. (ISO, 2011, p. 20)

The standard continues with the following guidance:

> Requirements elicitation is an iterative activity. Consider several different techniques for identifying requirements during the elicitation task to better accommodate the diverse set of requirements sources, including:
>
> - Structured workshops with brainstorming
> - Interviews, questionnaires
> - Observation of environment or work patterns (e.g., time and motion studies)
> - Technical documentation review
> - Market analysis or competitive system assessment
> - Simulations, prototyping, modelling
> - Benchmarking processes and systems
> - Organizational analysis techniques (e.g., Strength – Weakness – Opportunity - Threat analysis, product portfolio)
>
> System stakeholders will be authoritative sources for requirements of the system that represent their interests or area(s) of expertise. However, they usually are not familiar with how to transform their expertise into well-formed requirements statements. In addition to these human sources of requirements, important system requirements often are imposed by other systems in the environment that require some services of the system, or act to constrain the system, or even from fundamental characteristics of the application domain. There may also be safety or other regulatory constraints that drive system requirements. (ISO, 2011, p. 22)

This guidance for requirements elicitation is broad enough to allow a constructivist approach, but does not go so far as to recommend it. For example, the suggestion of "structured workshops with brainstorming" encourages collaboration but the document does not provide guidance on facilitating learning during the workshop.

In contrast to the complex symbols and notations developed for precise communication among requirements experts, practitioners of Agile methods use stories to convey needs. "User stories are the agile replacement for most of what has been traditionally expressed as software requirements statements… and they are the workhorses of agile development" (Leffingwell, 2011, p. 57). Stories are intended to simplify requirements processes rather than adding complexity. "We need to make sure that the team's requirements artifacts are the simplest thing that could possibly support the needs of *all* stakeholders" (Leffingwell, 2011, p. 55).

Pine and Barrett (2005) report on a qualitative study they conducted involving surveying a small group of practitioners about what verbal and written communication skills should be strengthened in undergraduate software engineering courses. The results are organized as suggestions from the respondents for verbal communication and writing assignments that will help prepare students to communicate effectively in the workplace. The assignments involve forms of communication that are common in the field and are often challenging for some information systems professionals. Suggested verbal assignments include: incident presentations, sales briefings to customers, technology interpretations, help desk sessions, and the elevator speech. Writing assignments identified include: requirements documents, design documents, user documentation, budget and time progress reports, requests for quote, quick reference guides, product comparisons and recommendations, incident reports, executive summaries, rationales for development, short memoranda, self-evaluations, and personnel evaluations.

Power and Moynihan (2003) present a theoretical framework for requirements documents based on various elements commonly found in such documents and various situations. The framework is the result of a qualitative study involving 30 experienced practitioners participating in semi-structured interviews. The authors analyzed the collected data using grounded theory and a research tool, ATLAS.ti. The authors used this approach to construct their theory from the available data which yielded seven elements prevalent in requirements documents, seven situation types, and eight sources of requirements. They are careful to point out that their theory is a framework and the results would likely change with different input data.

Despite the existence of complex graphical notations and modeling tools for requirements, plain English narration dominates. Neill and Laplante (2003) found that 51% of respondents in their study reported use of natural language representation of requirements while only 7% reported using a formal notation. Furthermore, 33% indicated using no formal methodology at all to analyze and model requirements. Neill and Laplante (2003) concluded that formal methods are rarely used and that ad hoc practices do not impact end-product quality. Although rarely used, the study did mention some popular approaches including group-consensus-type techniques, User-Centered Design, Joint Application Design, focus groups, and structured analysis and design. In the grounded theory approach from Galal and Paul (1999), requirements are gathered using basic qualitative methods and the resulting document is a natural language, or plain English, narrative consisting of scenarios supported by very simple supporting diagrams.

Like most graphical notations, KAOS uses shapes and lines to represent concepts and relationships. While such a notation may be used by information systems experts to unambiguously represent complex scenarios, the diagrams are not useful for communication with non-technical stakeholders. Since the requirements document is the basis for agreement among developers and customers, I see the inclusion of KAOS diagrams as more problematic than helpful. KAOS diagrams might be appropriate in an architecture or design document which is intended to model a solution and be interpreted by technical experts, but not in a requirements document which must be understood by a wider audience. It is interesting that some examples used by Heaven and Finkelstein (2004) show the KAOS syntax taking up more space on the page than the equivalent informal narrative description.

This literature review summarizes only a small portion of the material available describing information system failures caused by faulty requirements. I believe information systems professionals will be better prepared to facilitate constructive requirements workshops if equipped with a better understanding of the challenges they may face.

Summary

This chapter presented a review of relevant literature to support this study. First presented was literature supporting the rationale for choosing this worthy topic. Also included were sources of foundational material that will help readers understand the terminology and concepts that follow in later chapters, such as the waterfall and Agile lifecycle models as well as the focus on modeling tools and symbolic presentations common among information systems requirements. The next chapter presents the methodology for this study.

III. METHODOLOGY

Introduction

As described in Chapter I, the objective of this study is to identify what challenges are encountered by stakeholders when identifying requirements for information systems. The benefit is that once identified, these challenges may be addressed by practitioners as well as future researchers to improve the rates of success for information systems projects. This chapter provides details of the methodology followed to conduct the study.

Approach

The method for this study is to collect, analyze, and interpret rich qualitative data from a set of knowledgeable practitioners in order to develop a grounded theory. My approach is a combination of interpretivism and constructivism, as described by Charmaz. She explains

> Interpretive theory calls for the imaginative understanding of the studied phenomenon. This type of theory assumes emergent, multiple realities; indeterminacy; facts and values as linked; truth as provisional; and social life as processual…. The interpretive turn in theory has gained attention as social constructionist principles gained advocates among diverse scholars, particularly since the 1960s. This theoretical approach emphasizes practices and actions. (2006, pp. 126-127).

During analysis and interpretation of the collected data, I draw on my own previous experience and foundation of knowledge to construct new meaning which influences my understanding of the emerging theory. Charmaz describes

> A constructivist approach places priority on the phenomena of study and sees both data and analysis as created from shared experiences and relationships with participants…. Constructivist grounded theory lies squarely in the interpretive tradition…. We do so from as close to the inside of the experience as we can get, but realize that we cannot replicate the experiences of our research participants. A constructivist approach means more than looking at how individuals view their situations. It not only theorizes the interpretive work that research participants do, but also acknowledges that the resulting theory is an interpretation. The theory *depends* on the researcher's view; it does not and cannot stand outside of it. (2006, p. 130).

Just as Charmaz explicitly links interpretive and constructive approaches, both are relevant to this research study and apply in a complementary fashion. The theory generated by this study is grounded in my interpretation of the collected data, which is influenced by my constructed understanding from previous experiences.

Conceptual Framework

Maxwell writes that a "conceptual framework is a theory, however tentative or incomplete it may be…This may also be called the 'theoretical framework'… for the study" (2005, p. 34). Charmaz distinguishes frameworks for grounded theory from quantitative research with

> We do not use theories for deducing specific hypotheses before data-gathering….In contrast, in a grounded theory study you put your sensitizing concepts and theoretical codes to work in the theoretical framework….

Sensitizing concepts account for your starting point. Theoretical codes can help you explain how you conceptualize the arrangement of key ideas. (2006, p. 169)

My conceptual framework begins with the sensitizing concept that information systems requirements are *constructed* rather than *gathered*. In fact, my approach to requirements is encouraged by Maxwell's approach to the conceptual framework as "something that is *constructed*, not found. It incorporates pieces that are borrowed from elsewhere, but the structure, the overall coherence is something that *you* build, not something that already exists ready-made" (2005, p. 35). In my experience, during the early stages of many information system projects, meetings are held among stakeholders to discuss and negotiate the envisioned system. I have observed these requirements meetings to be workshops where the work being unwittingly accomplished by the participants is the construction of knowledge and common understanding among the participants, each of whom is contributing what they have to offer, and expecting something greater to be achieved. An analogy might be a requirements *pot luck* gathering, where the resulting feast is dependent on the contributions of individuals and, after it is over, each participant will have an opinion regarding whether or not their appetite was satisfied.

This holistic view is supported by activity theory as applied by Korpela, Mursu, & Soriyan. They "regard work activity as a systemic entity comprising a number of elements which must fit together to some extent" (2002, p. 112). Requirements elicitation is a work activity involving multiple stakeholders each with individual expectations about the system to be designed. These expectations must be reconciled to determine whether and how they can fit together. The excerpt below applies by translating the actors as stakeholders, the shared object as the set of requirements, and the output as the resulting information system:

> The actors perform their individual actions of work on the shared object through mediating instruments or means of work which can be material (technology) or immaterial (language, skills, theories)…. Each actor may have his or her own means, or some of the means can be shared. The individual actions taken together form the process through which the object is transformed to the output. In order to merge the individual actions into a collective activity, there needs to be some form of coordination between them, mediated by the means of coordination and communications; for instance rules, division of labor, timetables, meetings, phone calls, and so forth. (Korpela et al., 2002, pp. 112-113)

The coordination and communications required serves as a framework to guide participants toward a common understanding of the desired outcome and to manage expectations. Appendix C is a summary of my conceptual framework.

Methodology

This qualitative research study is guided by grounded theory according to Charmaz. This methodology was chosen for its "emphases on examining processes, making the study of action central, and creating abstract interpretive understandings of the data" as well as having "flexible guidelines, not methodological rules, recipes, and requirements" (Charmaz, 2006, p. 9).

A qualitative methodology was selected in order to generate detail-rich data from participants. This methodology also enabled me to play the role of both researcher and practitioner in order to interact with participants in meaningful ways (Bloomberg & Volpe, 2008).

I followed an iterative grounded theory process based on guidelines from Charmaz (2006, p. 11). The steps followed were

1) Identify the research problem and research questions
2) Identify the sensitizing concepts
3) Perform data collection and initial coding
4) Analyze initial codes to form tentative categories
5) Continue data collection with focused coding
6) Analyze codes and tentative categories to form conceptual categories
7) Perform theoretical sampling seeking new data based on categories
8) Adopting certain categories as theoretical concepts
9) Reexamination of earlier data (loop back to step 3 and refine codes and categories)
10) Diagramming concepts
11) Writing the first draft, including referring back to step 5 and refining categories.

Data

Data for this study came from interviews, literature, and observations during personal experience. I selected participants to be interviewed for this study using purposeful, theoretical sampling to obtain data applicable to the emerging categories and theory following Charmaz explanation that "Theoretical sampling means seeking pertinent data to develop your emerging theory" (2006, p. 96). The initial participant was purposefully chosen being known by me to have broad experience with information systems requirements and ability to articulate deep, genuine perspective. Subsequent participants were selected based on being referred by a participant or suspected by me to have desirable experience relevant to categories emerging from the data with each interview. Referrals from other participants occurred at the end of interviews and were inspired by the questions asked during the interviews. For example, one early interview generated the unanticipated tentative category of diverse perspectives and at the end of that interview came a referral to another participant who might have relevant experience. Five participants were selected from my professional colleagues based on my knowledge of their involvement with relevant information systems projects. I asked each participant to identify additional candidate participants they know to have relevant experience based on the interview questions and discussion. Two of these referrals led to participants I did not previously know. This process enabled inclusion of participants who were previously unknown to me and improved the richness of the data and variety of perspective, while also being driven by the collected data. The number of participants was determined by saturation as defined by Charmaz to mean “when gathering fresh data no longer sparks new theoretical insights, nor reveals new properties of your core theoretical categories” (2006, p. 113). Another factor that influenced the end of sampling was that enough data had been gathered to support the emerging theory.

Participants were purposefully selected from these general target populations:

- practitioners with experience sponsoring information systems projects and specifying requirements

- practitioners with experience facilitating requirements elicitation efforts in order to deliver solutions
- practitioners with experience participating in requirements workshops as providers of solutions based on requirements

Another source of relevant data was discovered in the form of a book consisting of a collection of documented critical incidents. *H.I.T. or Miss: Lessons Learned from Health Information Technology Implementations* (Leviss, 2010) contains 17 case studies highlighting specific failures of healthcare information technology projects. Some of the challenges documented by Leviss support the findings from my research. These critical incidents provided a nice complement the interview data collected.

During the period of this study, I also had the opportunity to directly engage as a participant/researcher in a large Information System project that involved determining requirements. The project began after the interviews were completed and during the analysis of the data. I used this experience as an opportunity to directly observe some of the challenges that had emerged from my interview data. Observations from participating in this recent project contributed to my overall collection of professional experience as a practitioner. The result was an increased confidence in the trustworthiness and relevance of my collected data and the emerging grounded theory.

Participants

In order to obtain the most accurate and insightful data possible for this study, participants were assured confidentiality of their contributions. This section provides summary demographic information about the population of participants interviewed and safeguards personally identifiable information that could be used to attribute specific quotes to specific individuals. In accordance with American Psychological Association Style guidance for interviews, in order to maintain confidentiality the remarks made by research participants are not cited (2014). Multiple sources are referenced for each challenge demonstrating patterns of recurrence. The focus of this study is on the challenges commonly experienced across a variety of information systems projects, not the identities of the individuals who experienced them. Subsequent research with consenting participants could explore relationships between individuals and the challenges they encountered.

Eight private interviews were conducted between January and November 2013 yielding information from 23 critical incidents. Each interview lasted at least 60 minutes and the audio was recorded for later transcription. The interview participants are described in Table 1 while maintaining confidentiality of the participant identities. The order of the table matches the order of the interviews.

Table 1
Summary of Interview Participants

Participant Order	Fields of Experience, Primary and Secondary	Roles	Years of Experience	Critical Incidents Described
1	Government and Healthcare	Customer and provider	Over 20 years	3
2	Government and Healthcare	Customer and provider	Over 25 years	3
3	Insurance and Government	Customer and provider	Over 20 years	3
4	Commercial and Government	Provider	Over 35 years	2
5	Government and Healthcare	Customer and provider	Over 25 years	3
6	Government and Healthcare	Customer and provider	Over 30 years	3
7	Government and Healthcare	Provider	Over 20 years	3
8	Insurance and Commercial	Customer	Over 30 years	3

GENDER:

- Four female
- Four male

BACKGROUND:

- All have over 20 years of experience across multiple large information systems projects.
- Three have extensive experience in federal government.
- One has experience as both contractor and government employee.
- Three have extensive experience with large corporations delivering services to federal and state government customers.
- One is retired while the others remain professionally active.
- Five have been professional colleagues of the researcher for over 5 years.
- One is a personal friend but not a professional colleague.
- Two were referred by other participants and previously unknown to the researcher.

Data Collection Techniques

The foundation for data collection is Critical Incident Technique (CIT) as described by Flanagan (1954) and expanded by others (Butterfield, Borgen, Amundson, & Maglio, 2005). While CIT has been applied to a wide variety of qualitative studies over the past half-century, its roots are in analyzing the reasons for success or failure of a process and thus CIT seems especially appropriate for this study. Oaklief describes an effective incident as one "which helps us to do a job well" and an ineffective incident as "one which causes a delay or failure and may prevent the job from being completely satisfactory" (1976, p. 8). Similar phrases could be used to describe effective requirements and ineffective or faulty requirements.

I conducted interviews using a semi-structured interview guide with open-ended interview questions that stimulated discussion centered on the research questions while welcoming and exploring unanticipated responses. The pre-determined questions established the semi-structured framework for each interview while the openness enabled focus on the categories that continually evolved with each interview and the continually emerging theory. The initial interview guide is shown in Appendix D and evolved slightly to the final interview guide in Appendix E. The interviews were digitally recorded with minimal written notes taken during the interview to avoid creating distractions for either the participant or the interviewer (Hanson, 2012). Analysis of the data commenced immediately during the first interview and continued throughout the study with constant comparison as described by Charmaz (2006). Data comparisons during initial coding identified differences and similarities. Later, comparisons across multiple incidents and across multiple participants enabled refinement of the codes and categories.

I personally conducted each interview privately with each participant in a mutually agreeable and comfortable place. Expected duration of each interview was approximately 60 minutes while two interviews extended to 90 minutes. I explained that I was conducting research for my dissertation and provided a copy of the Informed Consent form for signature before each interview. The Informed Consent form is represented in Appendix F.

Each participant was prompted to provide three critical incidents, one at a time, with the same interview questions asked about each incident. The first incident prompt was: Think of the *most recent* time when you were involved with an information system that was either purchased or developed and did not meet the requirements. This prompt attempted to help the participant recall the most recent incident which may benefit from having the most accurate memory of facts. Flanagan (1954) states that data collected from observations made previously and reported from memory are usually satisfactory when the incidents are recent. Oaklief points out that "collection of recent incidents gives credence to the reporting of actual happenings and behavior" (1976, p. 12).

Following the prompt, the following interview questions were asked:

1. Tell me about the project. *(Validates the participant has an appropriate incident in mind, and also provides background that may be further examined later.)*
2. What factors led to the requirements not being met?
3. What could have been done differently to improve the likelihood of success?
4. What was your role on the project? *(Clarifies the participant's perspective)*
5. How did the project turn out?

The second prompt for an incident was: Think of another time when you were involved with an information system that was either purchased or developed and did not meet the requirements and had the *most significant impact* on the organization. This prompt was intended to elicit a second incident from each participant by focusing on significance rather than recent occurrence. Once the participant had an incident in mind, the same interview questions were asked as for the first incident prompt.

The third and final prompt for an incident was: Think of another time when you were involved with an information system that was either purchased or developed and did not meet the requirements and generated your feeling of "*here we go again*" during the requirements phase. This prompt targeted an incident that may be memorable for the participant because of repeated behaviors or results. The same interview questions were asked as for the first and second incident prompts, plus one final question of: What steps were taken to avoid repeating mistakes? This final question is intended to be one last attempt at revealing unanticipated results which may be further explored.

Each interview was captured using a digital audio recording device and each recording transcribed by me into a text document using Microsoft Word. Each interview was stored in a separate computer file. Audio files and transcribed documents were organized into one electronic folder for each participant. I secured the collection of audio files and transcribed documents. Identities of participants remain confidential and are known only to me. This study was submitted to the Texas State University Institutional Review Board and assigned identifier #EXP2012Z6144 with exempt status granted on December 3, 2012. Data collected from participants were under my control or stored in a locked cabinet at all times to further protect confidentiality and integrity.

Data Analysis

Each recorded interview was transcribed promptly to leverage fresh memory, to adjust questions before future interviews, and to make the effort less onerous (Hanson, 2012). Initial coding and constant comparison of the data began during the first interview and continued throughout the study (Maxwell, 2005). Even as the first participant spoke about the first critical incident, constant comparison had begun as I compared new data with data revealed earlier in the conversation and made mental and written notes about preliminary codes related to topics of interest. This constant comparison continued across critical incidents, across interview participants, across transcription sessions, and throughout analysis of the data.

During transcription, words and short phrases that immediately emerged as preliminary codes were quickly noted in the margin to distinguish them from the body of data (Saldaña, 2009). The Microsoft Word feature of entering comments in the right margin was used to tag these preliminary codes for later analysis. This immediate capture of initial codes during transcription allowed me to benefit from the fresh memory of the participants tone and non-verbal expressions that helped clarify the intended meaning.

Following transcription, each interview was printed on paper, including any preliminary codes in the margin, and read with pen and highlighter in hand. For each critical incident, line-by-line coding for topics and *in vivo* coding was used. Initial codes were written by hand in the margin of the paper. Some *in vivo* codes were also highlighted in the body of the document in addition to being written in the margins. The initial codes are listed in Table 2 and also in Appendix G in a sample memo.

Table 2
Initial Codes

Initial Codes Alphabetized
• Accountability • Assumptions • Blame • Communication Skills • Conflicting Priorities • Constructivism/Positivism • Culture/Background • Deception • Disappointment • Distractions • Ego/Arrogance/Overconfidence/Inexperience • Existing vs. Custom Solution • Expectations (unstated) • Gender • Incentive • Interpretation/Understanding • Lack of commitment • Lack of Respect • Multiple Stakeholders (2many/2few) • Omissions/Completeness • Perfection vs. Good Enough • Postpone Conflict • Power/Authority/Empowerment/Control • Time Pressure • Timing of problem discovery • Top Down • Trust (too little & too much) • Turnover/Perspectives • Vague • Waste (common result, fuels trust issues)

Open coding with constant comparison from the concept-indicator model was applied to the transcribed data in order to identify codes and categories based on recurring themes. Following initial coding, focused coding was performed to organize the initial codes into preliminary categories and to examine the most significant and interesting concepts that emerged. During this phase, I used Microsoft Excel worksheets to organize and analyze the data because I am comfortable with using that tool to analyze and sort data. Charmaz (2014) describes this process as memo-writing and points out that "no single mechanical procedure defines a useful memo. Do what is possible with the material you have" (p. 171). Samples of my Excel worksheet memos are included in Appendix G.

Focused coding and analysis involved sorting and grouping the initial codes into categories. The categories that emerged from my analysis are change, communications, and knowledge. Each category will be covered in Chapter 4.

Another type of memo-writing also occurred during this study. While continuing to analyze codes and refine categories, opportunities arose for me to submit abstracts of my preliminary work for possible presentations. The opportunity to present my work, pressure of approaching deadlines, and encouragement from my dissertation chair, all provided helpful motivation to intensify my analysis in narrative form. One of my efforts was successful and I was invited to, and did, present my preliminary work at a local chapter meeting of the American Health Information Management Association. Although far from polished, that first public exposure of my research was very liberating and the feedback was encouraging. Much of the writing for the abstracts and presentation is now scattered throughout this dissertation. Charmaz writes "What is important is to get things down on paper and stored in your computer files. Keep writing memos however you write and in whatever way advances your thinking" (2014, p. 165). For me, memos in the form of Excel worksheets, preliminary abstracts, and presentation materials were helpful. One of the abstracts is provided as a sample memo in Appendix G.

Finally, theoretical coding was applied to integrate the organized data into a theory. The theory that emerged from the coded data is grounded in the categories of challenges that emerged across the various critical incidents. The theory is presented in Chapter 5.

As expected, there were some minor adjustments to my methodology over the course of the study. For example, the interview questionnaire was updated after the second interview to reflect a more comfortable and conversational flow. To help the participants organize their thoughts, I added an introduction to explain that I would be asking them about three separate projects from their professional career. I also added a demographic question of "how long have you been working with information systems" near the beginning of the interview to ensure they were thinking of critical incidents throughout their career.

Trustworthiness

I addressed trustworthiness in a variety of ways, including some of the specific recommendations for credibility and trustworthiness checks presented by Butterfield et al. (2005). First, I acknowledge my bias toward constructivism as an interesting approach to requirements based on my personal experience prior to initiating this research. During the study I looked for indicators both for and against constructivism as one of my sensitizing concepts. Since one of the challenges that emerged from the data related to cultural diversity, I will also acknowledge my point of view is influenced by my own perspective as a white male who spent my first three decades in East Tennessee. The significance that diverse gender and cultural background have on requirements could be an entire research study of its own, but in this study it is one of the challenges identified and included below.

Throughout the interviews, I verified with participants that my understanding of their responses was correct which helped ensure accurate transcription and interpretation during coding and analysis. One method of verification was to ask multiple related questions to triangulate responses and enhance the richness of the collected data.

I addressed authenticity and trustworthiness by presenting balanced, fair, and accurate data, regardless of whether the data support or conflict with my personal biases. The objective of the study is to develop a grounded theory based on the challenges faced by stakeholders, regardless of whether my sensitizing concepts turned out to be relevant.

Butterfield et al. present specific credibility and trustworthiness checks they propose to become standard for any CIT study in order to "determine the soundness of the results" (2005, p. 490) . I addressed each of the checks below.

Independent extraction involves having another person review a sampling of interview recordings or transcriptions to validate the researcher's identification of critical incidents (Butterfield et al., 2005). For this study, I engaged an experienced researcher to review five samples from my interviews and provide feedback regarding my coding of the critical incidents included. The reviewer concurred with my coding logic.

Participant cross-checking is a process of validating with each participant that the data captured during the interview accurately reflect what was intended (Butterfield et al., 2005). Following transcription, I presented participants a printed copy of their transcribed interviews to review for accuracy. None of the participants provided any corrections to my transcribed data.

The point of exhaustiveness or redundancy will be reached when "new categories stop emerging from the data" (Butterfield et al., 2005, p. 487) indicating adequate coverage of the domain being studied. Charmaz offers a slightly different view of when to stop collecting data for constructive grounded theory research. Theoretical saturation "refers to the point at which gathering more data about a theoretical category reveals no new properties nor yields any further theoretical insights about the emerging grounded theory" (Charmaz, 2014, p. 345). All 23 critical incidents collected for this study were coded and included in the analysis. After interviews seven and eight yielded no significant new theoretical insight, no more interviews were conducted as theoretical saturation was reached and there was sufficient data to develop a theory.

Participation rate is a measure of the frequency of occurrence for each of the categories that emerge from coding the data (Butterfield et al., 2005). The categories that emerged in this study occurred in at least three separate critical incidents as reported by participants and found in literature. Butterfield et al. consider a participation rate of at least 25% to indicate validity (2005). The final categories that emerged from my data are consistent with threshold.

Theoretical validity involves comparing the researcher's assumptions as well as the categories formed from the data to relevant literature sources (Butterfield et al., 2005). Sources were found in literature to corroborate challenges that emerged from the collected data. Chapter V includes the literature references that support the categories that emerged from the data collected for this study.

Accuracy of the participants' account during the interview was increased by audio recording the entire session and also by cross-checking the transcribed text document with the participant. By transcribing the recordings myself, I was able to ensure accurate representation of the incidents described by the participants in their own words and non-verbal expressions.

The above checks were done to address the credibility and trustworthiness of the data I collected. While these checks individually do not yield conclusive evidence of credibility, collectively they strengthen the overall trustworthiness of these qualitative data. The next chapter reports the findings from the collected data followed by a separate chapter where the findings are interpreted.

Limitations

While this study accomplished my objective of identifying common challenges encountered related to information systems requirements, it also has limitations and opportunities for additional research. For example, the participants in my study were a combination of known and unknown individuals with professional experience spanning multiple fields including healthcare, insurance, government contracting, and information systems. Due to availability and logistics, the participants were located in the areas of Washington, DC, San Antonio, TX, and Austin, TX. These cities all have a high concentration of information systems work for various government agencies.

Seven of the eight participants interviewed for this study included at least one incident from a project sponsored by state or federal government. Projects unrelated to government were also included, but with interview questions focused on significant disappointment, the participants gravitated toward government-related projects first. Future research on this topic may benefit from focusing on incidents or participants unrelated to government projects in order to compare results.

Another limitation of this research is the overwhelming supply of candidate participants and critical incidents to be studied. Having a specific interest in dealing with rich data and discovering unanticipated results, I used qualitative methods with a manageable number of participants and critical incidents. While this study met the objective, there is clearly more knowledge to be gained in this area. Future research on this topic could apply quantitative methods to analyze data collected from a much larger set of subjects. Since the target population is technology savvy, electronic surveys might be effective.

As mentioned in Chapter I, this study adopted a broad definition for requirements and did not distinguish among different types of requirements. A more granular analysis would likely reveal some types of requirements are less subjective than others, and therefore benefit less from social construction. For example, requirements tied to industry best practices or regulatory compliance may not appear to be subjective. However, given the constant change of regulations and practices as well as the need for interpretation, there is likely more subjectivity than expected. Subsequent research studies could explore which types of requirements benefit most from a constructivist approach.

Summary

In this chapter, I presented the methodology for this study which is based on grounded theory from a constructivist perspective as described by Charmaz (2006). With an approach of interpretivism and constructivism, I presented my conceptual framework as starting with sensitizing concepts which inspired and informed data collection and analysis that led to a theory. After listing the steps of the methodology, I described the data sources and the participants involved, along with the Critical Incident Technique used for data collection. Analysis of the qualitative data involved initial, focused, and eventually emerging theoretical codes. I also addressed measures of trustworthiness and some limitations of the study.

IV. FINDINGS

Introduction

This chapter presents the most significant and interesting findings from the critical incidents collected during interviews. The findings presented were identified through coding and analyzing the transcribed interview data to identify challenges to requirements elicitation as attributed by participants. Direct quotes from participants are the basis for each challenge identified.

Challenges

This section identifies the challenges that emerged from the analyzed data as repeating across multiple critical incidents. Direct quotes from interview participants are provided as vignettes to explain and support each challenge. References from relevant literature sources are also included for additional support. Some of the interview excerpts apply to multiple challenges but are not repeated to improve readability.

The challenges that emerged from the data are addressed individually and also organized into categories. Categorization is open to interpretation, so there are multiple possible configurations that could be used. Based on my perspective and interpretation, I chose to organize the identified challenges into three major categories: change, communication, and knowledge.

Change and communication are not surprising categories for challenges regarding information systems projects. The change category of challenges includes significant focus on stakeholder turnover as well as changes to technology and processes. The communication category of challenges involve obstacles to effective communication among multiple stakeholders. Knowledge, and change to knowledge, is the subject of communication among the stakeholders and becomes the third category.

Challenges emerged from the data that draw attention to the need for consistent knowledge both individually and among stakeholders. The focus here is what each stakeholder, as well as the collective stakeholders, consider to be factual truths that drive requirements. Epistemology is one challenge in the knowledge category while assumptions, conflicting requirements, and confidence are also related to knowledge. The knowledge and communication categories are intertwined because it is often during effective communication that the refinement of knowledge, or learning, occurs. I kept the knowledge category separate due to its significance to my interpretations regarding constructing requirements.

Table 3 is a list of the challenges identified as focused codes and associated categories. The remainder of this chapter presents the data collected that supports each challenge. The order of presentation is simply for readability and does not imply any significance. Appendix H is a diagram showing the relationship among the challenges.

Table 3
List of Challenges Identified

	Category		
	Change	**Communications**	**Knowledge**
Focused Codes	Stakeholder Turnover Technology Process	Project Management and Leadership Respect and Trust Diverse Perspectives Ethics, Contracting, and Accountability	Assumptions Conflicting Requirements Confidence Epistemology

Change

The first category of challenges that emerged from the collected data have in common the element of change. The sections below describe challenges of change in stakeholders, change in technology, and change in process.

Change: Stakeholder Turnover

People come and go from information systems projects for a variety of reasons including promotions, transfers, personal circumstances, priorities, and many others. For this study, stakeholder turnover is defined broadly as any time a stakeholder joins or leaves the project. Turnover is a routine occurrence and cannot be prevented, but it should be acknowledged as a risk to be mitigated.

Analysis of the data revealed personnel turnover as a challenge to understanding and meeting requirements. One participant recalled a program that ended in disappointment after 13 years and over a billion dollars spent:

> The turnover was such that you'd have documents on requirements but nobody [on the current team] was around when those documents were written.... When we were testing the system it's not like you could go and ask someone what did you mean when you wrote this? Those people were long gone by that time.

That same participant went on to say "Another issue of turnover is also turnover of program management. That happened fairly routinely. That more or less changed everything. Previous deals, previous agreements, previous understandings went out the window." The participant described the outcome:

> Probably around $1.3 billion spent on this system that, at the end of the day, the [customer] decided they were better off without it. The day it went operational

> was the day it was declared a legacy system to be replaced by another one. So, essentially a billion dollar failure.

This participant was a stakeholder on the project and attributes turnover to be one of the factors that led to what he calls “a billion dollar failure.”

Another participant reinforced the impact of turnover on projects she had seen over her career:

> You would have individuals that came in with different priorities or different visions of what they think, which was different than the last one. Since the resources were limited, then the priorities would change to meet the priority of the new individual.

Later she summarized that “an individual at the executive level would be in and out of this [project] and they would lose the continuity.”

A third participant recalled a similar challenge with continuity due to turnover of a key team leader who had strongly advocated for introduction of a new technology only to “bail out” once the project began to stumble due in part to the team’s unfamiliarity with the new technology he endorsed. She also expressed concern that the replacement team leader had less seniority and experience:

> They hired a new project manager to come in and this was someone that was brand new to the team. He had never worked for this company before. He was someone new just out of college, had never done project management before, and he didn’t have any business coming in to try and take over a project and he just did not fit in with their team. So that was a struggle. They had brought in some people that were not the same ones that had worked on all their other projects

> before. Some of them were not a good fit. A couple of them got terminated, not quickly enough.

This challenge of turnover, along with change of technology and lack of respect, resulted in this project concluding with disappointment from the customer, a strained relationship between the customer and long-time supplier, and the supplier having to cover hundreds of thousands of dollars of labor costs.

Statements from interview participants such as "priorities would change to meet the priority of the new individual" and that routine turnover of program management "more or less changed everything" reflect the highly subjective nature of information systems requirements. It is worth repeating that requirements do not originate from the target systems, requirements originate from the stakeholders. People bring expectations and perceptions which are the raw materials for many information systems requirements. Authors have equated software requirements to desires. "A desire is a discrete piece of demonstrable functionality that is valuable to a stakeholder or a group of stakeholders" (Cardinal, 2014, p. 32). Characterizing requirements as desires underscores their subjectivity.

One recurring observation from Leviss, as well as from data collected for this study, is the need to identify and involve all stakeholders early and throughout the project (Leviss, 2010, pp. 62, 80, 99). Other authors' concurrence with this perspective is represented by Leffingwell when he says "To be successful, all project stakeholders must actively work with the team to achieve the team's goals" (Leffingwell, 2011, p. 122).

Thinking holistically, adding and removing individuals from the stakeholder team forms a new team and will likely change the collective requirements. Hazzan and

Dubinsky point out that software "requires *huge* and *strong* dependencies between all the project stakeholders" (2008, p. 231). One of the incidents presented by Leviss involved the unexpected turnover of the champion for a project to deploy an electronic health record system. Although responsibility for leading the project was transferred to a trusted and capable colleague, the project floundered with one major factor identified to be a lack of committed leadership (Leviss, 2010, p. 60). In my professional experience, I have rarely seen an explicit validation of requirements following a change of stakeholder. Instead, the typical behavior is to assume the new stakeholder will adopt and support the requirements previously identified by the team as requirements for the target system. One way to mitigate the impact of turnover is to acknowledge and anticipate that a personnel change may result in a change to requirements and warrant re-validation of requirements among the new set of stakeholders.

As one of my sensitizing concepts for this study, it was not surprising to see stakeholder turnover emerge from the data as a common challenge, however the explicit mention of turnover by several participants was a pleasant surprise. I expected the impacts of turnover to be more subtle, but instead it was specifically called out multiple times as a specific challenge that contributed to major failures. The impact of stakeholder turnover significantly influences my interpretation in Chapter V by reinforcing the importance of each individual stakeholder perspective.

Change: Technology

Technology is another source of change that impacts requirements for information systems. As new technology products emerge, previous constraints are removed and new capabilities are introduced. The expectations of stakeholders change as they envision new possibilities.

One participant reflecting across his long career with information systems acknowledged that learning and situational change occurs throughout a project's lifecycle by pointing out:

> The system, or the project, or something like that has to be malleable like the organization so the thing has to change with time. You can't just build something and then just leave it. It has to change over time. So when you build something, you know that you have to build in a concept of easy modification and requirements change."

He later recalled one of his long-duration projects that anticipated and planned for technology change:

> One of the best projects that we've done…We had a good team… We had an excellent technical background, but we also knew what the change in technology was [going to be]. We built into our schedule the change in technology during the project…That was probably the best planned out operation that I had seen in my 30 years.

In this incident, the participant's team anticipated technology change over time and properly planned for the learning that would occur. This incident shows effective project management through applying a constructivist approach and anticipating change.

Another participant also mentioned technology change. When asked about factors that led to requirements not being met she responded:

> They kind of changed over time because everything got dragged out so long. Some of the requirements changed. You know in IT it's ever changing…. The PM would come to the table and [say] "the doctor did say he wanted this, but he's changed his mind because they're not treating the patient that way anymore", or some of the health approaches have changed, as well as some of the technology has changed. And because the project was so lengthy, we were at working groups for better than a couple of years.

This incident mentions challenges associated with both change of technology as well as change of processes. Some of the changes in this incident are attributed to a stakeholder changing his mind which hints at being arbitrary, but the rest of the explanation points to changes to process likely associated with new knowledge. A change of mind may actually be the result of learning and indicate progress toward a more fitting solution.

The fast pace of change in technology presents challenges to requirements for information systems projects. As technology changes, constraints are removed and new possibilities emerge. It is worth pointing out that technology changes are controlled by people and are the result of human endeavors. Technology change is a challenge identified by my participants and is represented in the resulting models. There are often processes that are closely aligned with technology. The next section describes changes in associated processes.

Change: Process

Changes in process also drive changes in requirements for information systems, including people's expectations that technology will reduce costs. One participant described the impact on requirements of consolidation efforts to bring previously separate healthcare organizations and their processes together:

> The need to be more efficient. Costs - the demand to decrease costs and to do more with less. We hear that a thousand times. It's all about the costs. IT is an integral part in everything we do now. It isn't like just the occasional… desktop or the computer where somebody sits down and does part of their work. All their work is done through some type of technology. All those healthcare systems all plug in now, they are all smart devices. So, everything involves IT. When that happens, it causes a change of requirements and the way things have to get done. We've got smarter people, a smarter community, and that community is all looking for technology to meet their demands. Versus years ago they'd sit at their desks with a computer and some new technology came in and we gave it to them they got all excited. It's the other way around now. We're not pushing, they are pulling. And that pull is driving us to expend more and more money to get stuff done so the budget's gone sky high…. It's outrageous what we spend on IT. So naturally it's the first target when you need budget cuts. So we have a national debt the first thing we're going to look at is how do you cut IT? So we've got to be more efficient. That changes requirements, big time. It tells me now that instead of me doing it my way, and [another organization] doing it their way, we have to figure out how to do it the same way together and cost less.

This incident relates to processes that are changing due to organizational mergers as well as "smarter people" having greater expectations about their information systems. While in decades past technology was forced on people, the workforce of today demands technology solutions in the workplace keep up with their personal use of technology.

Wang and Biedermann conducted a study which included identifying barriers to adoption of electronic health record systems by long-term care facilities in Texas. One of the barriers they identified was "the risk of new state or federal requirements" which was reported by 18.2% of their respondents (2012, p. 4). My interpretation is to consider that risk a potential process change which inhibits a clear understanding of requirements.

These three sources of change: people, technology, and processes exert constant influence on the requirements for information systems. When any of these changes occur it should be acknowledged that the requirements may have also changed. This reinforces the time-boxed approach popularized by Agile. It is desirable to complete units of work before expectations change.

Communication

Communication is a broad term that is relevant to many of the challenges that emerged from the data. This section focuses on specific mentions of communication by participants. One participant summarizes with:

> There's projects that I'm not involved with personally that I've seen succeed very well because they've had good requirements and then one's that were either shut down or are floundering because of lack of good requirements. I still think it comes down to communications, talking through requirements, and taking your time to think through what you want and document that. Just like when you build a house. You tell a contractor to "build me a house" you're going to get four walls and six windows. Well, I wanted tile floors. Well, I wanted stainless steel fixtures.

> There are some projects that you see come down the road that the red flags are going off that say OK, stop a minute. Let's talk through this…. I think if you have a story of what the capability needs to provide will help the providers either build it or develop it or whatever they are doing. It's a lot easier to support something if you know what it's supposed to do.

In addition communications, the above also invokes social construction. She went on to explain some red flags that indicate potential problems:

> When [stakeholders] are not clear on what they need. We get a lot of requests that say "I need four servers and SQL database" instead of saying "this is the capability I require, build it for me"…. Vagueness. We get a lot of one-liners. A requirement is not one line. You can't read in between the lines. So when these new requirements come in they're too dang vague and they want you to build costs around it and you can't. So there's the "here we go again" thing.

Another participant pointed out communication challenges on a project that ultimately was delayed three months with the vendor covering the significant costs of the delay. This was a combination of overconfidence, deceit, as well as communication challenges:

> I think that they were not communicating with their owner and he was being kept out of the loop as far as how things were going. They were telling him that things were going very well and they weren't. He was very involved in a lot of other things and so he wasn't really paying attention to how this project was going. This wasn't an enormous project. This was about a 3 million dollar project. So in terms of some people that's not an enormous project. For them it was a pretty good size

project. It's not an enormous company. But for us it was a decent size project. He was not involved and really didn't know what was going on until it was just about too late. I told him quite a bit in the last year, I guess, this was not going right. Things are not going the way they should. Your project manager is not talking to me. He won't return calls. He won't talk to us. This is not working. When you have to kick back a problem ticket ten, twelve, thirteen times on the same ticket, there's something really, really, bad that's going on. There got to be a lot of friction between the people in my office, my staff, and their staff because of things not working. This back and forth.

Another participant attributed issues to general communication of requirements between the customer and provider organizations:

> As often happens, there was a lot of miscommunication and lack of understanding, frustration on the part of parties, [the customer] and [the vendor]. There was an acknowledgement on the part of [the vendor] that the requirements that they now understood to be true requirements could not be met by the solution…and that additional modifications, enhancements would have to be done to the solution by [the vendor's] software engineering product group that's responsible for developing the product and that would take time. So, one issue was proper communication of the requirement after it was determined that the product was not doing what [the customer] thought it should do, so there was an issue around identifying the requirement clearly and then communicating that clearly to [the vendor] and then [the vendor] evaluating that requirement and making a determination as to what work would be required to actually modify the

> solution to implement that requirement and then to get that through whatever testing [the vendor] requires it to go through and get it deployed back to the customer, into the customer environment. It was a series of steps that were required. I think there were hiccups all along the way.

Contracting and project management were challenges also faced on this incident.

Wang et al. (2010) conclude existing approaches are not sufficient to support clear and objective requirements. They propose a framework combining *Collaborative Requirements Elicitation* and *Problem-driven Requirements Elicitation* which are approaches they define with prescribed steps and structure which includes debate and voting to select solutions to problems. I suspect the framework used is less important than approaching the process with the attitude that new knowledge in the form of requirements will be constructed during the collaboration process.

Sajid et al. (2010) point out that customers sometimes do not understand the importance of communicating even though they may place a priority on getting exactly what they ordered. They highlight the dependence between communication and meeting expectations. Lack of a common understanding of expectations leaves to chance whether the expectations will be met.

One of the incidents presented by Leviss described a project to deploy an electronic health record that repeated many of the known bad practices that lead to failure. At the top of the list of bad practices was ignoring the input of users of the system. Commentary by Associate Editor of the incident, Larry Ozeran, summarized:

You have a choice: You can ignore the needs of your users and implement what you think they need, what you want them to have, the cheapest available solution, or any number of wrong answers.

Conversely, you can work with your users. This often costs more up front and takes longer to implement, but it is cheaper in the long run and the only way to obtain an functional system. (Leviss, 2010, p. 80)

Involving users as stakeholders for information systems projects is one of the most frequently voiced issues with communication. Since the user population is normally large and often distributed, it is a significant task to obtain user participation.

Wang and Biedermann also encourage improved collaboration among stakeholders. Their study revealed barriers related to design of electronic health record (EHR) systems for long-term care (LTC) facilities by vendors of software originally designed for other healthcare settings. "Better communications between vendors and LTC facilities is indispensable for more effective EHR system design and development for LTC facilities" (2012, p. 5).

This section highlighted the findings where interview participants specifically mentioned communication as a challenge to requirements. The underlying theme is that while individual stakeholders may have expectations, their inability to effectively share those expectations with other stakeholders, to achieve a common set of expectations, dooms many projects to disappointment or failure.

Communication: Project Management and Leadership

The category of challenges related to communication includes sub-categories which have in common the need for collaboration among multiple stakeholders to achieve a common understanding of the target system. One of these sub-categories is project management. Within Project Management are the elements of time pressures and contracting.

Several participants reported challenges organized under project management because they involve managing scope, schedule, and resources. While timing is critical on most projects, participants pointed out that schedule must be balanced with quality, scope, and budget, therefore challenges related to time pressures are included under project management.

These words from one participant are representative of several others' comments: "I think there was so much pressure to get this out fast that a lot of the upfront leg work was not properly done.… I think it was timing. I think it was just moving too fast. It had unrealistic timelines." Later, while summarizing what could have been done differently to improve the likelihood of success:

> I think if we would have slowed down, looked at what had been provided, and had some internal discussions to identify the gaps. That was just never done….Nobody ever stopped to say it's way too aggressive. We can't do it.

These comments refer to a multi-year project that was continuing to progress at the time of the interview. The challenges had not yet caused complete failure, but were contributing to perceptions of likely failure for the program.

Ineffective project management can lead to a challenge of accountability. One participant recalled an incident where adherence to schedule became an issue:

> The schedule didn't seem to be very put in concrete. In other words it just seemed very fluid. So, if something was due on a certain day, the vendor would get on the

> call, because it was always telephonic for them, they were hardly ever there in person. I think they were out of the DC area, if I remember correctly. But any way they would get on the call and say "we haven't got that, we need another 2 weeks". So, OK, another 2 weeks, you know? And I'm like "don't we have a schedule here? Aren't we supposed to have this done?" I don't think I recall seeing printed schedules.

Reflecting across many information systems projects over a multi-decade career, one participant observed:

> The people who become involved with requirements are always doing this as an add-on to their normal work. So you feel that they are always kind of rushed and hurried and they are not doing this as their main purpose. This is an add-on to their fulltime job.

She later added:

> After you get through the real push with your requirements and you get them documented, you have a tendency to just skim over and gloss over reading especially when it's a 30 or 40 page document. I think that just goes back to the time factor… [Stakeholders on her project] just weren't giving the right amount of time and dedication to it.

This last comment points out that dedication or commitment is sometimes a challenge to achieving expected results. Stakeholders involved with the delivery of the solution who skim over documented requirements are demonstrating a lack of commitment to meeting expectations of other stakeholders.

Another participant describing his view of a successful approach to requirements indicated the importance of commitment. He said "The biggest thing with requirements is…you've got to take the time and planning to really do them." The context of this comment was to reinforce the need for disciplined project management even during times of pressure to accelerate progress.

Drawing on recent highly publicized events, an aggressive schedule is reported to be one of the challenges that contributed to the performance issues faced by the government website for the Patient Protection and Affordable Care Act known as healthcare.gov. Cleland-Huang summarized:

> The sheer scope and challenges involved in the healthcare.gov project should have served as a trigger for more formality in the process. While stressful time-to-market deadlines often lead to corner cutting, smarter development teams will realize that such projects have little time for missteps and that investing time in understanding requirements early in the process will pay dividends over the long run. (2014, p. 29)

This example reinforces the need to invest time and effort up front to achieve a common understanding of requirements before commencing design and build activities.

Incidents documented by Leviss also reported challenges from time pressures. One contributor recalled: "The respondents also made complaints about the short timeframe for the implementation at the pilot site. They believed the decision makers implemented the system too quickly…" (Leviss, 2010, p. 78).

In the study performed by Wang and Biedermann, one of the top barriers identified by 29.2% of their respondents was "insufficient time to select, contract, install software/technology” (2012, p. 14). The above findings reinforces that time pressures applied by management can sometimes have a negative impact on outcomes. While adhering to a reasonable schedule is good project management, rushing through requirements activities may have catastrophic results later.

Besides time pressures, another challenge related to project management is accountability. Regarding a project that had been cancelled by the customer due to the vendor’s inability to meet agreed requirements, one participant recalled:

> I think the program management wasn’t handled appropriately. What I mean by appropriately is it was rare to have full team meetings. The PM normally spot checked. Walked to this contractor, walked to this sub-contractor, walked to the hardware provider, and just kind of said how are you doing? How is this doing? And was assuming to some degree that it was being accomplished. Some of that was trust. Some of it misplaced trust. And unfortunately some of that misplaced trust was also because the small companies that were brought in to solve the software problem were also personal friends of the program manager. So I think they thought this is a personal friend. I’ve known them for quite some time, obviously they are going to tell me if something’s wrong. I think because that sub-contractor was a personal friend they didn’t want to let anyone down and say no, this is just not being solved the way we thought it was going to be solved. That cycle just kept on rotating over and over and because it just took so long people just assumed this was just going to take a long time.

This incident also experienced challenges associated with holding the vendor accountable to deliver what was expected. The personal friendship between the program manager and vendors interfered with proper project management oversight.

A similar incident with a project management challenge was experienced by another participant whose project went over budget and missed the target completion date because the vendor did not meet expectations. This was a vendor who had performed well in the past and so was not managed closely which led to a lack of accountability. When asked why requirements were not met, she said:

> I think the main thing is that having some more stringent rules going in from the very beginning. Because we had had this relationship with them from previous projects and having this trust and these successes from previous projects, we were more relaxed with them and more trusting that this is going to be OK. This time it would be there's no trust. Treat them like a brand new vendor. This rule, this rule, this rule. It's got to be done exactly this way. And just be much more strict, much more stringent in every single step. I think there needs to be a lot closer watch over what they were doing and none of this "oh, it will be OK", not letting things slide. I had too much trust that they were doing what they were supposed to and my staff were doing what they were supposed to because things had always been OK. And so you can't take everybody's word for it.

This incident included challenges of project management, communication, and assumptions.

Participants pointed out that sometimes important factors that should influence requirements are either ignored or never voiced due to political pressure. Intimidation, intentional or not, by stakeholders with perceived power can be a challenge to getting to consensus for requirements. Effective leadership balances power in order to draw out the requirements from all stakeholders necessary to achieve success.

One participant recalled a critical incident on a project that was cancelled after several months of effort with disappointment from the customer and the providers of the solution. Some stakeholders on this project had specific expectations that simply could not be met due to regulatory constraints. The participant pointed out the conflict to the stakeholders, expecting the group to understand the unavoidable failure and to halt the project or adjust the requirements. He recalled “These are not people you inherently want to disappoint. They are more trusted than you are with the boss and the boss’ boss. And so there is political cost to [speaking up].” He later added:

> What I did was inform my [leadership] about where we stood. What I didn’t do was recommend to them what I thought they needed to do in response….I probably needed to think through the next steps for my management versus trusting they would do something about it because ultimately they didn’t do anything about it.

The project continued to waste resources for several more months toward unachievable expectations. Ultimately, the project ended with all parties disappointed and the work scrapped. Such waste could have been avoided had the participant been more empowered to speak up with authority about his valid concern and if his input were respected.

Lack of clear and powerful leadership can also be a project management challenge. One participant observed a lack of leadership on a project to deploy an information system to be shared across five separate organizations. The project was continuing, but behind schedule and with disappointing results so far. She said

> You have program managers at a higher level reaching out to project managers at four locations, but there wasn't a single integrator that had their [finger on the pulse]. There's not one person orchestrating integration of these five organizations to make sure the technology meets the needs.

This led to:

> A lot of finger pointing. One organization has one component and when we did the user acceptance testing there's a lot of finger pointing about whose responsibility is that. That's one example. I think that might be one of the bigger challenges… just a lot of finger pointing. A lot of misunderstanding how our [business] works. That person wasn't necessarily at the table up front so there was a lot of assumptions made there.

The participant considers the lack of project management and powerful leadership a source of challenge for this project.

An incident presented by Leviss described a project to deploy an electronic health record system that was halted due to a variety of problems. One of the problems labeled as "hard to solve" was that "the staff in the surgical clinic thought that they had no influence in the system design and development" (Leviss, 2010, p. 97). Achieving stakeholder involvement is a challenge identified across many of the incidents documented by Leviss. In this incident, key stakeholders in the surgical clinic did not feel empowered to voice their requirements.

Similarly, Sutcliffe and Sawyer (2013) identified suppression of knowledge as a challenge. Their category of *unknown knowns* includes requirements that are known to at least one stakeholder but not articulated for emotional, social, or political reasons. For a variety of reasons, stakeholders sometimes withhold their input about requirements. Some cultural influences inhibit open communication which impacts stakeholder involvement (Leviss, 2010).

My perspective is that strong project leadership should ensure all stakeholders know they are empowered to share their knowledge and influence the outcome. Project management and leadership must find the right balance so as to empower all the stakeholders to contribute. The findings above demonstrate what can go wrong when that project management balance is not achieved.

Communication: Respect and Trust

Participants in my study reported that communication among stakeholders about requirements was impacted by a lack of respect for the views of others. Mutual trust and respect were challenges on some projects where there was conflict among stakeholders' expectations.

One participant described a project to enable clinicians in a hospital to streamline access to computerized records. While everyone agreed on the general objective to make access as easy as possible, the provider of the solution was obligated to follow applicable regulations regarding user authentication and logging access to protected health information. When my participant reminded a stakeholder physician about this requirement, the physician laughed at the suggestion of having to endure such an administrative hassle and insisted there must be a workaround. My participant insisted the regulation was very clear, but the physician was overconfident that he could get the requirement waived due to his trusted connections in positions of power. The conflict remained unresolved and development of the project continued for several more weeks until the system was delivered, including the required authentication and logging elements which did not meet the customer's expectations. In the end, the solution was discarded and the entire effort a waste of time and resources. One significant factor in this failure was the physician's lack of respect for the expertise of the solution provider. Had the intractable expectation been addressed when identified, at least the wasted effort would not have occurred. In hindsight, the participant observed "there was an inherent lack of trust between them and us that pre-dated that particular instance. They had been disappointed before." He continued with "we also had something of a negative perception that they were a difficult customer." This two-way lack of respect and trust resulted in the commencement of the project without a clear path to success and ended with failure.

Another participant recalled working with a vendor company who had performed well on previous software modernization projects. Given the previous success, my participant, the customer, placed trust in her supplier that she later regretted:

> Part of the problem was the tools they chose to use for part of the modernization were some software tools they were unfamiliar with. They assured us they would be able to use these tools. They were not as familiar with these as they should have been. Because we had been using [the vendor] and had a working relationship with them for so many years, we trusted that they could do what they said. As they got into it a lot of the things that we needed the system to do, we found out these tools couldn't do. Oh, this product can't do that. It won't allow us

> to do these things. So that was a problem. They started running into issues with that more and more and more and it's like "no, we can't do that". Well, this is supposed to modernize the system. It has to do at least as well as our current system and then better. It can't go backwards. We can't regress. We have to do at least as good as we have now. So, to tell us that we can do less with a new modernized system is not acceptable. So that was a big part of the problem.

The customer acknowledged placing too much trust in the vendor was a mistake she would not repeat again. She claimed to have adopted a "trust but verify" approach for future projects. She became frustrated with the vendor respecting the views of his own employees versus listening to his customer:

> I think that if their owner would have listened to us and our complaints, our objections, that would have made a difference. But he didn't take them seriously. We called him. We had him come over for meetings. We pointed out things. We said "look at this, look at this, listen". And it was like "Ok, Ok". But then nothing was done about it. His real interest was somewhere else. It was like "Ok, Ok" and it was like he was just pacifying us by listening to us but he wasn't taking us seriously because I think he was still listening to them. "Oh no, there's really no problem". I understand listening to your employees and wanting to take your employees word for things but you've also got to listen to your client and trust that they know what they are talking about. And so I think if we had been taken more seriously I think that could have made a difference too.

Beyond lack of respect, this incident also includes overconfidence.

One participant reflected across his long career in information systems and provided this summary observation:

> It's hard to get people to sit down and say how can we make this work for everybody? You give me some, I'll give you some. You have to build that trust that everyone knows that we're not out to get you, we're here to help you. It's not the technical issues that cause failures.

The above examples of challenges related to trust and respect highlight their impact on effective communication about requirements. It takes engagement over time for a person to earn trust and respect from another. Collaboration among stakeholders helps enable the trust and respect necessary for a common understanding of requirements to be achieved.

Communication: Diverse Perspectives

One challenge I did not anticipate prior to conducting the study is the significance of diverse perspectives due to cultural and other differences. One example came from a participant involved with information systems for automobile insurance claims. The team responsible for developing software from written requirements included individuals outside the United States. These information systems professionals were familiar with the tools required to build software solutions, but unfamiliar with terms and concepts of the problem domain. The requirements were written for an audience assumed to be familiar with insurance terms and concepts. The participant observed:

> It's a cultural problem. An American who knows and understands this culture and lives in it and understands what happens when you have an accident in a vehicle and somebody has suffered an injury versus somebody in [another country] who –

> they don’t even have insurance. They don’t even have auto insurance. There is no such thing as general damages compensation and so they just don’t know the problem domain and they read the requirements without that background – without that basic thing we take for granted – and they just turn it into something else”.

This challenge led to communication issues among the international software development team which resulted in re-work and delayed completion.

Within the United States there are cultural differences. Authority in the military is heavily associated with rank. The military culture is one where lower ranking people, regardless of level of expertise, are less likely to correct or challenge higher ranking leaders. This cultural factor leads to issues discussed above in the section on power. One participant involved with a military project recalled:

> Here are people…representing an issue driven by, in this case the…[ranking officer], who are trusted advisors to the CIO. These are not people you inherently want to disappoint. They are more trusted than you are with the boss and the boss’ boss. And so, there is a political cost to [speaking up]. At the end of the day you pay the political cost either way. It becomes a creditability issue where you’ve got someone who doesn’t understand any of it anyway and it’s not probably an open and fair hearing into the matter.

Another example from a participant points out cultural differences and lack of respect between geographic regions of the United States.

> So this customer is in the South and everybody speaks with a southern twang. The software provider [is] far metropolitan up North. Always suits, always ties, almost

> a bit of arrogance. That cultural difference of “you think you are better than me” or “I perceive that you think you are better than me” and, on the flip side, “you guys are kind of just rednecks down here” and “we know a lot more than what you think”. That was one culture. It was kind of taking the city boy into the country or the country boy into the city and not really having that level of respect with each other that they should. There was nothing ever egregious that I saw on the surface that would make the customer say “you’re rude” or really arrogant, but seeing both sides of it we were able to tell the city slickers came to town thinking they could do what they could do and the other side of “these guys just don’t know what they are doing”.

This lack of respect between groups resulted in communication problems and distractions from addressing technical requirements. Months of work were wasted and eventually scrapped when deadlines were missed and the customer lost faith in the vendor. This project did not have difficult technical requirements, but suffered from lack of communication fueled by cultural differences and lack of respect.

Another example of communications being impacted by organizational culture differences came from one of Leviss’ incidents where a medical college and associated non-profit hospital were acquired by a university and a for-profit hospital chain. “Merging cultures of the medical college and university was proving a challenge. A culture of mistrust and anxiety about cross-college collaborations with the acquiring university seemed to prevail at the medical college” (Leviss, 2010, p. 53). These cultural differences led to a key stakeholder being excluded from participating in requirements discussions for deployment of a new electronic health record. A medical informaticist, with relevant experience, was hired by the university and offered assistance to the medical college to avoid problems she had seen before and anticipated for this project. The assistance was declined. “Because of existing sociocultural issues that divided the acquired medical campus and the acquiring university… the informaticist’s services or advice were not utilized” (Leviss, 2010, p. 54). The outcome of the project was a lawsuit filed by the medical college against the vendor for not meeting contractual requirements, as the informaticist had anticipated.

Gender differences became a challenge experienced by one participant. She encountered one project manager who “did not like working with women. He had a real issue dealing with women. It was evident to everybody.” Communication about expectations and system requirements suffered on this project due to gender differences. The project did complete but was significantly delayed with a financial penalty absorbed by the vendor.

One interesting, and unanticipated, challenge was mentioned in one of the incidents analyzed by Leviss. The project appropriately involved stakeholders from across all functions of a hospital who provided requirements for the system including user interface specifications such as screen layout, fonts, colors, etc. After the initial deployment of the system to a small group of targeted users, a larger population of users began to use the system. The larger group included users who were older than the people involved with specifying the user interface and complained of having difficulty reading the small fonts selected (Leviss, 2010). While subtle and probably easily corrected, this incident involving degrading vision of older users points out the many reasons why diverse perspectives have a strong influence on perceptions of success for information systems. Individual perspectives are influenced by race, gender, age, health, nationality, education, and many other factors that establish individual context. These factors influence priorities. In other words, replacing one technical subject matter expert with another may change requirements.

Literature sources support this finding that diverse perspectives present challenges to communications. Leviss wrote “a culture that inhibits direct and open communication leads to diminished stakeholder input in favor of conflict avoidance behaviors that often defeat the intent of HIT systems” (2010, p. 18). Sajid et al. (2010) also identified cultural and perspective differences as some of the most common sources of ambiguities and communication barriers that must be resolved.

The findings in this section highlight the communications challenges that may arise from cultural differences among stakeholders. All stakeholders should be sensitive to these potential obstacles to clear communication and understanding.

Communication: Ethics, Contracts, and Accountability

The communications category includes incidents of challenges to ethics, contracting and accountability. This section presents incidents where the interview participant questioned the intentions or motivations of others. In all these cases, the ethical concern raised is that a vendor is intentionally misleading a customer in order to increase profit. In several of the cases, accountability and contracting are challenges because payment is issued to vendors even though the customer’s desired outcome is not achieved.

One participant recalled an incident where he questioned the ethics and motivation of the company contracted to deliver an information system solution:

> There's an incentive to win the contract but not necessarily an incentive to be fully [*did not finish sentence*]. There's a profit incentive. To bid something that will win and then potentially go back and change it in order to make it work is not at all uncommon in this line of work. In those change requests there is potential for immense profit because at this point you have the contract. I don't doubt that there was some of that going on when you are looking at an acquisition that's in the billion dollar range that you look for ways to do whatever you can to get the work and then figure out how to make it work afterwards. I think there are people who are aware that this wouldn't work, but ultimately the structure of the contract came to be a cost-plus contract so the vendor did not have an incentive necessarily to deliver a fully capable solution when they were going to get paid to change it anyway. The structure of the relationship was not one that ultimately reinforced [the customer] getting what it needed. At the end of the day the program got killed, the [customer] didn't get what it needed, but the contractor was fully employed for fifteen years at [tens of millions of dollars] a year.

To make matters worse, that company continues to provide service to the same customer on other projects. The participant continued:

> In that line of work it's very easy to, even when you have high-profile failures, to still come out clean on the other end and be able to bid on work in the same field. In fact, that company won another large-scale … system right on the heels of this thing failing. And all the people who failed on this one went to work on the other one.

This example involves challenges of ethics as well as contracting practices and accountability.

Two other participants had separate yet similar experiences. One recalled a project where her company "got kicked out" by the customer for not meeting contractual requirements. The participant worked for a large company responsible for integrating systems provided by other sub-contracted vendor companies. The overall solution to be delivered by her company was dependent on effective contributions from all involved vendors. My participant observed one of the sub-contracted vendors was "there to bill, not to accomplish" which became the weak link in the chain. This project was cancelled due to non-performance, yet the bills from vendors for their time and materials were still paid by the customer. Once again, accountability and contracting challenges led to the customer paying for services and yet not achieving their desired outcome.

Similarly, another participant experienced a project which ended as "a complete loss" and embarrassment for both the customer and the contracted vendor. The participant recalled:

> They were not able to solve that challenge in the timeframe that was allocated and the communication back to the customer was always "everything was on time and on target." And it wasn't until three quarters down the road that the integrator had to come back to the customer and say "you know what, we are having a hard time solving this problem. We spent a million, two million dollars on all this shiny hardware but not enough on the developers."

The misleading "on time and on target" status reports demonstrate at least the challenge of overconfidence, and possibly the challenge of ethics, depending on the vendors intentions which were not known. Despite the cancellation of the project due to disappointing progress, the vendor did collect payment from the customer for the wasted labor effort but not for hardware that was consumed by the project. This recurring practice of paying vendors for failures contributes to the challenges of accountability and contracting.

Contracting and ethics challenges are intertwined. One participant recalled an incident when the customer project manager was inexperienced and did not control the contracted vendor who exploited the situation to increase profits. She said:

> The project manager on the [customer] side that was assigned to it did not have what I feel is sufficient PM experience to control the project. So I sat on there as kind of a consultant for the infrastructure side, for general IT purposes. There were budget people on the working group…. But the contractor, with each meeting, basically said "Well, we're going to add this", and "we're going to buy this hardware because we think it needs load balancing", and "we're going to buy this". So they kept throwing additional requirements, the scope kept increasing. A lot of it made sense, however, the project manager did not control it and just agreed with everything the contractor said. So the budget just went way off schedule. The deployment schedule went way off to the right. It just…everything got lost in there. So by the time the customer started getting something they could use it didn't meet the requirements. They didn't concentrate on what the customer needed, they concentrated on being able to expend additional funds and drag the project out.

Not surprisingly, this vendor-led project ended without success. The participant summarized:

> By the time we got anything to the actual user community they were very dissatisfied with it. There were some pieces they said were OK, but overall dissatisfied. They said it was too slow. It didn't do this, didn't do that.

This incident shows challenges with project management, ethics, and contracting.

Proper attention to contract details is an important part of project management. Negotiating the terms of the written contract, and later enforcing those terms, is a form of communication about requirements. One participant from a large corporation observed:

> We would often joke that we didn't need to spend a whole lot of time writing these proposals because the customer would never read them, that they just trusted us to do the right thing. So that's really not any way to contract for a very complex…system.

The participant was frustrated by not receiving more detailed requirements from customers in contracts resulting from proposals. Having earned the customer's trust by delivering on previous projects led to a lack of collaboration with the customer on subsequent work.

These incidents demonstrate the intertwined challenges of ethics, contracting, and accountability. Individually and together these challenges may raise obstacles that inhibit a common understanding, or at least prioritization, of requirements. Stakeholders should be aware of these challenges and be proactive to address them when encountered.

In addition to communications challenges, another source of challenge for information requirements is getting consensus among stakeholders about what is considered factual knowledge. The next section describes challenges related to knowledge.

Knowledge

The final category of challenges is knowledge. Based on my interpretation, this category includes challenges associated with commitment, assumptions, conflicting requirements, confidence, and epistemology.

Knowledge: Commitment

Lack of commitment is sometimes a challenge to thorough requirements. There are a variety of reasons why stakeholders may not be fully committed. Distraction due to other priorities and limited time is one example. One participant recalled a project she considered a “huge deal” because it involved identifying millions of dollars of fraudulent payments.

> It all went back to the requirements. It all went back to just trying to get the time from the resources to do it. It felt like there wasn’t a commitment, even though there was a signed contract and this was something that this department said they wanted to do. The will to do it, there was something not there.

She later added

> The people who become involved with requirements are always doing this as an add-on to their normal work. So you feel that they are always kind of rushed and hurried and they are not doing this as their main purpose. This is an add-on to their fulltime job… You’ve just got people who are maybe 10 – 20% of their work is that implementation and I just don’t think they give themselves the time to really think it through.

When later asked whether the right people were involved in the requirements process she replied “I would say yes. They just weren’t given the right amount of time and dedication to it.” The challenge of commitment relates to both the individual as well as project management as there are priority conflicts that occur both individually and organizationally that are bound by fixed available time.

Another incident presented by a participant involved a requirements workshop where a senior subject matter expert met with a visiting project manager. The participant recalled:

> It was probably too preliminary for the requirements to get very detailed. When we needed the details that was where it was more difficult. It wasn’t like a really true, good, hashing out session. It was kind of more of an overview session so we could get started. And then he was just not ever really made available to us again.

This example demonstrates a lack of commitment, either by the individual or the company leadership, to support the effort of identifying requirements.

Another participant experienced a project that was shut down “after millions of dollars” of investment over two years due to the customers changing their minds. She summarized:

> I would say that it was the [customer] didn’t really know. I think they were kind of making [the requirements] up as they went. So again instead of having one document listing everything it has to do, it was one of these: well, we want this. OK now do this. Well, that’s not really what I wanted on that, I want to see it like this. So we could never meet their expectations. I think at some point they just said we’ve sunk enough money into this and we’re not doing it.

The lack of commitment to think before acting contributed to this project's failure. When asked what could have been done differently to improve likelihood of success, she said:

> In hindsight, I think we probably should have sat down after the first time they came and said "that's not really what I want", I think we should have said "OK, we've got to stop. Let's sit down and talk through this." Get a signed document that says this is what we're working toward and get that agreement. That just never happened.

This incident also includes the challenges with contracting and project management.

One participant's project achieved most of the intended scope but was significantly delayed due to lack of commitment and communication. He said about the requirements:

> It was kept loose because they couldn't define it in their mind, appropriately. They wanted to play it, a little bit, by ear. As they moved farther down the road they could sit there and look at "Well, how much more expensive is it to add option C?" Since nobody wanted to actually commit to the exact thing this left it open for them to go "Well, I can afford option A and B, that's great it gets me 2/3 down the road. Maybe next year I'll afford option C. Unfortunately they couldn't even get through option A. And because of that, and neither side committing, since the [customer] would not commit to an exact thing, the software provider would not commit to an exact thing, neither side really committed to anything really precisely and that let the expectations down for the customer, but it was to allow some flexibility but the flexibility didn't meet the needs, really, of the customer.

Challenges with contracting and project management contributed to this incident suffering from a lack of commitment by the customer and the vendor.

Another participant recalled a critical incident on a project that was underway and struggling at the time of the interview. She observed:

> I think they're having trouble identifying the actual functional people who will provide the requirements. Even within a [medical facility], who do you go to to set the requirements? Do you go to the CIO shop? Probably not, they are just the techies. Do you go to the nurses? Doctor's won't like that. Go to the doctors, the nurses won't like that. You go to pharmacy, you're not including lab people. So it's difficult. You would have to bring someone from each of those major functions of a hospital to the table to determine what they want as far as a collaborative solution.

When asked why all the stakeholders were not being brought together she replied "I think it's just too much. Everyone is looking at it as too hard to do right now." The project was continuing but without clear requirements or commitment from leadership.

From literature, McKinsey & Co. determined lack of a committed leader was a factor in the healthcare.gov launch (2013). Sajid et al. identify lack of willingness to commit sufficient time as a challenge to eliciting requirements (2010). These examples from other studies support the findings above that lack of commitment can be a challenge to common vision among stakeholders.

This section highlighted challenges associated with commitment which led to shortcomings of requirements. Without commitment, efforts to construct requirements may be superficial and perfunctory leading to weak alignment of thought among stakeholders.

Knowledge: Assumptions

Assumptions were identified by several participants as sources of challenge for requirements. Assumptions become problematic when they are not promptly resolved to be factual or not.

One incident involved different divisions from the same large vendor organization working on a project for a customer. Ineffective communication challenged the project and the stakeholders avoided opportunities to collaborate about requirements. When asked what factors led to the problems, the participant responded:

> I think assumptions. The assumption that they understood each other when they really didn't. I think the other thing that played in was not having constructive, if not daily, weekly touch points – feedback and really, up front, setting the customers' expectations. "I am hearing this, let me repeat it back to you. Your expectation is to see A, B, and C come from this product. Am I correct or am I off?" That was not confirmed to make sure each side was understanding each other. And since the customer said "well my service provider is from the same company, they must really know how I operate here. That's why I'm picking you." In reality it was "No, you're dealing with a whole different division that has never ever seen you before and has no idea how you operate. That was detrimental. And to some degree the service provider not following up on either side was also detrimental because that group did know exactly what they were saying, how they said it, and what they meant. But again they did not follow up.

> They just expected “They’re from my company. Of course, they are the best guys out there that does this type of work. I don’t have to babysit them.

In this example, the customer made assumptions about the effectiveness of communication between different divisions of the vendor company. Also the vendor company representatives made assumptions about the capabilities of their colleagues in another division. Along with the challenge of assumptions are challenges with project management and communication.

Another participant recalled a major project where an overzealous contractor moved forward with sparse requirements and filled the gaps with assumptions. She said:

> It was about a million dollar project. It went on for – it’s been a long time ago. I tried to put it out of my mind. That was when I would come home crying almost every night. They would deliver code down to us and I would look at it and just cry. It was so bad. It went on for months. It was horrible. It was just atrocious. They didn’t ask any questions. They just made all of these decisions. It was terrible.

The emotion in her voice revealed the magnitude of her frustration about the vendor not asking questions to clarify requirements before writing software code to implement their assumptions.

Another participant shared a similar experience where lack of clear requirements allowed for assumptions to be made:

> The first meeting was in person. More of a formality than anything else, kind of a kickoff meeting. Introductions and roles were kind of laid out in this meeting and then everyone kind of went to their own corners thinking they knew what they

> needed to know to proceed. There really wasn't another interaction for almost 3 weeks, roughly. Might have been 3 to 4 weeks. At that point, the customer is expecting something big. You've been gone 3 to 4 weeks I should be seeing something pretty spectacular at this point. Came back with something that was not even hitting the 50% mark on what they were looking for. And not identifying that they were just misinterpreting each other and requirements and definitions. They said "Oh, we're sorry. We're going to go back to our own corners again and come back with something that is really the thing you are looking for.

The challenge of making assumptions is related to the challenges of confidence among stakeholders and also lack of respect for the customer's input.

One of the incidents presented by Leviss described a project where one resident physician dedicated a year of research to developing an information system to manage patient data. The physician built the system based on his own understanding of needs without involving the people he expected to use the system. False assumptions regarding the needs and desires of the target users was the major issue that plagued the resulting system which was never used (Leviss, 2010). Despite having a year to validate his assumptions as legitimate requirements, the passionate champion chose to proceed with the assumptions and the project was a failure.

This section highlighted challenges related to assumptions. While assumptions are normal, especially in the early phases of an information system project, they must be resolved at some point into confirmed knowledge. Left unconfirmed, assumptions propagate uncertainty among the stakeholders which may lead to issues with commitment and common understanding.

Knowledge: Conflicting Requirements

Involving multiple stakeholders is both essential and problematic. As the number of people involved with a project increases, so do the risk and volume of conflicting expectations. While involving all the stakeholders while eliciting requirements for an information system has been shown to improve results, it also exacerbates the challenge of resolving conflicting expectations. The following incidents demonstrate this challenge.

One participant recalled a specific incident where the stakeholders involved with requirements for a system chose to avoid rather than resolve conflicting expectations:

> It was a difficult issue. We were really at an impasse. Instead of dealing with that issue then, fully and completely, to adequate resolution, we kind of tabled the issue. And in tabling the issue it didn't go away and when it came time to deliver it was still there and hadn't been resolved. I see that very consistently in the requirements processes that people tend to push off things that are difficult or not politically expedient and those things never go away and they ultimately reveal themselves at a very inopportune time after a lot of time and a lot of money has been spent.

This was a large project with a specific and well known conflict in requirements with no clear resolution. Despite the obvious impasse, his team "moved on knowing they were not satisfied with our answer but not having any other answer." This project continued for several months and was eventually cancelled with both customer and provider disappointed by the wasted effort.

Another participant recalled a project where the project manager thought he knew all the requirements for a large population of users and worked alone with an aggressive vendor to specify a system. Unfortunately the project manager avoided including other stakeholders in the requirements effort, which also relates to the challenge of overconfidence and trust. The project ran for over three years at a cost in the millions of dollars before being cancelled due to lack of progress. The participant recalled:

> We invested so much in it, when I think of all the personal time. I would really have like to have seen that success and I really think we should have convinced some of the [other stakeholders] that were going to use it to participate with us from the get go. I know they don't like doing that because their time is to [get work done]. But I think if we had convinced them that it was in their best interest to sit down with us all along and get it done the way they wanted it, I think we could have done it.

In this incident, avoiding the complexity of involving all stakeholders and dealing with their possibly conflicting requirements led to failure.

The well documented problems associated with the launch of the healthcare.gov web site for the Affordable Care Act are another example of a major project that reportedly suffered from the challenge of difficult questions not sufficiently addressed. Cleland-Huang observed:

> For whatever reason, people seem to assume that everyone implicitly understands and agrees on the required system qualities. However, questions such as "how secure?," "how fast?," or "how reliable?" often remain unanswered. The Healthcare.gov project serves as a chilling reminder of what happens when such questions are not adequately addressed or not fully implemented in the delivered

system. While specifying quality concerns in no way guarantees that the system will satisfy them, it is clearly an important first step. (2014, p. 28)

The highly visible healthcare.gov failures reinforce that major information systems projects suffer from challenges associated with communication about requirements.

This section highlighted challenges related to resolving conflicting requirements. As was the case with assumptions, conflicting requirements are normal and must be resolved among stakeholders to determine consensus requirements for an information systems. Compromise may be required to remove the conflict and agree upon an actionable requirement. If conflicting requirements cannot be resolved, the stakeholders must consider whether the project can be perceived as successful. Sometimes, a change of stakeholders is the only resolution to conflicting requirements.

Knowledge: Confidence

Stakeholders involved with requirements must have an appropriate level of confidence in their own ability to perform, but not so much as to interfere with the performance of others on which they depend.

On one project, a participant experienced individuals who were responsible for developing an information system, yet they “won’t ask questions” which the participant attributed to a lack of self-confidence. When pressed for more explanation, she continued “I think [asking questions] makes them feel like they are inadequate, that they should just get it.”

Another participant experienced a stakeholder who stated “I’ve got all the requirements for the customer and we don’t need to involve them anymore.” This attitude of overconfidence eventually led to cancellation of the project after three years of work and millions of dollars spent. Repeated attempts by a consultant to advise the stakeholder to engage others was rebuffed. She explained:

> He was in an awkward role. He used to be the [leader] for that…group. He came in kind of as the functional [leader] when the project started. He was that customer, but he retired…and he came back as the PM. So he thought he had down what the customer wanted, unfortunately he was not in that role anymore. So as the first year goes by things change and he didn't catch the change to it. You know what I'm saying? So now he's in this PM role and has no experience as a PM and is losing sight of where the… program had gone. So he didn't have a customer perspective anymore.

Based on his earlier experience, this customer believed he understood the requirements more thoroughly than he actually did. This overconfidence contributed to the project's failure. This project ended after three years and millions of dollars spent with disappointment from all involved. Funding was cut and the customer was left to find a different solution.

Overconfidence and arrogance was experienced by another participant who was on the customer side of a project. "[The vendor representatives] talked to my boss who was an arrogant SOB and they did not talk to anybody else. They just came to the conclusions that he knew what he was talking about." She concluded with "there was not one thing about the entire thing that was done right." For example, the vendor and boss chose to replace older printers with new laser printers. Unfortunately, they failed to consider that the "reports were thousands and thousands of pages long" and printed on continuous feed paper so it would conveniently fold and stack. The new laser printer technology heated the paper to the point where it would not fold correctly and so printer "was spitting all of this paper out and it was just all over the floor". She recalled it was "like being in an I Love Lucy show." When she asked if the vendor had calculated the cost of printing supplies for the new system she said the vendor responded with "Well, no, I didn't think about that." The project suffered from choices that resulted in an unsustainable increase in printing costs and a very disappointed customer.

Another incident involving overconfidence resulted in a project disappointing customers. The participant recalled:

> Everything was achievable, but because they thought that was a very achievable piece of software to develop, the effort was not put into it to make that piece successful. Something that was really 30% of the project became 100% of the failure.

He went on to say:

> There was a secondary piece where the small companies that were tasked with this, I think, were overconfident. I think they looked at it and said this is really not that difficult, this is not that big of a problem and couldn't come to grips that they had a problem that was maybe beyond their capability.

The project ended badly as "the customer lost faith in this contractor and shut off the project deeming it unsolvable for lack of delivery of the original requirements."

Another participant identified a challenge resulting from a stakeholder allowing his own pride to interfere with clarifying requirements. The project involved a company developing custom software for a new customer with whom they had no prior experience. "After a few meetings we were able to recognize that both the customer and the software provider were basically talking past each other." They both used technical jargon, but with different meanings for the terms. The participant reported the software vendor wanted to avoid the embarrassment of admitting "we just don't understand their business as well as we thought we did." While communication is an identified challenge, this separate challenge of pride led to the appearance of understanding when in fact there was a lack of understanding. The result was unfortunate delays and re-work, wasting time and funds, and eroding trust between the parties involved. Ultimately, the customer lost faith in the contractor and cancelled the project after spending millions of dollars, which was a major embarrassment for both the customer and the contractor.

Overconfidence leads to questions not being asked. A participant shared this observation:

> Developers are pretty smart people, for the most part. Some of them can be very intelligent. When they do a great job, they do a great job. But they also have a tendency to believe that they are always right and that "why wouldn't the user do it this way?" Well, they don't know the business knowledge. They don't know the business domain. They are not a [subject matter expert]. But they just move forward and think this is the right way, "Hey, I came up with it". So it's kind of a mix of … the developer not going back and questioning some of the requirements or construing them in such a way as… What ended up happening was it was a different interpretation of the requirements than what should have been. Just didn't ask because he didn't feel like he needed to ask because he knows best.

This project ended when the contracted company was terminated by the customer. The customer's perspective was the vendor "didn't do anything for us but billed us, so goodbye." Project management and communication were other challenges on this incident.

This section highlighted that effective stakeholders have an appropriate balance of confidence. Stakeholders must have sufficient confidence to convey their positions, but not overly confident so as to not respect the positions of others stakeholders. Stakeholders who lack confidence may suppress sharing their knowledge and unwittingly lower the likelihood of perceived success for the system. Overconfidence may suppress knowledge from other stakeholders leading to the same result.

Knowledge: Epistemology

Epistemology is the study of knowledge and how people acquire knowledge. Since I consider requirements as a form of knowledge acquired by stakeholders, it is important to consider how requirements are established. As one of my sensitizing concepts for this research, I watched for indicators of positivist and constructivist behaviors. One participant described challenges experienced during a project that was struggling to meet expectations at the time of the interview:

> That project is working on establishing a centralized collaboration suite, if you want to call it that. A place where we can share information, everybody share documents, without mailing things and large files. I have seen attempts to do that several times. It's almost an impossible task because everybody has, at least for us, I think, has a different approach on the way they collaborate and share information.

She continued later with:

> One [stakeholder] says “This is the way I do business. I’m not going to do the same as that other [stakeholder] because I have a different customer base. I’m in a rural area, he’s in the city. We just do things different here.” So I think you’re going to have that kind of requirements clash on getting things established, like workflow.

And then:

> We’ve tried to sit down with representatives from different communities and gather some of their requirements, see what they said and try to figure out which ones are the same, which ones are really different. Can we do a bunch of different ones? Well, yeah we can. The vendor said “I can do anything”, but there goes the budget. You know, those kinds of things. I’m not sure there is a way. That’s why I think this would be a very difficult sell. Because I think it would be really hard to come to a standard set of requirements that everybody would be happy with.

The approach described attempts to “gather” requirements from the various sources. She acknowledges there is conflict among the stakeholder perspectives, but does not reach a point of reconciling differences in order to construct actionable requirements.

Another incident described a successful information system project with these actions:

> We met with all the [stakeholders]. We told them this was their opportunity. We actually had video conferences. Sat with them. Talked with them. Had a lot of phone calls. Just a lot of personal time with a lot of the sites. Didn’t necessarily have them coming to a lot of [large group meetings] because then there’s too many and you get very little accomplished. So we left the working group to do

> working group stuff and took the conferences and requirements work as a separate structure. I think that was the biggest part of the planning. Making sure everybody understood. Then once we had an idea of what we wanted to do we did briefings and we briefed the heck out of it. We briefed it over and over. Some audiences must have heard it twenty times, the ones that come to everything. But we caught new people every time. We briefed it at the technician level. We briefed it at user level community. We briefed it to the [executive] levels. Briefed the heck out of it. Everybody knew that we were doing this consolidation effort. They just knew. So it wasn't a surprise.

This successful approach was much more constructive than the earlier approach. Success required commitment as well as sufficient time and funding.

Another participant described some elements of partial success within an overall failed project with:

> They got very invested in working with people who actually did the work to really understand what they were doing and understand their business process and that didn't come from reading the requirements document, that came from time with the customer, sitting with them and going through their work and capturing that. Ultimately that didn't translate very well into paper… We made dramatic improvements, ultimately, to the effectiveness of the system for the [customer] because he understood what it was supposed to do. His job actually was a tester, he was just trying to construct good test cases, but in doing that and in capturing that data he shared it with the developer who then built to the test case which represented the actual real world case and we actually started to get a system that

> worked the way it was supposed to work. But that was very dependent on him. If and when he left, (he ultimately ended up going to work for the vendor, so it wasn't a loss to the program in that sense) there was no way to translate that knowledge into something that was easily shared. It was born of the time, and the drive of an individual, to really understand that so he could do his job better. When I've seen programs or projects work well, there's always someone like that. There's always someone who essentially is the authority on what the real requirement is because they really understand what it's got to do and they can make the decisions and they can steer the program. If that person's not there you end up with a bunch of people who aren't really sure and hence either make the wrong decision or make, more often I think, make no decision and just kind of muddle on with what they have because at least it's something.

Another participant succinctly stated a limitation about relying on written communication about requirements:

> You don't want to assume that whoever's reading this is going to be able to read your mind. You have to make it really, really clear as far as what you need and what you want, what you expect this to do for you. I think people don't always do that. You have to spell it out.

This final incident reinforces the challenge to achieving a common understanding among stakeholders. Written specifications have the limitation of asynchronous feedback and delayed validation of understanding. It may not become clear that written requirements are not being correctly understood until too late. Interactive collaboration, in real-time, provides immediate access to validate understanding and make corrections as needed.

This section highlighted challenges related to epistemology, or how the knowledge about requirements is established among stakeholders. Traditionally, a positivist approach has been used to gather requirements. The incidents above point out challenges with the traditional approach and the perceived improvement in results from a constructive, collaborative approach. More detail about this topic is covered in the next chapter.

Summary

This chapter presented the findings from the interview data collected and analyzed for this research study. Through analysis, the findings were organized into categories of related challenges that impacted perceptions of success for information systems projects. The categories of change, communications, and knowledge were covered. In the next chapter, I will present my interpretation of these data as a theory along with supporting observations from my own experiences.

V. INTERPRETATION

Introduction

My interpretation of the above findings leads to the theory that information systems requirements elicitation commonly faces challenges of change, communications, and knowledge which may be mitigated by social constructionism and constructivism approaches. These challenges hinder achieving common understanding and expectations among stakeholders and therefore jeopardizes perceived success. This study revealed specific challenges encountered by practitioners involved with information systems requirements and my analysis of those challenges yielded the categories of change, communications, and knowledge along with the theory.

In contrast to natural sciences and engineering disciplines that are enabled and constrained by physical laws of nature, information systems are often limited by human-imposed constraints. While there are elements of adherence to scientific and engineering truths that influence judgments of accuracy about information systems, many perceived failures are attributed to factors that are more social and subjective.

The theory leverages the perceived effectiveness of Agile methods, as documented in literature, combined with the challenges identified by my interviews to highlight the collaborative learning about requirements that must occur on information systems projects. This necessary learning will be accelerated by approaching information systems requirements with an attitude that embraces constructivism and social constructionism.

This chapter combines my interpretation of the above findings with additional relevant literature and my own observations. The result is a theory and recommendations for a subtle, yet profound, change of attitude about requirements.

Constructing Requirements

Contrary to common expression, requirements for successful information systems are not gathered or discovered, they are learned. More specifically, requirements are constructed knowledge formed through active collaboration of stakeholders. Too often, eliciting requirements is regarded by practitioners as the administrative preparation before commencement of the real, more challenging work. Such an attitude has doomed projects to fail. I maintain that constructing requirements is an integral part of the real work for any information system being developed or integrated.

As a crude analogy, sociologist Gerhard Lenski's typology shows the evolution of humans from hunters and gatherers through agricultural and industrial societies (1970). With each advancement, humans became more proactive and improved our ability to influence the future outcomes of our endeavors, leaving less to chance. Now we are in the information age, yet we continue to gather requirements for information systems. We need to evolve by changing our approach to recognize requirements as knowledge that we can and should proactively construct to improve the outcomes of information systems projects.

Philosopher Auguste Compte developed the idea in the 1830s that a society engaged in the quest for knowledge, or truth, progresses through three stages: theological, metaphysical, and eventually reaches what he termed the positive stage (Patton, 2002). Positivism promotes that "only verifiable claims based directly on experience could be considered genuine knowledge" (Patton, 2002, p. 92). Compte's position was to distinguish empirical positivist knowledge from metaphysics and theological beliefs which he considered less reliable. In other words, positivism holds that the truth *is out there* and can be gathered or experienced. In my professional experience, many information systems professionals approach requirements with a positivist attitude, applying the scientific methods we have been taught, and yet projects fail too often. Frequently when projects fail, blame is placed on being supplied with bad requirements. My theory proposes that requirements for information systems are not *out there* to be gathered or supplied, but instead must be constructed and refined in the minds of stakeholders as expectations.

Constructivism and social constructionism are more recently developed approaches to truth and knowledge. Promoted by developmental psychologist Jean Piaget in the 1960s, constructivism advocates that the physical world exists separately from our understanding of it and that knowledge about the world is constructed by each individual based on previous experiences (Piaget, 1970). At about the same time, Peter Berger and Thomas Luckmann wrote that knowledge is constructed by social interactions, popularizing social constructionism (Berger & Luckmann, 1967). Together, these approaches can be applied to information systems projects to help drive the evolution from *gathering* requirements to *constructing* requirements. This simple change of attitude may have significant impact when combined with other contemporary methods such as requirements engineering and Agile. From my perspective, the concept of constructing requirements helps explain why the prescribed iterative approach of Agile is effective.

Constructivist approaches have been applied to other creative endeavors. Muise and Wakkary found a constructivist framework for interactive design can articulate designer intentions and user experiences (2010). Constructivist language helps describe intentions and outcomes, which directly relate to requirements.

Gottesdiener (2002) encourages the workshop format for requirements, but she does not explicitly point out the constructivist approach and benefits. I believe the construction of knowledge occurs during these workshops both at the group and individual levels, applying philosophies of Piaget's constructivism (1970) as well as Berger and Luckmann's social constructionism (1967). Individuals construct and refine their own knowledge and understanding by gaining insight to the perspectives of other participants. The group as a whole also constructs knowledge through the process of negotiation and reconciliation of conflicts across the group. I submit that while this construction of requirements does occur, usually unconsciously, it is too often constrained by a traditional positivist approach assumed by many stakeholders.

Berg applies constructivism in his holistic description of implementing an information system as a "mutual transformation" where "the organization is affected by the coming of this new technology, but the technology is in its turn inevitably affected by the specific organizational dynamics of which it becomes a part." He continues to apply constructivism and social constructionism when he adds "The most 'successful' implementation processes appear to be those in which an obsession for control and planning is replaced by an obsession for experimentation and mutual learning" (2001, p. 154). This is consistent with my perspective and observations.

I observed the impact of a positivist approach on a recent project. The project leadership took an approach of divide and conquer for a large, complex effort. After only a few hours of introductions and high-level goal setting, the dozens of stakeholders were divided into 10 teams to develop their own component solutions. This approach is positivist because it assumes that since each team is comprised of experts, they will generate the right solution for their area and it will magically fit with the solutions developed by the other teams with complementary subject matter expertise. As a participant/researcher directly involved, I observed the challenges of this positivist approach, such as frustration among teams without access to knowledge held by other teams. Far too many assumptions were made, leading to re-work and frustration once there was coordination among the teams. Ultimately, after weeks of frustration and limited progress, a more constructivist approach emerged where most of the stakeholders came together in the same room and through open discussion began to construct a common understanding informed by all perspectives. This approach reduced frustrations about solution quality and the dependence on assumptions.

Applies to All Lifecycle Models

Information system lifecycle models have evolved over decades from the waterfall model, through modifications to waterfall and various iterative models, to the Agile model popular today. While these models prescribe requirements elicitation at different points and frequencies in each respective lifecycle, the approach used to elicit the requirements remains discretional. Even while following the Agile model, some practitioners continue to gather requirements with the same positivist approaches used for decades. For example, lists of requirements are often collected via email or by requesting stakeholders fill out forms. Aggregation of requirements from multiple stakeholders is often done by a small team or even an individual requirements “expert” or, in the jargon of Scrum, the product owner (Schwaber & Sutherland, 2013). Collaboration about the requirements is too often a way to address perceived problems rather than the way to construct knowledge. These positivist approaches have proven to yield limited success.

Constructing requirements adjusts the approach to acknowledge that requirements do not exist on their own to be gathered. Instead, they must be constructed through proactive collaboration and reconciliation among the stakeholders. The requirements may indeed differ depending on which stakeholders participate, and often change when stakeholders change.

The waterfall model elicits requirements only at the beginning and does not inherently accommodate change to requirements from learning (Royce, 1970). The waterfall model is perhaps a classic example of positivism. In the waterfall model, once the requirements are specified, they are considered written in stone and eventually implemented. When issues arise later, my experience is that the blame is often attributed to bad requirements rather than to improved understanding.

Frameworks that follow the Agile model, such as Scrum, address requirements continuously throughout the project. In Scrum parlance, the requirements are represented in the product backlog which is constantly updated by the product owner as learning occurs. The product owner has responsibility for managing the requirements, but is not the source of the requirements (Schwaber & Sutherland, 2013).

Regardless of the lifecycle model used, the source of requirements for an information system is the set of people who are impacted by the outcome of the project, the stakeholders. Users, executive sponsors, funding sources, and even the team members responsible for delivering the solution are stakeholders. All stakeholders are potential sources of valuable knowledge of requirements and constraints which can influence whether a project is ultimately perceived as successful. Requirements are stakeholder expectations which must be understood and unified before they can be managed or engineered.

Achieving a common understanding of requirements, including reconciling conflicts and validating assumptions, requires effort and a constructivist approach. In Scrum, the responsibility for promoting common understanding of requirements is assigned to the product owner (Schwaber & Sutherland, 2013). In other models, this task may be accomplished by project manager, solution architect, requirements engineer, or many other roles.

My interpretation is that eliciting requirements for information systems is an activity of learning by all the stakeholders. The stakeholders are not only the learners, but also the mutual sources of information that must be commonly understood in order for expectations to be met. Some stakeholders contribute expectations while others contribute constraints. Collaborative compromise is essential to achieve a common target. Much like a pot-luck meal, the outcome is completely dependent upon the contributions of the individual people involved.

Information system requirements are highly subjective. Even though information systems are developed using components governed by physical nature and scientific principles, judgment and perception of whether an information system meets needs is socially constructed. An information system is successful only if it is perceived to be successful by the stakeholders. There is no objective, positive, test that validates an information system. Attempts to create objective tests are themselves subjectively influenced by the stakeholders developing the tests.

Agile is Constructivist

The Agile approach includes guidelines that are proven effective and practical. Although I found no evidenced in literature that the developers of Agile were specifically guided by constructivism or social constructionism, I believe Agile is inherently constructivist based on its sensitivity to people and change. Results from this study have helped me understand more deeply why Agile approaches work. Subsequent research may examine more specifically the relationship between Agile methods and the philosophies of constructivism and social construction.

Evolution from waterfall to iterative models was a move from legacy positivist approach toward a more constructivist approach, acknowledging change as inevitable as time progresses. Agile methods, such as Scrum, anticipate change and present structure to accommodate change. Constructing requirements embraces the changes that occur as a result of learning over time and from new stakeholders.

Perhaps considering requirements with a constructivist attitude helps explain why the principles of *The Agile Manifesto* are effective. The first principle prioritizes "Individuals and interactions over processes and tools." This at least reflects the highly subjective nature of information systems. People's expectations dictate the requirements more than any established processes. These expectations must be understood and managed. Understanding processes and tools may be important to keep expectations reasonable, but ultimately it is the stakeholder expectations that matter and will influence whether or not the project is considered successful.

The second principle is “Working software over comprehensive documentation” which could translate to “seeing is believing” or even more simply “show me.” Documentation is sometimes a promise of what will happen, while working software is demonstration of what has happened already and can happen again. There is more certainty and factual knowledge with working demonstrable solution than words on a page.

“Customer collaboration over contract negotiation” is the third Agile principle and reinforces the need for collaboration and negotiation. If the stakeholders have an attitude of constructivism, knowing they will all have input to defining the solution, there can be less time spent negotiating the contract which by itself does not guarantee a solution. Comprehensive contracts are important, but including excessively detailed requirements at the time of contact negotiations harkens back to the waterfall model with its invitation for failure. Brooks and Edwards recommend focusing on outcomes when negotiating contracts. Through collaborative inquiry with clients, desired outcomes can be identified which help achieve a common understanding among all stakeholders. “The task in this phase is both to draw our clients out and to help them be more specific about what *they* want to see come out of this engagement” (2014, p. 23). Such collaboration to clarify and negotiate expectations enables a more effective contract, adheres to Agile principles, and constructs requirements.

The final Agile principle is “Responding to change over following a plan” which underscores that change is inevitable and must be anticipated and addressed. Ideally, changes should be welcomed and embraced as new knowledge which will make the solution even better. Too often changes are unnecessarily delayed or even avoided in order to follow a plan or schedule. Changes need to be managed, not ignored.

Appendix I is a model showing a traditional approach to information systems including the major components of *people*, *process*, and *technology*. A three legged stool is often used to represent these three interdependent components of the whole system. The model represents the forces of change, based on requirements, on the three elements.

Appendix J is a model representing a constructive approach to information systems. The same three components are represented, but in a different arrangement. In the constructivist model, the *people* component is moved to the more prominent position of the stool seat representing the special role people have in information systems with relation to the process and technology legs. The model shows the challenges identified by this study as filters that influence the interpretation of influence and knowledge people encounter. Within the scope of an information system, while people have control over process and technology, people have only influence over other people. That influence is filtered by the challenges identified by this study.

Unanticipated Results

Although the questions I prepared for my semi structured interview guide attempted to focus discussion on projects that involved failure, almost all of the participants provided responses at some point during their interview that referenced characteristics of successful projects. I noted this trend about halfway through the interviews and so adjusted my expectations about the subsequent interviews to welcome and encourage discussion about experiences related to requirements on successful information systems projects. This adjustment provided a needed balance for the participants to recall and share experiences with both success and failure.

One challenge that emerged from analysis of the data was the impact of cultural diversity on eliciting requirements. While I had some level of expectation that various challenges to effective communication would surface, I had not considered the specific challenges associated with perspectives that differ based on gender, age, or understanding based on cultural background. These challenges are clearly consistent with application of constructivism and social construction. The fact that these were not anticipated underscores the need and difficulty of proactively recognizing and incorporating the varied perspectives of all stakeholders on a project. It is better to be surprised by unanticipated requirements up front rather than unintended consequences at the end.

Also unanticipated was the connection to Agile methods. Before beginning this study, my exposure to Agile and Scrum was superficial at best. I understood the prescribed processes, but I had not connected the dots between the mechanics of iteration with the learning that I now see is constantly occurring. For me, this was an epiphany that contributed significantly to the theory that emerged from the data.

Summary

In this chapter, I presented my interpretation of the findings from this study in the form of a theory that emerged from the data. The theory proposes that information systems requirements are a form of knowledge to be learned, not gathered, and that an approach combining constructivism with social constructionism will be more effective than traditional positivist approaches. While the benefits of a constructivist approach apply regardless of the lifecycle model followed, the documented perceptions of Agile methods as contributing to information system project success add support to the theory. Agile methods are inherently constructivist as the prescribed repetition and social interaction not only enable but encourage learning and change to occur throughout the project lifecycle. The chapter closes with a discussion of unanticipated results from the study. The next chapter presents conclusions and implications.

VI. CONCLUSIONS AND IMPLICATIONS

Review

This study was conducted to address the question of "What challenges are encountered by stakeholders when identifying requirements for information systems?" The challenges identified by this study emerged from analysis of the data collected and are supported by literature as well as reinforced my own observations from real-world projects. Based on my interpretation of the findings, I offer the following conclusions and implications.

Implications for Theory

The results of this study highlighted challenges which were organized into the categories of change, communications, and knowledge. These categories, individually and in combination, are prime candidates for further research to explore their association with information systems requirements. Fields such as conflict resolution and change management now also appear to be more clearly relevant to information systems requirements elicitation and could be perspectives for future theories to be developed.

Implications for Practice

Practitioners may benefit from this study by leveraging the findings and resulting theory to improve their own requirements elicitation practices and improve project outcomes. When faced with challenges similar to those presented in this study, practitioners now have an additional reference to help justify appropriate mitigations in order to increase their likelihood for success.

To maximize the likelihood that the outcome of an information systems project will be considered successful, the stakeholders must have common expectations. As stated earlier, an information system is successful only if it is perceived to be successful by the stakeholders. Collaboration among stakeholders, including reconciling conflicts and resolving assumptions, contributes to achieving a common understanding among participating stakeholders. Following are recommendations for practitioners to benefit from the theory of constructing requirements.

An overarching recommendation is to approach requirements for information systems as knowledge to be proactively constructed via collaboration and negotiation among all stakeholders. The requirements elicitation process is about collaboration and learning to build new knowledge, not simply to discover and compile existing knowledge. All stakeholders should enter the requirements process expecting to learn the needs of others and also to teach others about their own needs. Conflicting requirements should be expected and proactively addressed through compromise and reconciliation. An outside facilitator may be helpful in certain circumstances. Emphasis should be on desired outcomes rather than problems from the past, reinforcing that something new, involving change, is the objective.

Another key recommendation is to consider requirements to be an abstraction of the stakeholders' expectations rather than somehow originating from the target system. The information system is the outcome of the effort, not the source of requirements. The source of requirements is always the stakeholders involved.

Other recommendations are to ensure all current stakeholders are represented during the collaboration and that all are committed and empowered to fully engage. Power dynamics, egos, and cultural differences must be proactively mitigated to achieve full engagement from all stakeholders. Effective facilitation of collaborative requirements workshops as described by Gottesdiener (2002) and also Leffingwell (2014) can come from internal or external sources. If effective leadership and facilitation does not materialize naturally from within the stakeholder team, an outside facilitator may help optimize the participation and contributions of all stakeholders.

It is also recommended to verify requirements anytime there is a turnover among the stakeholders. Even if a stakeholder is replaced with someone from the same functional team, the new individual may have different priorities, experiences, and understandings that could result in weakened commitment to the previously established expectations. Stakeholder teams are organic, not mechanistic. Changing out one person can yield an entirely new team with new priorities and perspectives.

Minimize turnover when possible. This can be done by respecting the importance of committing resources to the project until it is finished, or at least until major milestones are completed. Consider stakeholder turnover as a major milestone where a new team is constituted. Agile methods, such as Scrum, help address turnover. Time-boxing sprints to no more than one month of effort increases the likelihood that a functional milestone will be achieved before the requirements change. When turnover does occur, the work already completed in earlier sprints to meet previous backlog requirements may influence the new stakeholder group to align understanding more rapidly than if the work was still considered in progress and therefore more subject to adjustment.

Short feedback loops combined with including stakeholders in all phases of an Agile project ensure work does not get too far off track. Each cycle of feedback is both a course correction as well as an opportunity for all stakeholders to learn and revise the requirements for the next cycle.

Practitioners involved with requirements elicitation should proactively and confidently address the challenges identified in this study as well as others that emerge from future research. Responsible practitioners cannot assume that someone else will address issues or that issues will resolve over time. All stakeholders should have the responsibility and opportunity to ensure that as issues arise they are addressed and not simply carried forward or ignored. Confidence and courage is required because postponing challenging decisions has led many projects to failure and wasted effort. Leaders concerned with optimizing resources should avoid commencing an activity without a reasonable understanding that the effort will be perceived as successful. In Scrum, the activity of sprint planning and managing the product backlog help address this challenge.

Acknowledge and proactively address imbalances of power among stakeholders to optimize collaboration and social construction of requirements. Utilize an external facilitator if needed to equalize power among stakeholders. In Scrum, the product owner has the responsibility for ensuring all perspectives are reconciled and represented in the product backlog. Organizational leadership must support the person or team involved with requirements elicitation by recognizing the value of constructing consensus knowledge and understanding rather than dictating based on positional authority.

Recognize that even trusted resources may have misunderstandings about expectations and that perspectives will change through learning. Trust, but verify using solid project management practices to avoid surprises and disappointments. In Scrum, the daily 15-minute meeting among the delivery team helps keep activities synchronized, including introduction of new information such as emerging obstacles. A project that is important enough to consume organizational resources should be well managed to optimize the return on that investment.

Appropriately acknowledge and leverage the unique contributions of every stakeholder as an individual. Some stakeholders will offer needs as expectations while others, such as the delivery team, will express constraints which serve to manage expectations. If a stakeholder does not contribute a perspective, that person may not be needed as a stakeholder. One way to address diverse perspectives is to include representatives of the delivery team along with all other stakeholders as requirements are constructed and to ensure all perspectives are empowered to ask for clarification and address issues. Technology solutions such as instant messaging and virtual meetings may be useful to enable frequent synchronous collaboration among stakeholders in order to close gaps in common understanding. In my experience, relying on asynchronous communication such as email and exchanging documents often slows collaboration.

In my experience, almost all projects begin with some assumptions along with initial requirements. Some assumptions are candidate requirements that have not yet been validated via collaboration among the stakeholders and authoritative sources. Effort is required to validate assumption to either confirm or dismiss the assumption as a requirement. Other assumptions are about events that have not yet happened and can only be monitored for the outcome. Resolve assumptions as soon as possible once identified. Perpetuating assumptions longer than necessary may lead to a cascade of misinformed decisions. One approach is to assign each assumption an expiration date and owner responsible for promoting the assumption to a requirement if validated or removing the assumption if determined false.

Conflicting requirements should also be resolved as soon as possible. Avoiding resolution of difficult conflicts has led to wasted effort that was later abandoned when the conflict was found to be unresolvable. Utilize an experienced project manager to arbitrate passionate perspectives. If there is no solution, don't waste resources on imminent failure.

"A delayed project that succeeds is more valuable, and less expensive, than a project started on-time that fails" (Leviss, 2010, p. 17). Schedule deadlines are sometimes important requirements to be met in order for overall success to be achieved. In those cases, resolving conflicting requirements early is even more important to determine whether success is even possible within time constraints.

Conflict avoidance lead to failure of some projects that never should have been started. Proper attention to requirements would force reconciliation of expectations earlier. This challenge wastes resources as teams unknowingly work toward misunderstood or downright wrong requirements. Conflicting priorities and lack of accountability often lead to this challenge, especially if there is reason to expect payment for effort will be made even if the solution fails. This lack of accountability is a problem with certain contracting approaches.

Confidence is an essential element and must be present in the appropriate proportion. Pay attention to levels of confidence. Both overconfidence and lack of confidence have had negative impacts on information systems projects. Verify assertions of capability, even by trusted partners. Monitor and track progress appropriately for the risks involved by applying proven project management principles. In cases where a lack of confidence may be an issue for a key perspective, use a facilitator to stimulate full participation. Stakeholders lacking sufficient confidence may not speak up when needed to provide input that could guide expectations to be more accurate. During requirements construction, all participants must be and feel empowered to speak up and collaborate equally in order to expose and reconcile conflicting requirements. An imbalance of power may impact levels of confidence and threaten the requirements process.

Proactively address any communication problems. Over-communicate details using a variety of methods to ensure all stakeholders have a common understanding of requirements. If the challenge of achieving effective communication is not met, the project is in jeopardy of being perceived as a failure. If necessary, consider changing the scope of the project by changing the set of stakeholders to a more effective group which may be either smaller or larger or simply different people.

Final Remarks

Following these, or any, recommendations cannot guarantee success for any project, but can help improve the likelihood of success by proactively addressing challenges that have led others to failure. Practitioners should use the knowledge generated by this study to strengthen their confidence and to boldly take actions that will construct requirements for their own information systems projects. Researchers should continue to study and report how the philosophies of constructivism and social constructivism relate to the effectiveness of Agile and other collaborative methods. Helping practitioners understand how and why constructivist methods work will help accelerate the evolution from gathering requirements to constructing requirements.

APPENDIX SECTION

APPENDIX A

THE AGILE MANIFESTO

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

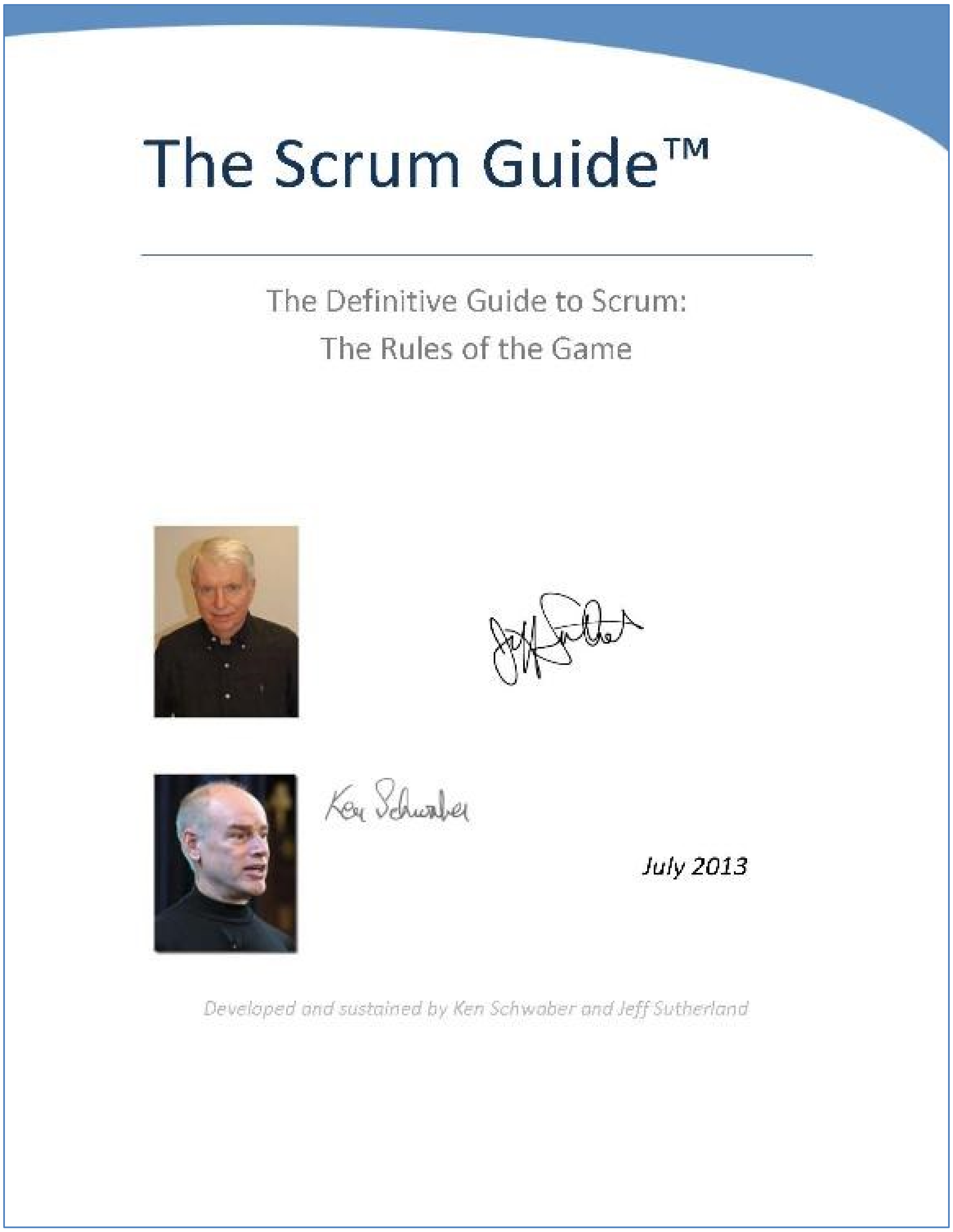
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

APPENDIX B

THE SCRUM GUIDE



The Scrum Guide™

The Definitive Guide to Scrum:
The Rules of the Game

[illegible]

Ken Schwaber

July 2013

Developed and sustained by Ken Schwaber and Jeff Sutherland

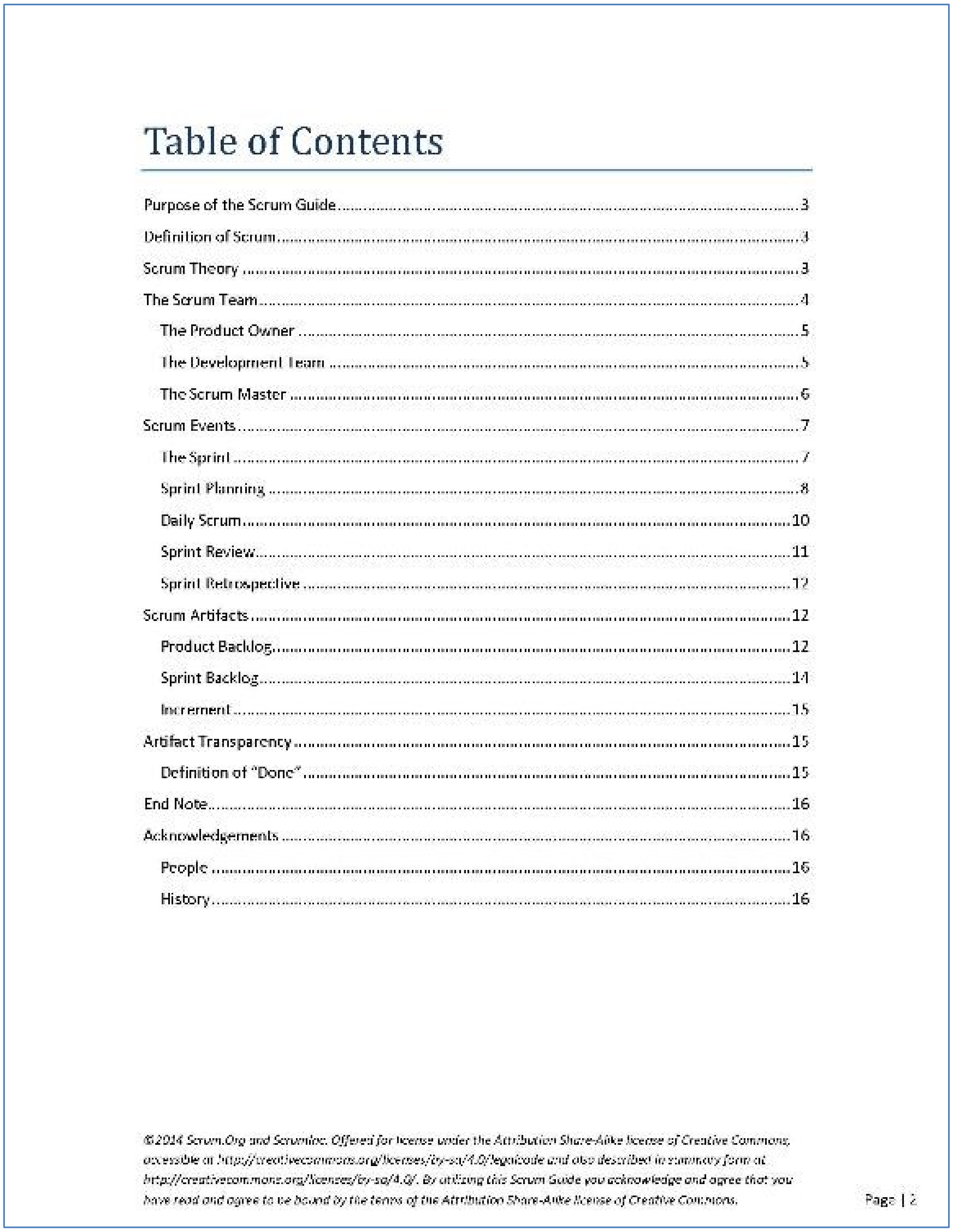
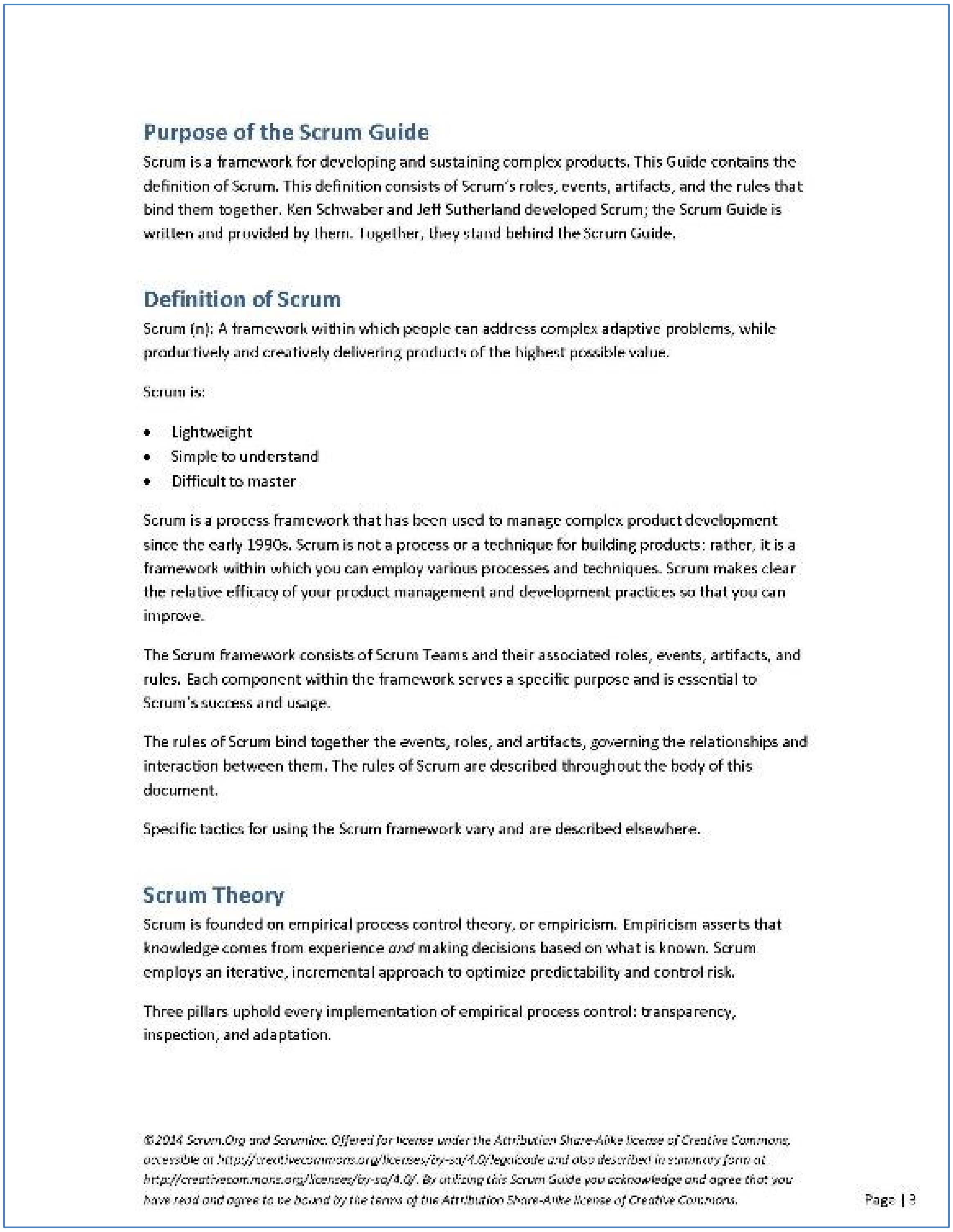


Table of Contents



Purpose of the Scrum Guide

Scrum is a framework for developing and sustaining complex products. This Guide contains the definition of Scrum. This definition consists of Scrum's roles, events, artifacts, and the rules that bind them together. Ken Schwaber and Jeff Sutherland developed Scrum; the Scrum Guide is written and provided by them. Together, they stand behind the Scrum Guide.

Definition of Scrum

Scrum (n): A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.

Scrum is:

- Lightweight
- Simple to understand
- Difficult to master

Scrum is a process framework that has been used to manage complex product development since the early 1990s. Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques. Scrum makes clear the relative efficacy of your product management and development practices so that you can improve.

The Scrum framework consists of Scrum Teams and their associated roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum's success and usage.

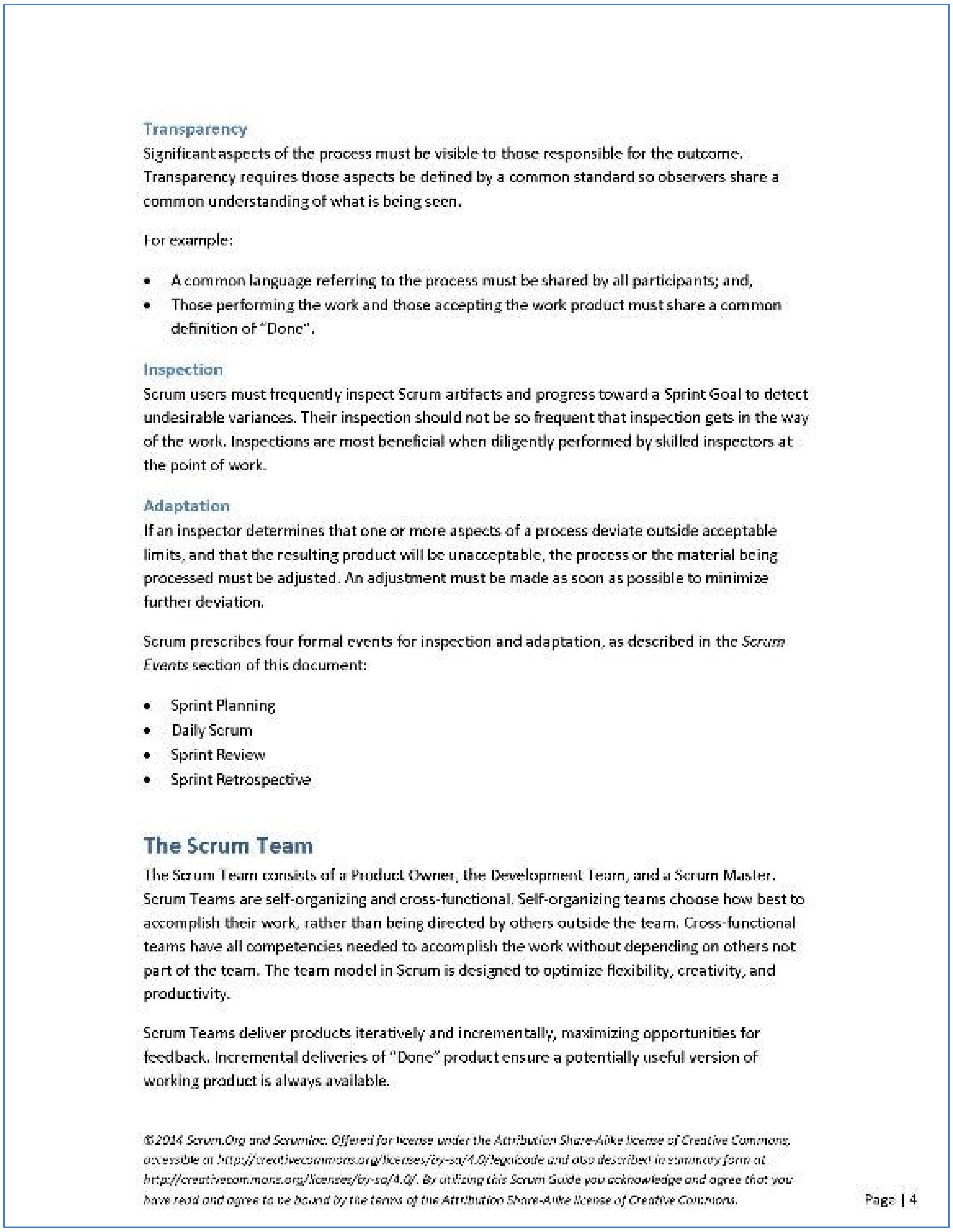
The rules of Scrum bind together the events, roles, and artifacts, governing the relationships and interaction between them. The rules of Scrum are described throughout the body of this document.

Specific tactics for using the Scrum framework vary and are described elsewhere.

Scrum Theory

Scrum is founded on empirical process control theory, or empiricism. Empiricism asserts that knowledge comes from experience *and* making decisions based on what is known. Scrum employs an iterative, incremental approach to optimize predictability and control risk.

Three pillars uphold every implementation of empirical process control: transparency, inspection, and adaptation.



Transparency

Significant aspects of the process must be visible to those responsible for the outcome. Transparency requires those aspects be defined by a common standard so observers share a common understanding of what is being seen.

For example:

- A common language referring to the process must be shared by all participants; and,
- Those performing the work and those accepting the work product must share a common definition of "Done".

Inspection

Scrum users must frequently inspect Scrum artifacts and progress toward a Sprint Goal to detect undesirable variances. Their inspection should not be so frequent that inspection gets in the way of the work. Inspections are most beneficial when diligently performed by skilled inspectors at the point of work.

Adaptation

If an inspector determines that one or more aspects of a process deviate outside acceptable limits, and that the resulting product will be unacceptable, the process or the material being processed must be adjusted. An adjustment must be made as soon as possible to minimize further deviation.

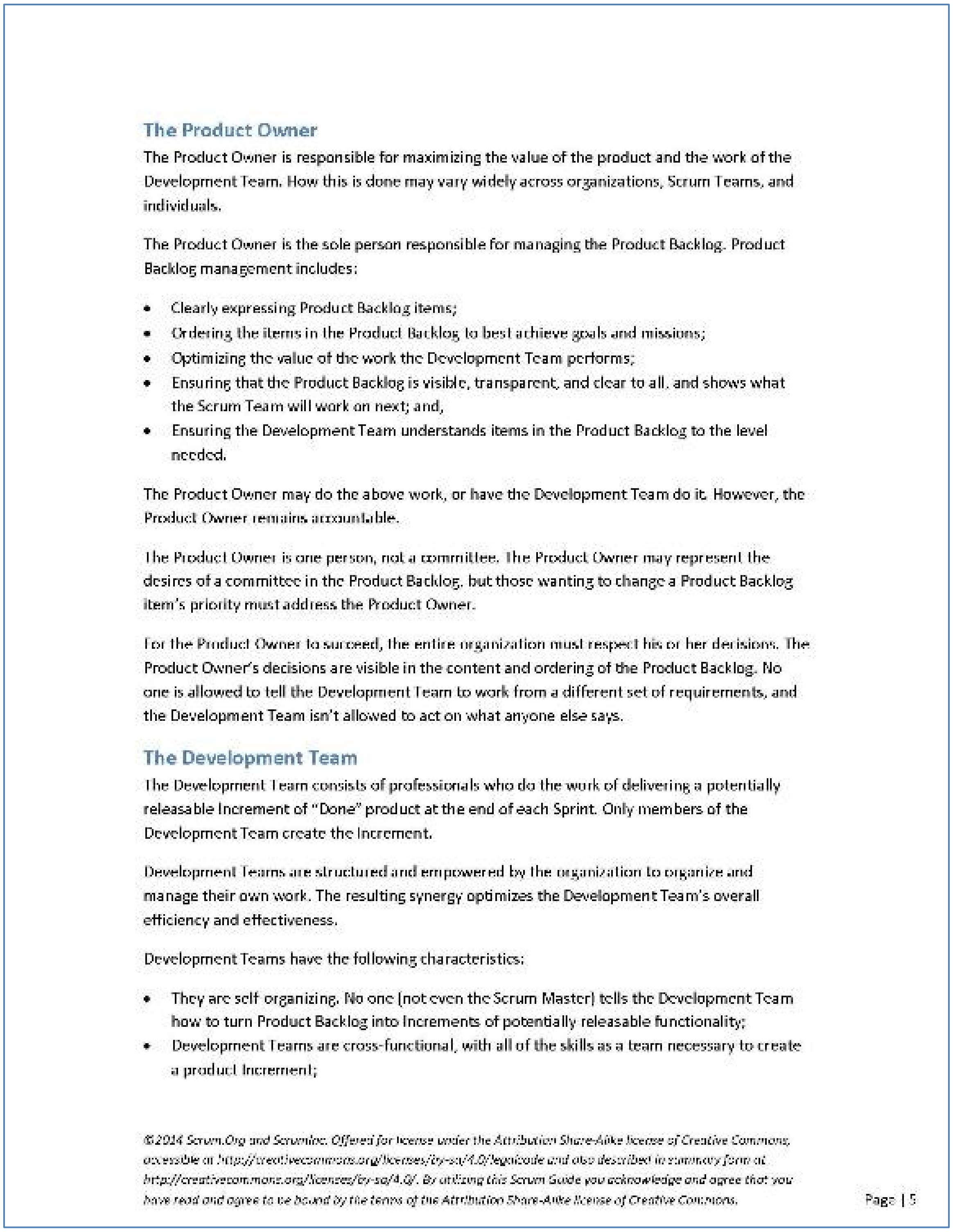
Scrum prescribes four formal events for inspection and adaptation, as described in the *Scrum Events* section of this document:

- Sprint Planning
- Daily Scrum
- Sprint Review
- Sprint Retrospective

The Scrum Team

The Scrum Team consists of a Product Owner, the Development Team, and a Scrum Master. Scrum Teams are self-organizing and cross-functional. Self-organizing teams choose how best to accomplish their work, rather than being directed by others outside the team. Cross-functional teams have all competencies needed to accomplish the work without depending on others not part of the team. The team model in Scrum is designed to optimize flexibility, creativity, and productivity.

Scrum Teams deliver products iteratively and incrementally, maximizing opportunities for feedback. Incremental deliveries of "Done" product ensure a potentially useful version of working product is always available.



The Product Owner

The Product Owner is responsible for maximizing the value of the product and the work of the Development Team. How this is done may vary widely across organizations, Scrum Teams, and individuals.

The Product Owner is the sole person responsible for managing the Product Backlog. Product Backlog management includes:

- Clearly expressing Product Backlog items;
- Ordering the items in the Product Backlog to best achieve goals and missions;
- Optimizing the value of the work the Development Team performs;
- Ensuring that the Product Backlog is visible, transparent, and clear to all, and shows what the Scrum Team will work on next; and,
- Ensuring the Development Team understands items in the Product Backlog to the level needed.

The Product Owner may do the above work, or have the Development Team do it. However, the Product Owner remains accountable.

The Product Owner is one person, not a committee. The Product Owner may represent the desires of a committee in the Product Backlog, but those wanting to change a Product Backlog item's priority must address the Product Owner.

For the Product Owner to succeed, the entire organization must respect his or her decisions. The Product Owner's decisions are visible in the content and ordering of the Product Backlog. No one is allowed to tell the Development Team to work from a different set of requirements, and the Development Team isn't allowed to act on what anyone else says.

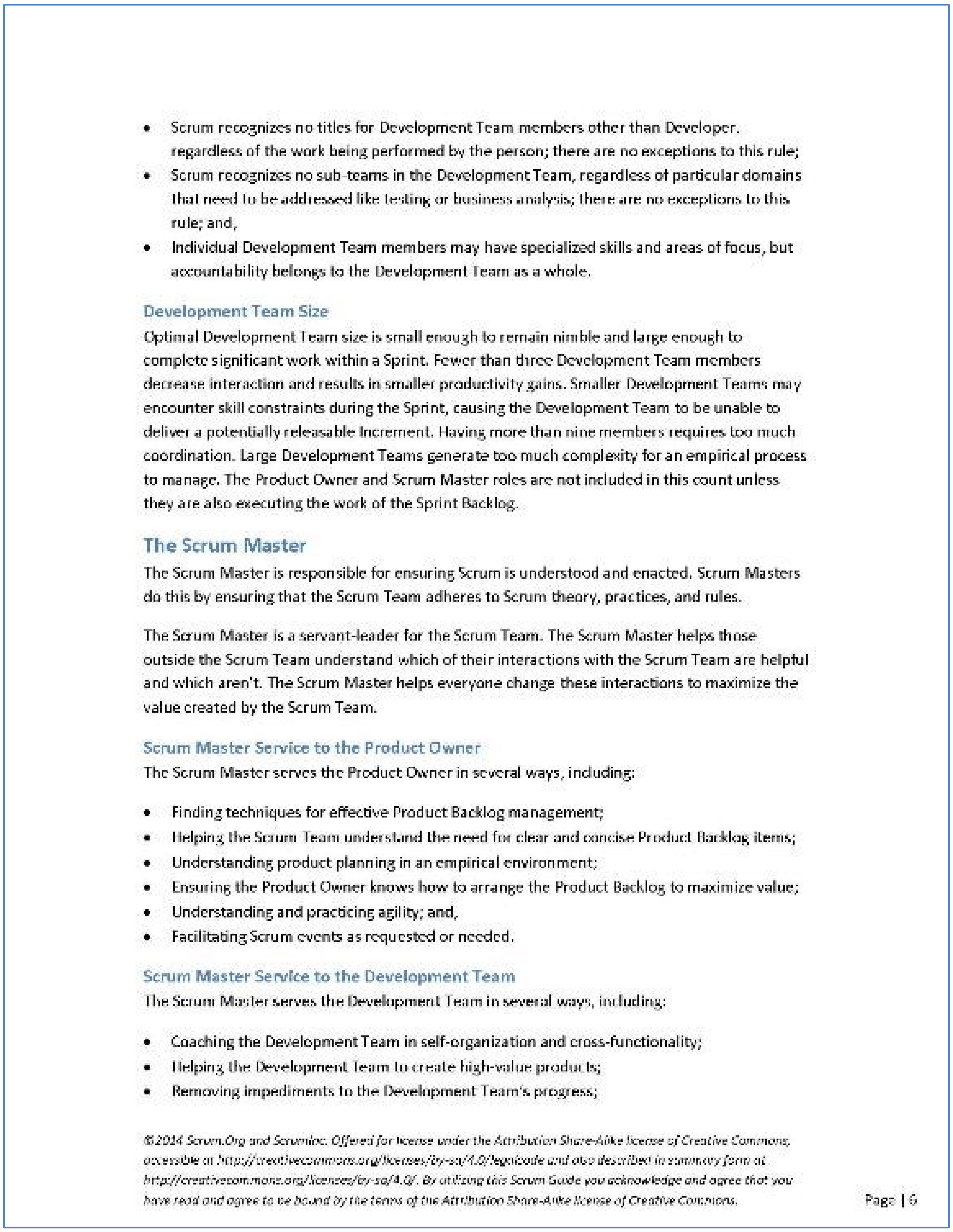
The Development Team

The Development Team consists of professionals who do the work of delivering a potentially releasable Increment of "Done" product at the end of each Sprint. Only members of the Development Team create the Increment.

Development Teams are structured and empowered by the organization to organize and manage their own work. The resulting synergy optimizes the Development Team's overall efficiency and effectiveness.

Development Teams have the following characteristics:

- They are self-organizing. No one (not even the Scrum Master) tells the Development Team how to turn Product Backlog into Increments of potentially releasable functionality;
- Development Teams are cross-functional, with all of the skills as a team necessary to create a product Increment;



- Scrum recognizes no titles for Development Team members other than Developer, regardless of the work being performed by the person; there are no exceptions to this rule;
- Scrum recognizes no sub-teams in the Development Team, regardless of particular domains that need to be addressed like testing or business analysis; there are no exceptions to this rule; and,
- Individual Development Team members may have specialized skills and areas of focus, but accountability belongs to the Development Team as a whole.

Development Team Size

Optimal Development Team size is small enough to remain nimble and large enough to complete significant work within a Sprint. Fewer than three Development Team members decrease interaction and results in smaller productivity gains. Smaller Development Teams may encounter skill constraints during the Sprint, causing the Development Team to be unable to deliver a potentially releasable Increment. Having more than nine members requires too much coordination. Large Development Teams generate too much complexity for an empirical process to manage. The Product Owner and Scrum Master roles are not included in this count unless they are also executing the work of the Sprint Backlog.

The Scrum Master

The Scrum Master is responsible for ensuring Scrum is understood and enacted. Scrum Masters do this by ensuring that the Scrum Team adheres to Scrum theory, practices, and rules.

The Scrum Master is a servant-leader for the Scrum Team. The Scrum Master helps those outside the Scrum Team understand which of their interactions with the Scrum Team are helpful and which aren't. The Scrum Master helps everyone change these interactions to maximize the value created by the Scrum Team.

Scrum Master Service to the Product Owner

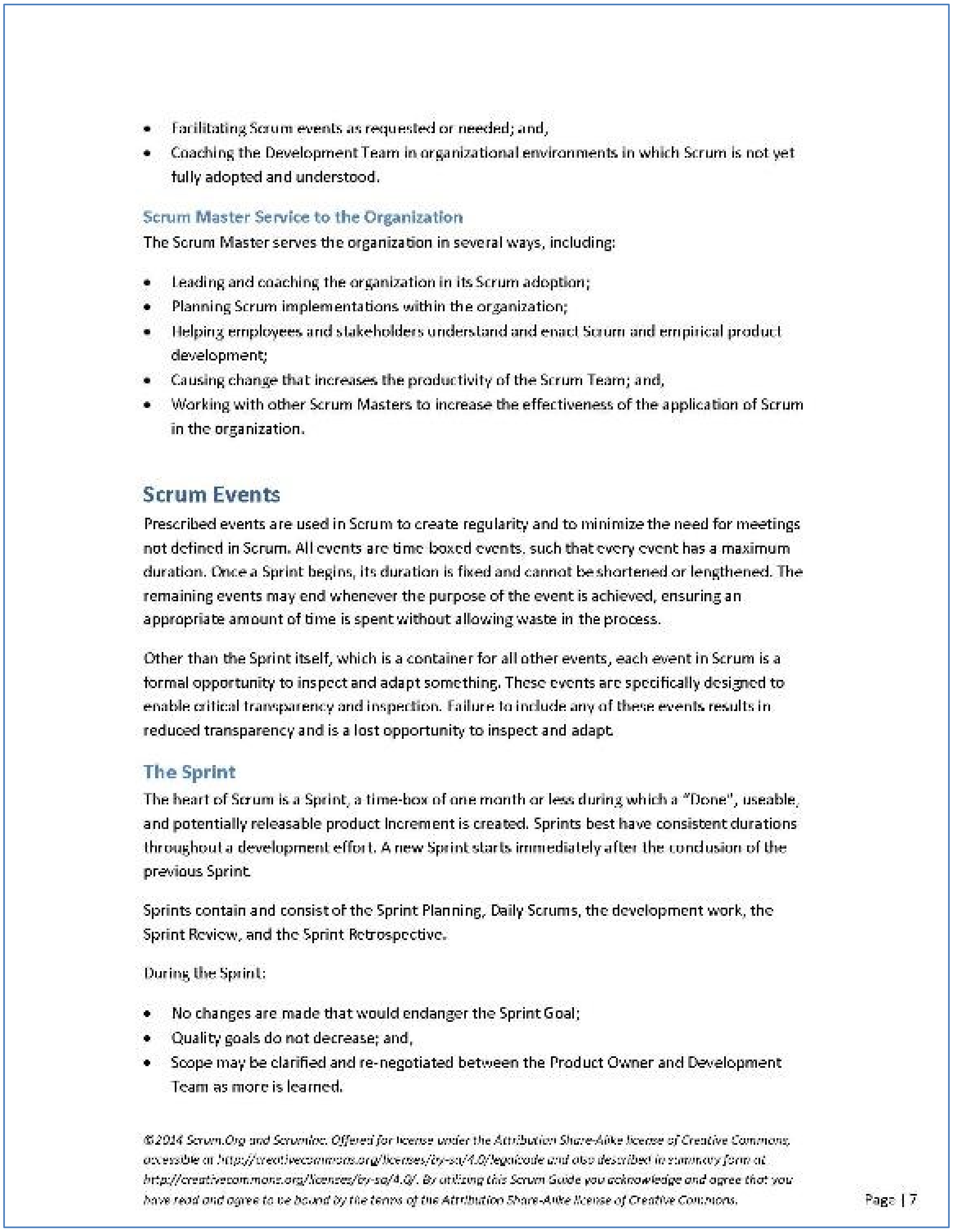
The Scrum Master serves the Product Owner in several ways, including:

- Finding techniques for effective Product Backlog management;
- Helping the Scrum Team understand the need for clear and concise Product Backlog items;
- Understanding product planning in an empirical environment;
- Ensuring the Product Owner knows how to arrange the Product Backlog to maximize value;
- Understanding and practicing agility; and,
- Facilitating Scrum events as requested or needed.

Scrum Master Service to the Development Team

The Scrum Master serves the Development Team in several ways, including:

- Coaching the Development Team in self-organization and cross-functionality;
- Helping the Development Team to create high-value products;
- Removing impediments to the Development Team's progress;



- Facilitating Scrum events as requested or needed; and,
- Coaching the Development Team in organizational environments in which Scrum is not yet fully adopted and understood.

Scrum Master Service to the Organization

The Scrum Master serves the organization in several ways, including:

- Leading and coaching the organization in its Scrum adoption;
- Planning Scrum implementations within the organization;
- Helping employees and stakeholders understand and enact Scrum and empirical product development;
- Causing change that increases the productivity of the Scrum Team; and,
- Working with other Scrum Masters to increase the effectiveness of the application of Scrum in the organization.

Scrum Events

Prescribed events are used in Scrum to create regularity and to minimize the need for meetings not defined in Scrum. All events are time-boxed events, such that every event has a maximum duration. Once a Sprint begins, its duration is fixed and cannot be shortened or lengthened. The remaining events may end whenever the purpose of the event is achieved, ensuring an appropriate amount of time is spent without allowing waste in the process.

Other than the Sprint itself, which is a container for all other events, each event in Scrum is a formal opportunity to inspect and adapt something. These events are specifically designed to enable critical transparency and inspection. Failure to include any of these events results in reduced transparency and is a lost opportunity to inspect and adapt.

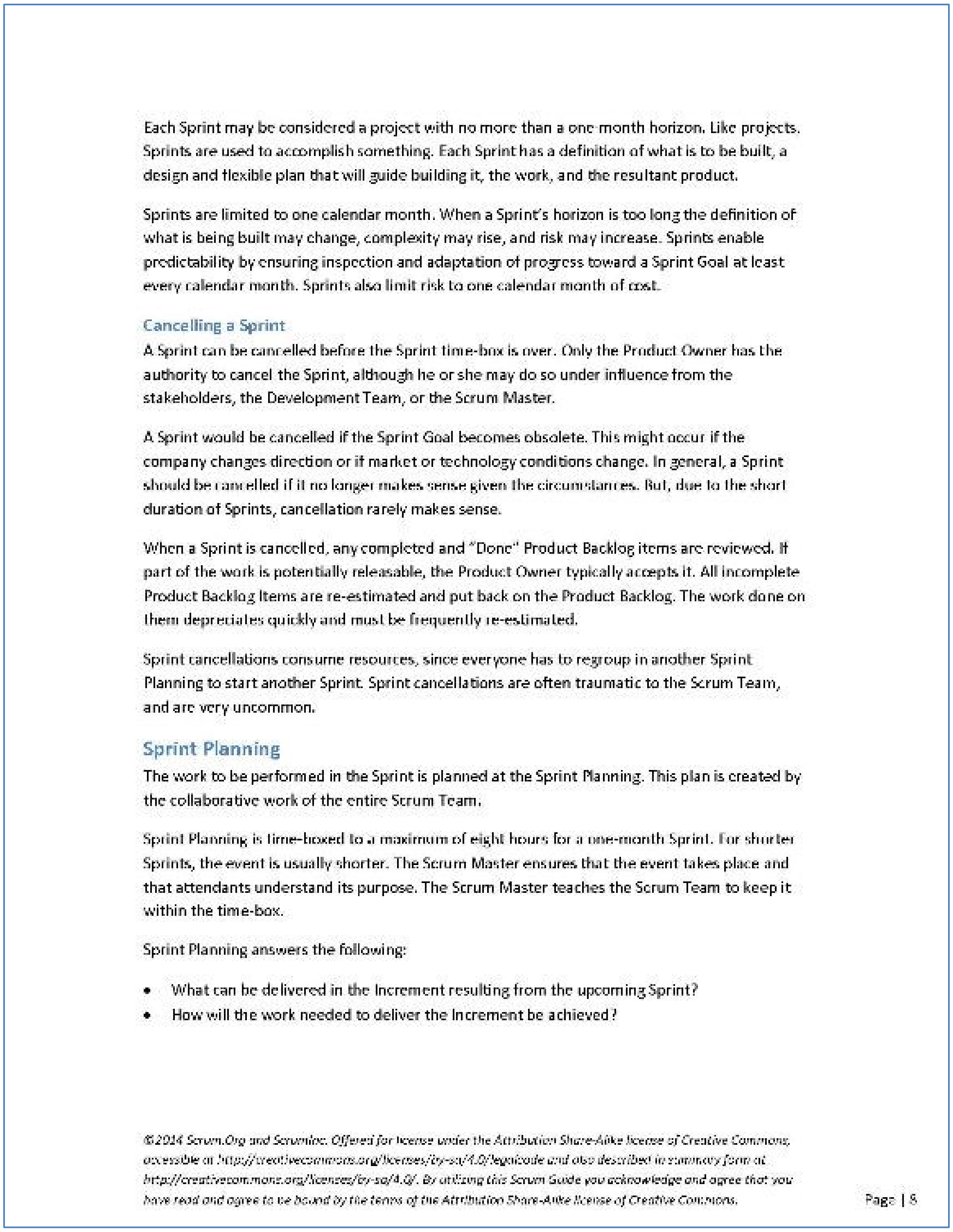
The Sprint

The heart of Scrum is a Sprint, a time-box of one month or less during which a "Done", useable, and potentially releasable product Increment is created. Sprints best have consistent durations throughout a development effort. A new Sprint starts immediately after the conclusion of the previous Sprint.

Sprints contain and consist of the Sprint Planning, Daily Scrums, the development work, the Sprint Review, and the Sprint Retrospective.

During the Sprint:

- No changes are made that would endanger the Sprint Goal;
- Quality goals do not decrease; and,
- Scope may be clarified and re-negotiated between the Product Owner and Development Team as more is learned.



Each Sprint may be considered a project with no more than a one-month horizon. Like projects, Sprints are used to accomplish something. Each Sprint has a definition of what is to be built, a design and flexible plan that will guide building it, the work, and the resultant product.

Sprints are limited to one calendar month. When a Sprint's horizon is too long the definition of what is being built may change, complexity may rise, and risk may increase. Sprints enable predictability by ensuring inspection and adaptation of progress toward a Sprint Goal at least every calendar month. Sprints also limit risk to one calendar month of cost.

Cancelling a Sprint

A Sprint can be cancelled before the Sprint time-box is over. Only the Product Owner has the authority to cancel the Sprint, although he or she may do so under influence from the stakeholders, the Development Team, or the Scrum Master.

A Sprint would be cancelled if the Sprint Goal becomes obsolete. This might occur if the company changes direction or if market or technology conditions change. In general, a Sprint should be cancelled if it no longer makes sense given the circumstances. But, due to the short duration of Sprints, cancellation rarely makes sense.

When a Sprint is cancelled, any completed and "Done" Product Backlog items are reviewed. If part of the work is potentially releasable, the Product Owner typically accepts it. All incomplete Product Backlog Items are re-estimated and put back on the Product Backlog. The work done on them depreciates quickly and must be frequently re-estimated.

Sprint cancellations consume resources, since everyone has to regroup in another Sprint Planning to start another Sprint. Sprint cancellations are often traumatic to the Scrum Team, and are very uncommon.

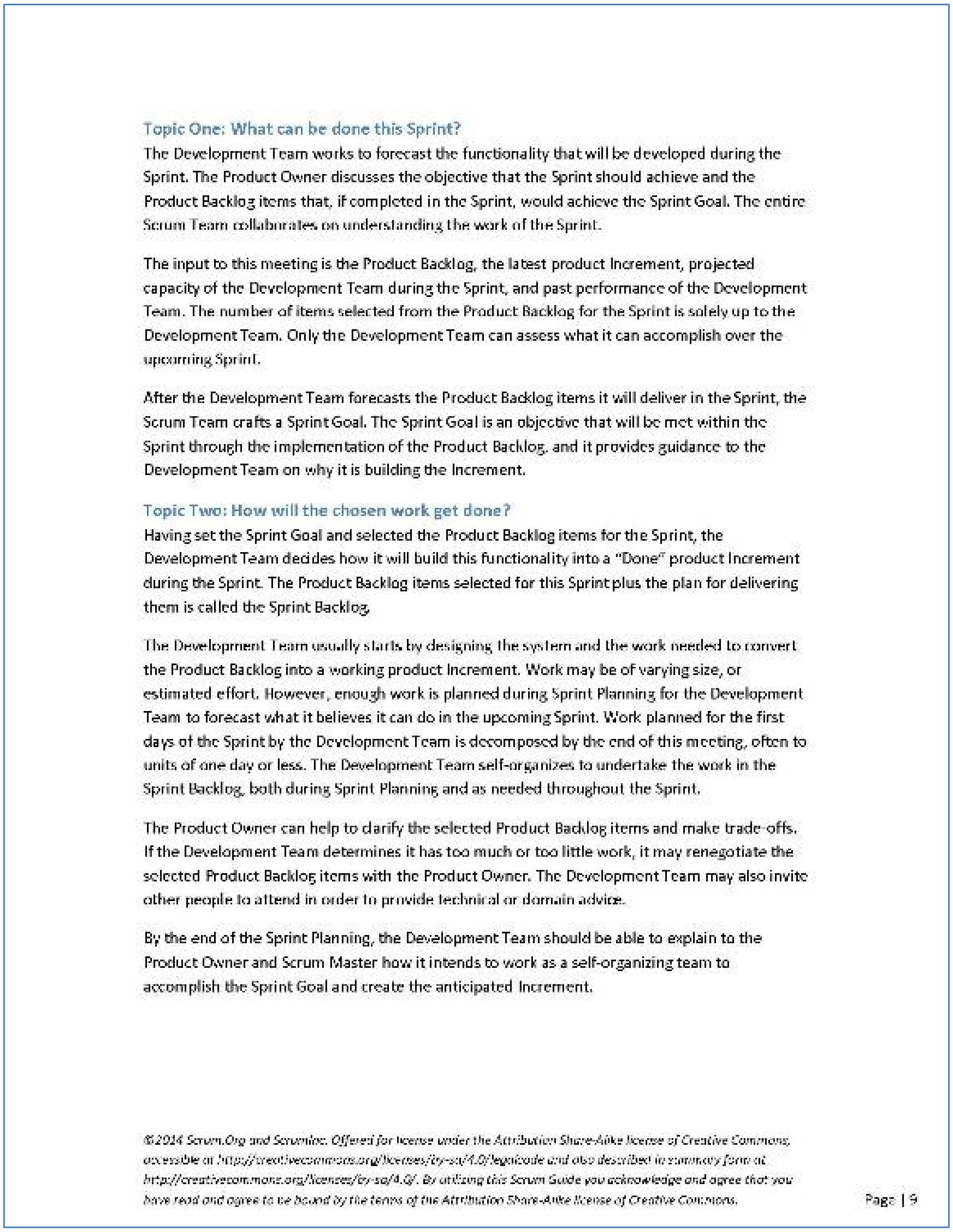
Sprint Planning

The work to be performed in the Sprint is planned at the Sprint Planning. This plan is created by the collaborative work of the entire Scrum Team.

Sprint Planning is time-boxed to a maximum of eight hours for a one-month Sprint. For shorter Sprints, the event is usually shorter. The Scrum Master ensures that the event takes place and that attendants understand its purpose. The Scrum Master teaches the Scrum Team to keep it within the time-box.

Sprint Planning answers the following:

- What can be delivered in the Increment resulting from the upcoming Sprint?
- How will the work needed to deliver the Increment be achieved?



Topic One: What can be done this Sprint?

The Development Team works to forecast the functionality that will be developed during the Sprint. The Product Owner discusses the objective that the Sprint should achieve and the Product Backlog items that, if completed in the Sprint, would achieve the Sprint Goal. The entire Scrum Team collaborates on understanding the work of the Sprint.

The input to this meeting is the Product Backlog, the latest product Increment, projected capacity of the Development Team during the Sprint, and past performance of the Development Team. The number of items selected from the Product Backlog for the Sprint is solely up to the Development Team. Only the Development Team can assess what it can accomplish over the upcoming Sprint.

After the Development Team forecasts the Product Backlog items it will deliver in the Sprint, the Scrum Team crafts a Sprint Goal. The Sprint Goal is an objective that will be met within the Sprint through the implementation of the Product Backlog, and it provides guidance to the Development Team on why it is building the Increment.

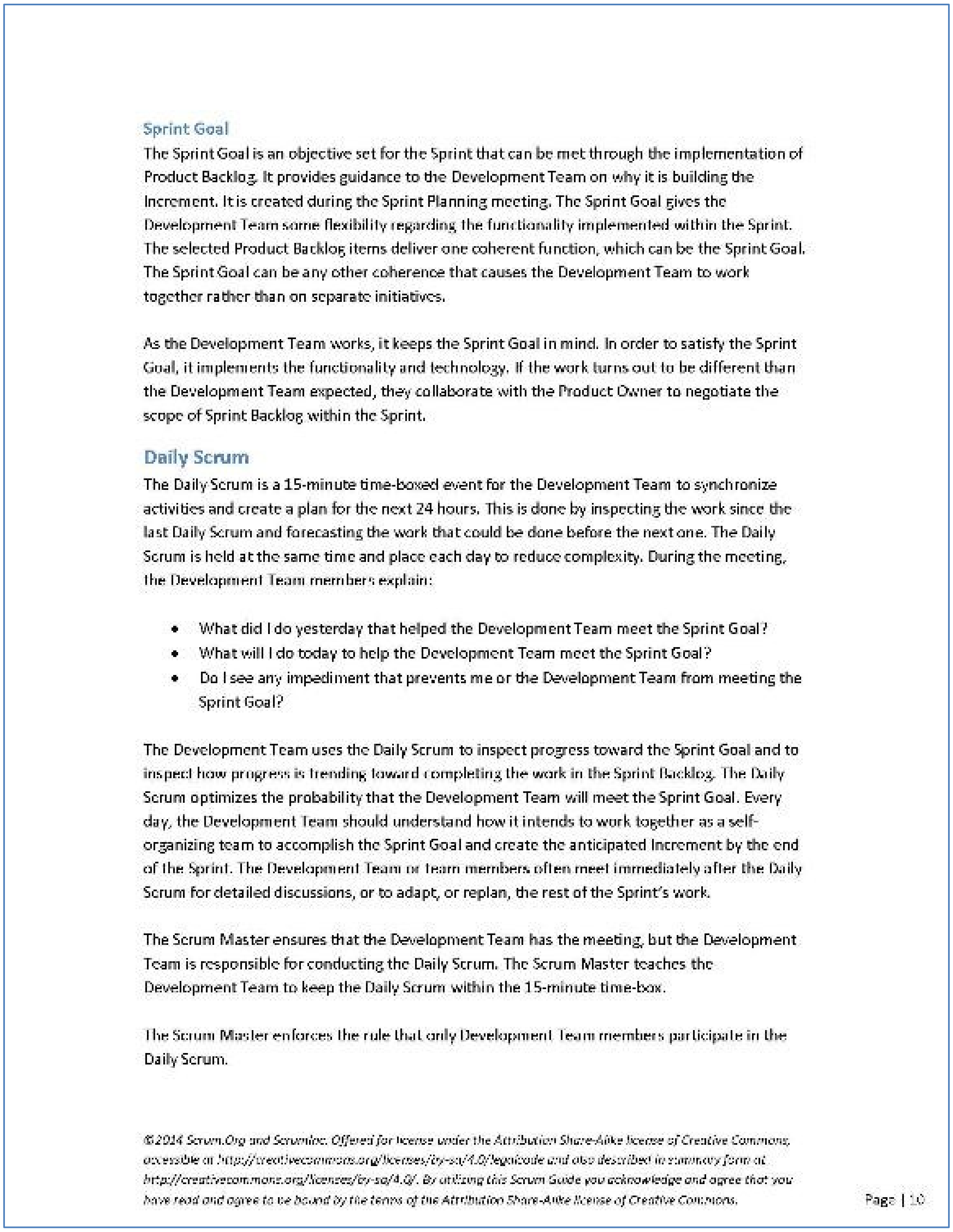
Topic Two: How will the chosen work get done?

Having set the Sprint Goal and selected the Product Backlog items for the Sprint, the Development Team decides how it will build this functionality into a "Done" product Increment during the Sprint. The Product Backlog items selected for this Sprint plus the plan for delivering them is called the Sprint Backlog.

The Development Team usually starts by designing the system and the work needed to convert the Product Backlog into a working product Increment. Work may be of varying size, or estimated effort. However, enough work is planned during Sprint Planning for the Development Team to forecast what it believes it can do in the upcoming Sprint. Work planned for the first days of the Sprint by the Development Team is decomposed by the end of this meeting, often to units of one day or less. The Development Team self-organizes to undertake the work in the Sprint Backlog, both during Sprint Planning and as needed throughout the Sprint.

The Product Owner can help to clarify the selected Product Backlog items and make trade-offs. If the Development Team determines it has too much or too little work, it may renegotiate the selected Product Backlog items with the Product Owner. The Development Team may also invite other people to attend in order to provide technical or domain advice.

By the end of the Sprint Planning, the Development Team should be able to explain to the Product Owner and Scrum Master how it intends to work as a self-organizing team to accomplish the Sprint Goal and create the anticipated Increment.



Sprint Goal

The Sprint Goal is an objective set for the Sprint that can be met through the implementation of Product Backlog. It provides guidance to the Development Team on why it is building the Increment. It is created during the Sprint Planning meeting. The Sprint Goal gives the Development Team some flexibility regarding the functionality implemented within the Sprint. The selected Product Backlog items deliver one coherent function, which can be the Sprint Goal. The Sprint Goal can be any other coherence that causes the Development Team to work together rather than on separate initiatives.

As the Development Team works, it keeps the Sprint Goal in mind. In order to satisfy the Sprint Goal, it implements the functionality and technology. If the work turns out to be different than the Development Team expected, they collaborate with the Product Owner to negotiate the scope of Sprint Backlog within the Sprint.

Daily Scrum

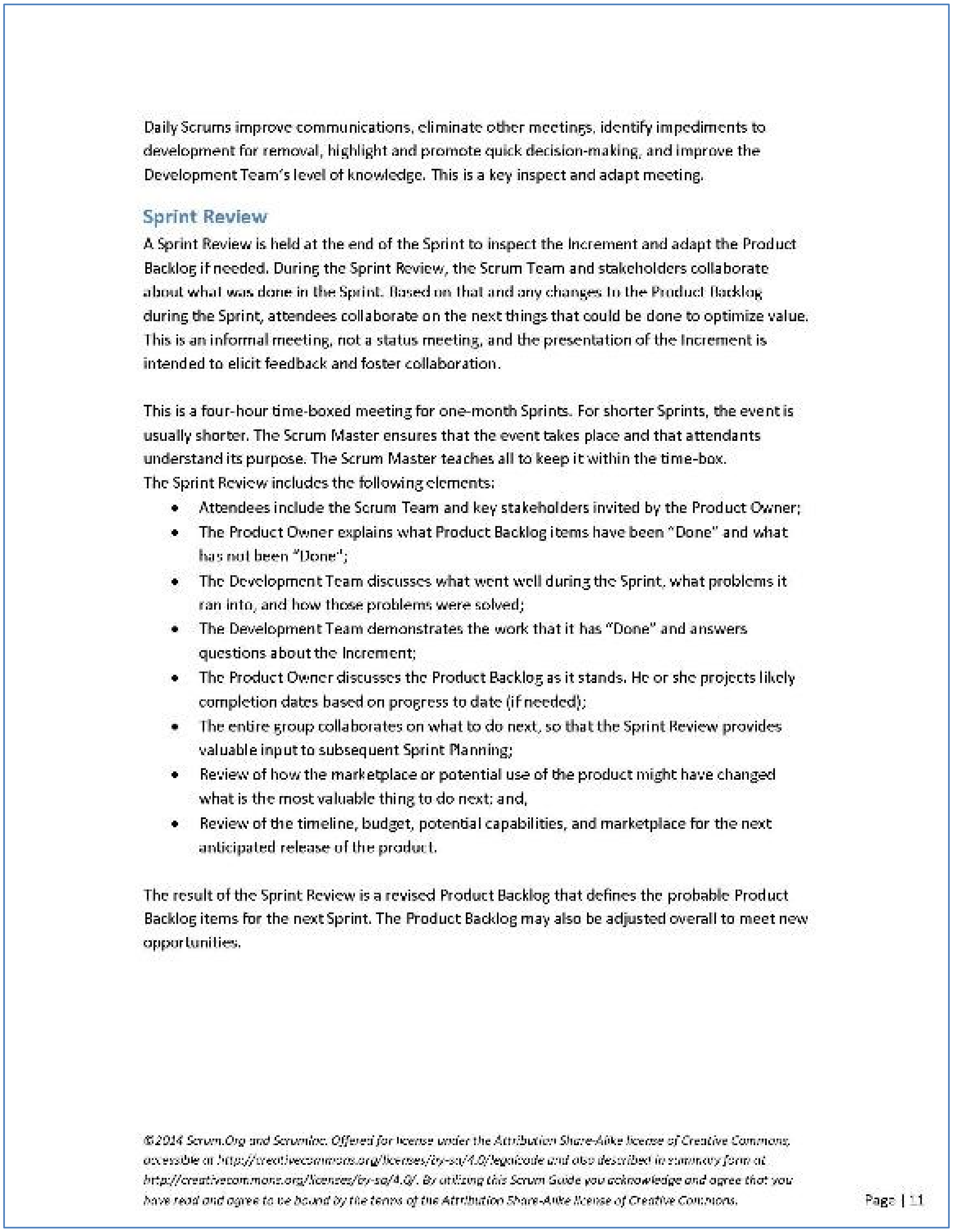
The Daily Scrum is a 15-minute time-boxed event for the Development Team to synchronize activities and create a plan for the next 24 hours. This is done by inspecting the work since the last Daily Scrum and forecasting the work that could be done before the next one. The Daily Scrum is held at the same time and place each day to reduce complexity. During the meeting, the Development Team members explain:

- What did I do yesterday that helped the Development Team meet the Sprint Goal?
- What will I do today to help the Development Team meet the Sprint Goal?
- Do I see any impediment that prevents me or the Development Team from meeting the Sprint Goal?

The Development Team uses the Daily Scrum to inspect progress toward the Sprint Goal and to inspect how progress is trending toward completing the work in the Sprint Backlog. The Daily Scrum optimizes the probability that the Development Team will meet the Sprint Goal. Every day, the Development Team should understand how it intends to work together as a self-organizing team to accomplish the Sprint Goal and create the anticipated Increment by the end of the Sprint. The Development Team or team members often meet immediately after the Daily Scrum for detailed discussions, or to adapt, or replan, the rest of the Sprint's work.

The Scrum Master ensures that the Development Team has the meeting, but the Development Team is responsible for conducting the Daily Scrum. The Scrum Master teaches the Development Team to keep the Daily Scrum within the 15-minute time-box.

The Scrum Master enforces the rule that only Development Team members participate in the Daily Scrum.



Daily Scrums improve communications, eliminate other meetings, identify impediments to development for removal, highlight and promote quick decision-making, and improve the Development Team's level of knowledge. This is a key inspect and adapt meeting.

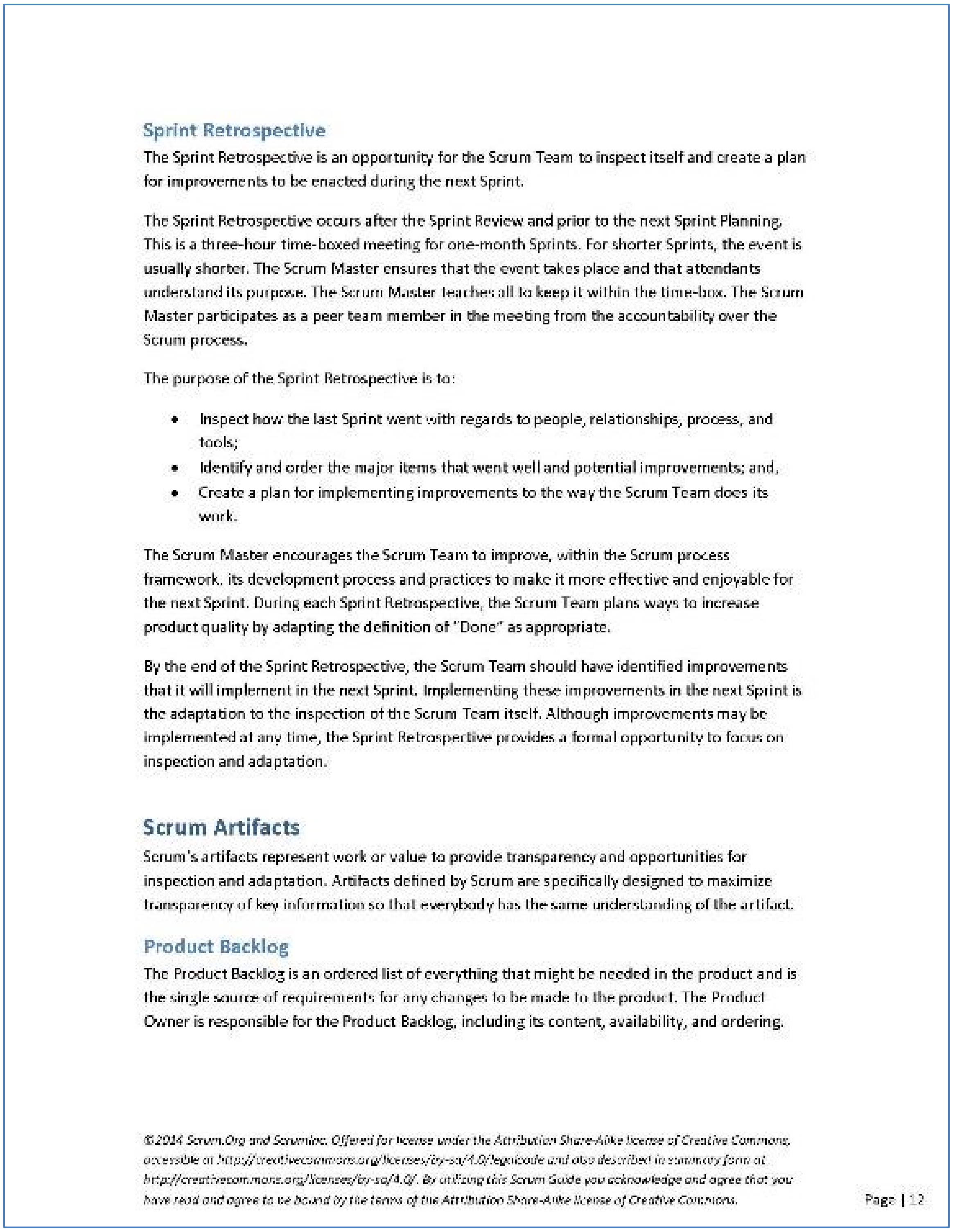
Sprint Review

A Sprint Review is held at the end of the Sprint to inspect the Increment and adapt the Product Backlog if needed. During the Sprint Review, the Scrum Team and stakeholders collaborate about what was done in the Sprint. Based on that and any changes to the Product Backlog during the Sprint, attendees collaborate on the next things that could be done to optimize value. This is an informal meeting, not a status meeting, and the presentation of the Increment is intended to elicit feedback and foster collaboration.

This is a four-hour time-boxed meeting for one-month Sprints. For shorter Sprints, the event is usually shorter. The Scrum Master ensures that the event takes place and that attendants understand its purpose. The Scrum Master teaches all to keep it within the time-box.
The Sprint Review includes the following elements:

- Attendees include the Scrum Team and key stakeholders invited by the Product Owner;
- The Product Owner explains what Product Backlog items have been "Done" and what has not been "Done";
- The Development Team discusses what went well during the Sprint, what problems it ran into, and how those problems were solved;
- The Development Team demonstrates the work that it has "Done" and answers questions about the Increment;
- The Product Owner discusses the Product Backlog as it stands. He or she projects likely completion dates based on progress to date (if needed);
- The entire group collaborates on what to do next, so that the Sprint Review provides valuable input to subsequent Sprint Planning;
- Review of how the marketplace or potential use of the product might have changed what is the most valuable thing to do next; and,
- Review of the timeline, budget, potential capabilities, and marketplace for the next anticipated release of the product.

The result of the Sprint Review is a revised Product Backlog that defines the probable Product Backlog items for the next Sprint. The Product Backlog may also be adjusted overall to meet new opportunities.



Sprint Retrospective

The Sprint Retrospective is an opportunity for the Scrum Team to inspect itself and create a plan for improvements to be enacted during the next Sprint.

The Sprint Retrospective occurs after the Sprint Review and prior to the next Sprint Planning. This is a three-hour time-boxed meeting for one-month Sprints. For shorter Sprints, the event is usually shorter. The Scrum Master ensures that the event takes place and that attendants understand its purpose. The Scrum Master teaches all to keep it within the time-box. The Scrum Master participates as a peer team member in the meeting from the accountability over the Scrum process.

The purpose of the Sprint Retrospective is to:

- Inspect how the last Sprint went with regards to people, relationships, process, and tools;
- Identify and order the major items that went well and potential improvements; and,
- Create a plan for implementing improvements to the way the Scrum Team does its work.

The Scrum Master encourages the Scrum Team to improve, within the Scrum process framework, its development process and practices to make it more effective and enjoyable for the next Sprint. During each Sprint Retrospective, the Scrum Team plans ways to increase product quality by adapting the definition of "Done" as appropriate.

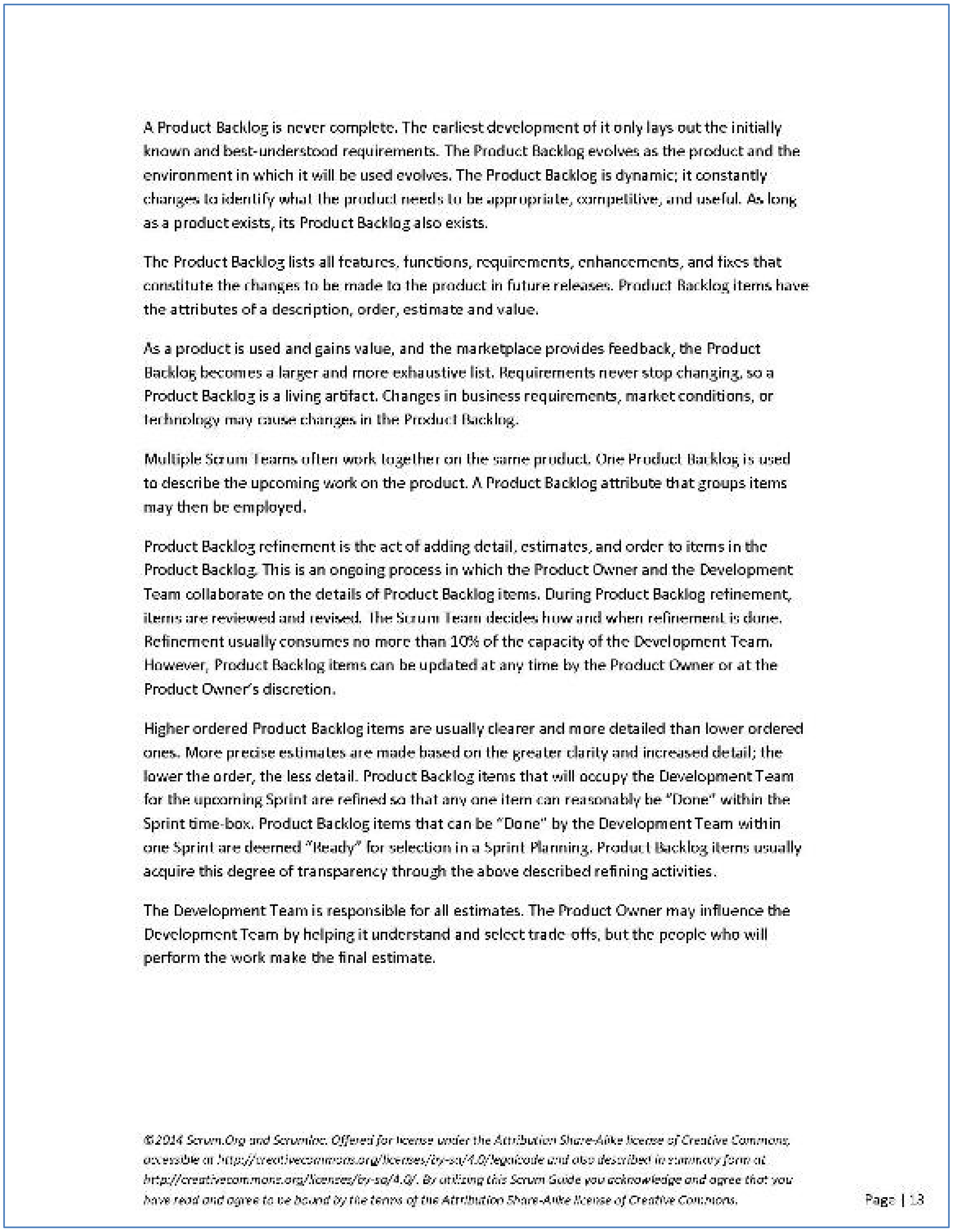
By the end of the Sprint Retrospective, the Scrum Team should have identified improvements that it will implement in the next Sprint. Implementing these improvements in the next Sprint is the adaptation to the inspection of the Scrum Team itself. Although improvements may be implemented at any time, the Sprint Retrospective provides a formal opportunity to focus on inspection and adaptation.

Scrum Artifacts

Scrum's artifacts represent work or value to provide transparency and opportunities for inspection and adaptation. Artifacts defined by Scrum are specifically designed to maximize transparency of key information so that everybody has the same understanding of the artifact.

Product Backlog

The Product Backlog is an ordered list of everything that might be needed in the product and is the single source of requirements for any changes to be made to the product. The Product Owner is responsible for the Product Backlog, including its content, availability, and ordering.



A Product Backlog is never complete. The earliest development of it only lays out the initially known and best-understood requirements. The Product Backlog evolves as the product and the environment in which it will be used evolves. The Product Backlog is dynamic; it constantly changes to identify what the product needs to be appropriate, competitive, and useful. As long as a product exists, its Product Backlog also exists.

The Product Backlog lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases. Product Backlog items have the attributes of a description, order, estimate and value.

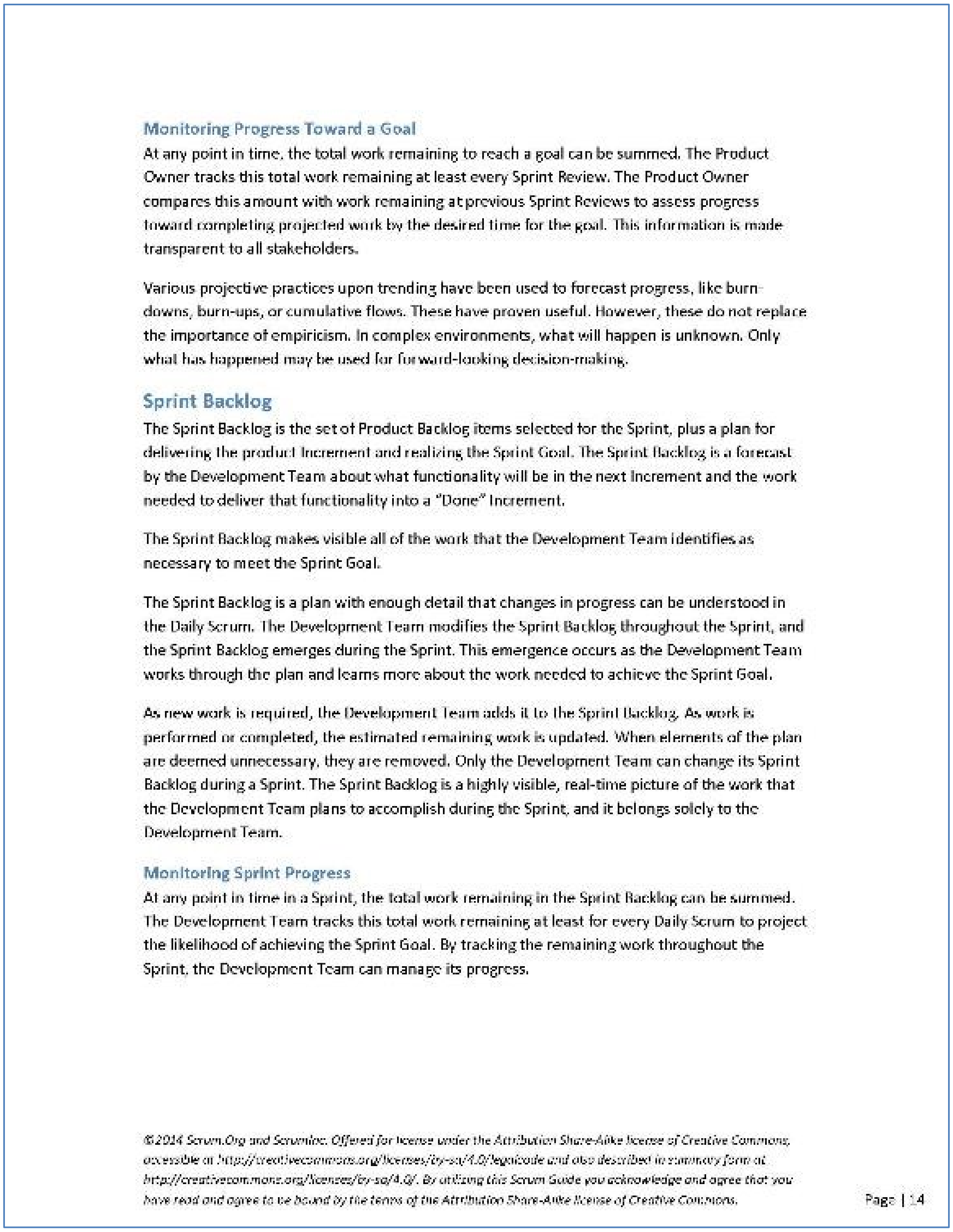
As a product is used and gains value, and the marketplace provides feedback, the Product Backlog becomes a larger and more exhaustive list. Requirements never stop changing, so a Product Backlog is a living artifact. Changes in business requirements, market conditions, or technology may cause changes in the Product Backlog.

Multiple Scrum Teams often work together on the same product. One Product Backlog is used to describe the upcoming work on the product. A Product Backlog attribute that groups items may then be employed.

Product Backlog refinement is the act of adding detail, estimates, and order to items in the Product Backlog. This is an ongoing process in which the Product Owner and the Development Team collaborate on the details of Product Backlog items. During Product Backlog refinement, items are reviewed and revised. The Scrum Team decides how and when refinement is done. Refinement usually consumes no more than 10% of the capacity of the Development Team. However, Product Backlog items can be updated at any time by the Product Owner or at the Product Owner's discretion.

Higher ordered Product Backlog items are usually clearer and more detailed than lower ordered ones. More precise estimates are made based on the greater clarity and increased detail; the lower the order, the less detail. Product Backlog items that will occupy the Development Team for the upcoming Sprint are refined so that any one item can reasonably be "Done" within the Sprint time-box. Product Backlog items that can be "Done" by the Development Team within one Sprint are deemed "Ready" for selection in a Sprint Planning. Product Backlog items usually acquire this degree of transparency through the above described refining activities.

The Development Team is responsible for all estimates. The Product Owner may influence the Development Team by helping it understand and select trade-offs, but the people who will perform the work make the final estimate.



Monitoring Progress Toward a Goal

At any point in time, the total work remaining to reach a goal can be summed. The Product Owner tracks this total work remaining at least every Sprint Review. The Product Owner compares this amount with work remaining at previous Sprint Reviews to assess progress toward completing projected work by the desired time for the goal. This information is made transparent to all stakeholders.

Various projective practices upon trending have been used to forecast progress, like burn-downs, burn-ups, or cumulative flows. These have proven useful. However, these do not replace the importance of empiricism. In complex environments, what will happen is unknown. Only what has happened may be used for forward-looking decision-making.

Sprint Backlog

The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal. The Sprint Backlog is a forecast by the Development Team about what functionality will be in the next Increment and the work needed to deliver that functionality into a "Done" Increment.

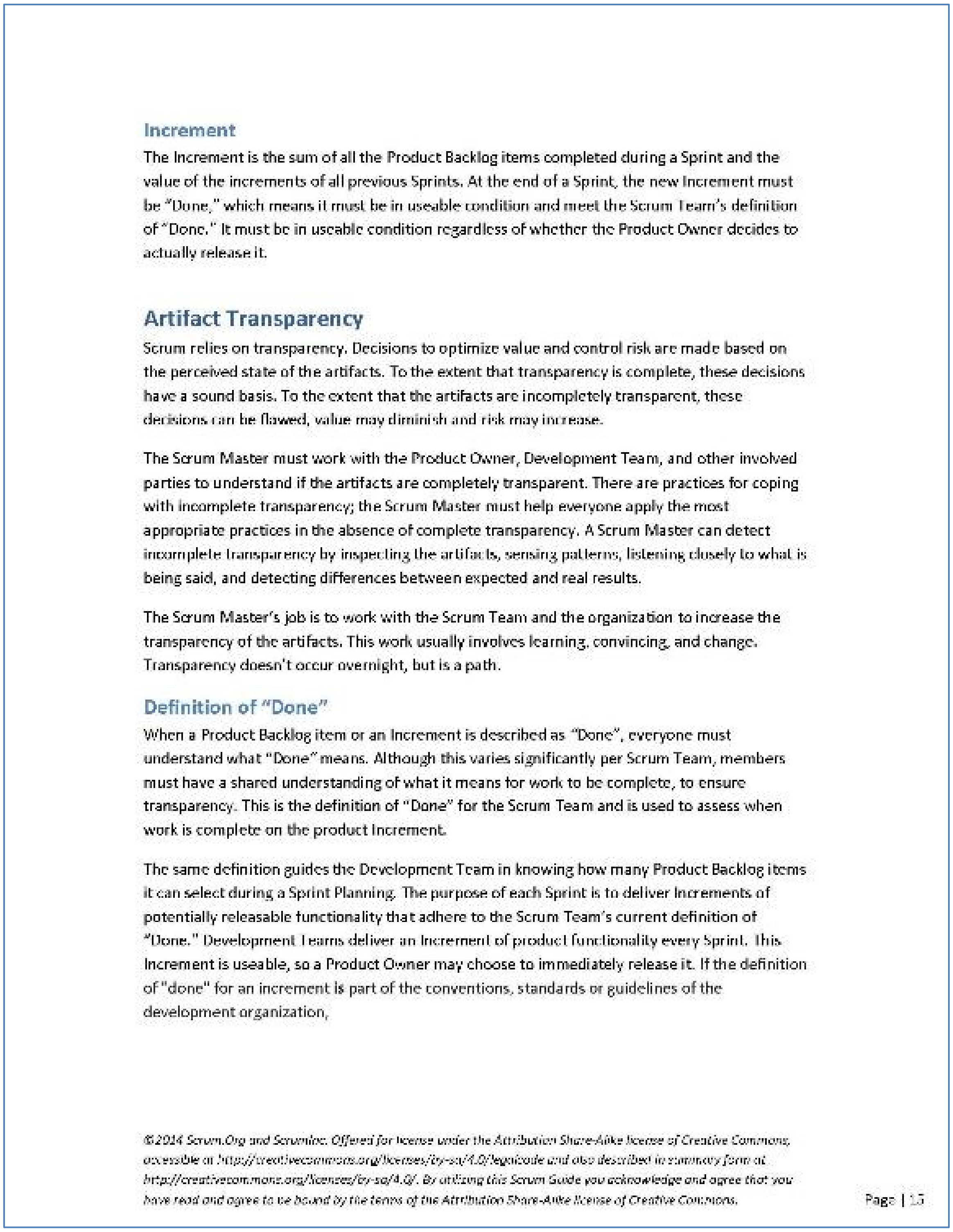
The Sprint Backlog makes visible all of the work that the Development Team identifies as necessary to meet the Sprint Goal.

The Sprint Backlog is a plan with enough detail that changes in progress can be understood in the Daily Scrum. The Development Team modifies the Sprint Backlog throughout the Sprint, and the Sprint Backlog emerges during the Sprint. This emergence occurs as the Development Team works through the plan and learns more about the work needed to achieve the Sprint Goal.

As new work is required, the Development Team adds it to the Sprint Backlog. As work is performed or completed, the estimated remaining work is updated. When elements of the plan are deemed unnecessary, they are removed. Only the Development Team can change its Sprint Backlog during a Sprint. The Sprint Backlog is a highly visible, real-time picture of the work that the Development Team plans to accomplish during the Sprint, and it belongs solely to the Development Team.

Monitoring Sprint Progress

At any point in time in a Sprint, the total work remaining in the Sprint Backlog can be summed. The Development Team tracks this total work remaining at least for every Daily Scrum to project the likelihood of achieving the Sprint Goal. By tracking the remaining work throughout the Sprint, the Development Team can manage its progress.



Increment

The Increment is the sum of all the Product Backlog items completed during a Sprint and the value of the increments of all previous Sprints. At the end of a Sprint, the new Increment must be "Done," which means it must be in useable condition and meet the Scrum Team's definition of "Done." It must be in useable condition regardless of whether the Product Owner decides to actually release it.

Artifact Transparency

Scrum relies on transparency. Decisions to optimize value and control risk are made based on the perceived state of the artifacts. To the extent that transparency is complete, these decisions have a sound basis. To the extent that the artifacts are incompletely transparent, these decisions can be flawed, value may diminish and risk may increase.

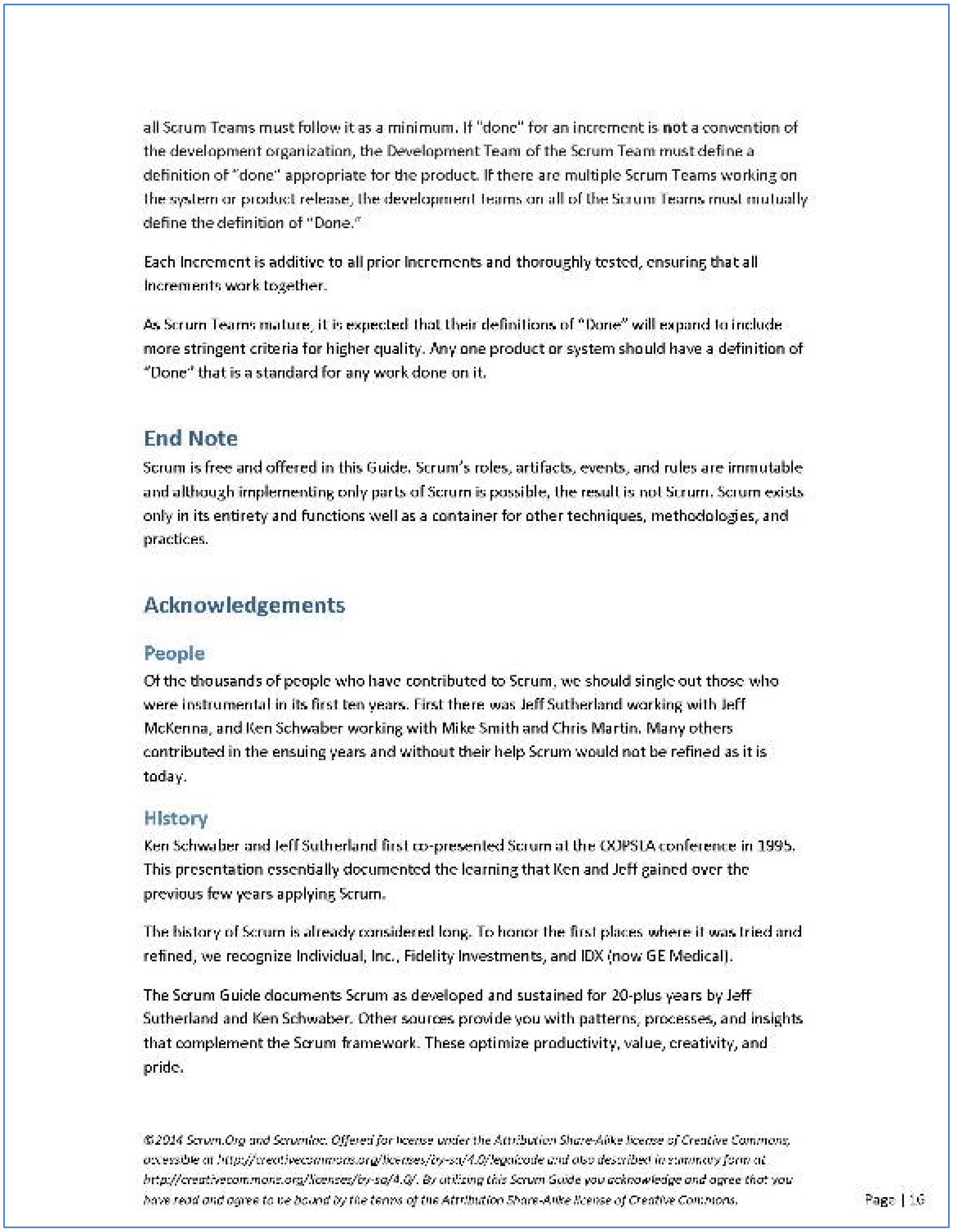
The Scrum Master must work with the Product Owner, Development Team, and other involved parties to understand if the artifacts are completely transparent. There are practices for coping with incomplete transparency; the Scrum Master must help everyone apply the most appropriate practices in the absence of complete transparency. A Scrum Master can detect incomplete transparency by inspecting the artifacts, sensing patterns, listening closely to what is being said, and detecting differences between expected and real results.

The Scrum Master's job is to work with the Scrum Team and the organization to increase the transparency of the artifacts. This work usually involves learning, convincing, and change. Transparency doesn't occur overnight, but is a path.

Definition of "Done"

When a Product Backlog item or an Increment is described as "Done", everyone must understand what "Done" means. Although this varies significantly per Scrum Team, members must have a shared understanding of what it means for work to be complete, to ensure transparency. This is the definition of "Done" for the Scrum Team and is used to assess when work is complete on the product Increment.

The same definition guides the Development Team in knowing how many Product Backlog items it can select during a Sprint Planning. The purpose of each Sprint is to deliver Increments of potentially releasable functionality that adhere to the Scrum Team's current definition of "Done." Development Teams deliver an Increment of product functionality every Sprint. This Increment is useable, so a Product Owner may choose to immediately release it. If the definition of "done" for an increment is part of the conventions, standards or guidelines of the development organization,



all Scrum Teams must follow it as a minimum. If "done" for an increment is not a convention of the development organization, the Development Team of the Scrum Team must define a definition of "done" appropriate for the product. If there are multiple Scrum Teams working on the system or product release, the development teams on all of the Scrum Teams must mutually define the definition of "Done."

Each Increment is additive to all prior Increments and thoroughly tested, ensuring that all Increments work together.

As Scrum Teams mature, it is expected that their definitions of "Done" will expand to include more stringent criteria for higher quality. Any one product or system should have a definition of "Done" that is a standard for any work done on it.

End Note

Scrum is free and offered in this Guide. Scrum's roles, artifacts, events, and rules are immutable and although implementing only parts of Scrum is possible, the result is not Scrum. Scrum exists only in its entirety and functions well as a container for other techniques, methodologies, and practices.

Acknowledgements

People

Of the thousands of people who have contributed to Scrum, we should single out those who were instrumental in its first ten years. First there was Jeff Sutherland working with Jeff McKenna, and Ken Schwaber working with Mike Smith and Chris Martin. Many others contributed in the ensuing years and without their help Scrum would not be refined as it is today.

History

Ken Schwaber and Jeff Sutherland first co-presented Scrum at the OOPSLA conference in 1995. This presentation essentially documented the learning that Ken and Jeff gained over the previous few years applying Scrum.

The history of Scrum is already considered long. To honor the first places where it was tried and refined, we recognize Individual, Inc., Fidelity Investments, and IDX (now GE Medical).

The Scrum Guide documents Scrum as developed and sustained for 20-plus years by Jeff Sutherland and Ken Schwaber. Other sources provide you with patterns, processes, and insights that complement the Scrum framework. These optimize productivity, value, creativity, and pride.

APPENDIX C

CONCEPTUAL FRAMEWORK

My conceptual framework includes the sensitizing concept that requirements are *constructed* rather than *gathered.* Another concept is that that information systems do not have requirements, but rather stakeholders have requirements in the form of expectations. Requirements for information systems are subjective expectations based on each stakeholder's perspective, knowledge, experience, and understanding. With multiple stakeholders, the likelihood for conflicting expectations is high and so a collaborative process involving both constructivism and social construction of knowledge is required to reconcile the conflicts and generate a common understanding among all stakeholders. The result of successful, constructive collaboration among stakeholders is a set of constructed requirements.

Missing from this framework are the specific challenges that cause gaps in understanding among stakeholders for information. In other words, what challenges are encountered by stakeholders when identifying requirements for information systems? That is the question addressed by this study.

APPENDIX D

INITIAL INTERVIEW GUIDE

Each participant will be separately given the below three prompts to think of incidents. Ideally, each participant will provide three separate incidents, but some may only be able to provide one or two incidents. The same questions will be asked about each incident, with additional probing questions asked when determined appropriate by the interviewer.

In the unlikely event where a participant does not recall an incident where the information system did not meet requirements, use an incident when requirements were met but with challenges encountered during the requirements process.

Prompt #1: Think of the **most recent** time when you were involved with an information system that was either purchased or developed and ***did not**** meet the requirements.

(wait for participant to recall an incident)

Questions: *(same questions will be asked after all 3 prompts to each participant)*

1. Tell me about the project. *(Validates the participant has an appropriate incident in mind, and also provides background that may be further examined later.)*

2. What factors led to the requirements not being met?

3. What could have been done differently to improve the likelihood of success?

4. What was your role on the project? *(Clarifies the participant's perspective)*

5. How did the project turn out?

Prompt #2: Think of another time when you were involved with an information system that was either purchased or developed and ***did not**** meet the requirements and had the **most significant impact** on the organization.

(use same questions above)

Prompt #3: Think of another time when you were involved with an information system that was either purchased or developed and ***did not**** meet the requirements and generated your feeling of "here we go again" during the requirements phase. *(Targeting repeated behaviors)*

(use same questions above)

Extra question: What steps were taken to avoid repeating mistakes?

APPENDIX E

FINAL INTERVIEW GUIDE

Each participant will be separately given the below three prompts to think of incidents. Ideally, each participant will provide three separate incidents, but some may only be able to provide one or two incidents. The same questions will be asked about each incident, with additional probing questions asked when determined appropriate by the interviewer.

In the unlikely event where a participant does not recall an incident where the information system did not meet requirements, use an incident when requirements were met but with challenges encountered during the requirements process.

Introduction: Over the next hour I will be asking you to recall 2 or 3 projects from your career. I'll first ask you about a recent project and then later ask you about projects from earlier. How many years have you been working with information systems?

Prompt #1: Think of the **most recent** time when you were involved with an information system that was either purchased or developed and ***did not**** meet the requirements.

(wait for participant to recall an incident)

Questions: *(same questions will be asked after all 3 prompts to each participant)*

1. Tell me about the project. *(Validates the participant has an appropriate incident in mind, and also provides background that may be further examined later.)*
2. What factors led to the requirements not being met?
3. What could have been done differently to improve the likelihood of success?
4. What was your role on the project? *(Clarifies the participant's perspective)*
5. How did the project turn out? *(If needed, may have already been answered)*

Prompt #2: Think of another time when you were involved with an information system that was either purchased or developed and ***did not**** meet the requirements and had the **most significant impact** on the organization.

(use same questions above)

Prompt #3: Think of another time when you were involved with an information system that was either purchased or developed and ***did not**** meet the requirements and generated your feeling of "here we go again" during the requirements phase. *(Targeting repeated behaviors)*

(use same questions above)

Extra question: What steps were taken to avoid repeating behaviors? *(if needed, may have already been answered)*

APPENDIX F

INFORMED CONSENT

David L. Gibbs, a Ph.D. student in the Adult, Professional, and Community Education program at Texas State University-San Marcos, is conducting a research study entitled *Constructing Requirements: A qualitative study of the challenges associated with eliciting requirements for information systems.* This research project will be used to complete the dissertation requirement for the Ph.D. degree. David may be contacted at (512) 698-8707 or dgibbs@txstate.edu. The Dissertation Committee includes: Dr. Ann Brooks, Chair, Dr. Joellen Coryell, Dr. Steve Furney, and Dr. Tiankai Wang. All committee members are faculty at Texas State University and may be reached through the Education Department at (512) 245-8084.

The purpose of this study is to identify and document the challenges experienced by practitioners eliciting requirements for information systems projects. Why is it so difficult to communicate system requirements? Once these challenges are identified and better understood, future research and practice may help address the challenges and improve the success rate of information systems projects.

You are asked to participate in an interview for this research study because you have valuable experience as a practitioner working with requirements on large, complex, information systems projects. Participants in this study are either known to David as having at least five years of relevant experience or have been recommended by other practitioners.

Your participation will be to recall up to three projects where you were involved with information systems requirements and to answer a few questions about challenges you experienced on each project. The interview is expected to last approximately 60 minutes.

The interview will be digitally recorded in order to ensure accuracy. Your identity will be kept confidential at all times and will never be published. David will personally transcribe the audio recordings and one research committee member will validate a sampling of the recordings to ensure appropriate processes are followed. Only the original and one backup copy of the recordings will exist. In all writings, your responses will be associated with a pseudonym and no personally identifiable information will be revealed. David will keep the only list matching participant names and pseudonyms for the purposes of correct identification should follow up or clarification be necessary. Once data collection has been finalized, this list will be destroyed so no written link between you and the data will remain. Data associated with this study will be locked in David's office or a safe deposit box when not in use and will only be shared or discussed with members of the dissertation committee until publication. Data will be kept for four years and then destroyed.

No compensation is offered for your participation. However, your participation in this study may contribute to research and practices that will improve the success of future information systems projects.

Participation is voluntary and you may withdraw from the interview at any time or refuse to answer any question for any reason without penalty, prejudice or jeopardy. Final results from the study will be sent in an electronic format to all participants, while maintaining confidentiality.

Any questions about this research project or your rights as a participant should be directed to the IRB Chair, Dr. Jon Lasser at (512) 245-3413 or lasser@txstate.edu or to Ms. Becky Northcut, Compliance Specialist at (512) 245-2102 or bnorthcut@txstate.edu.

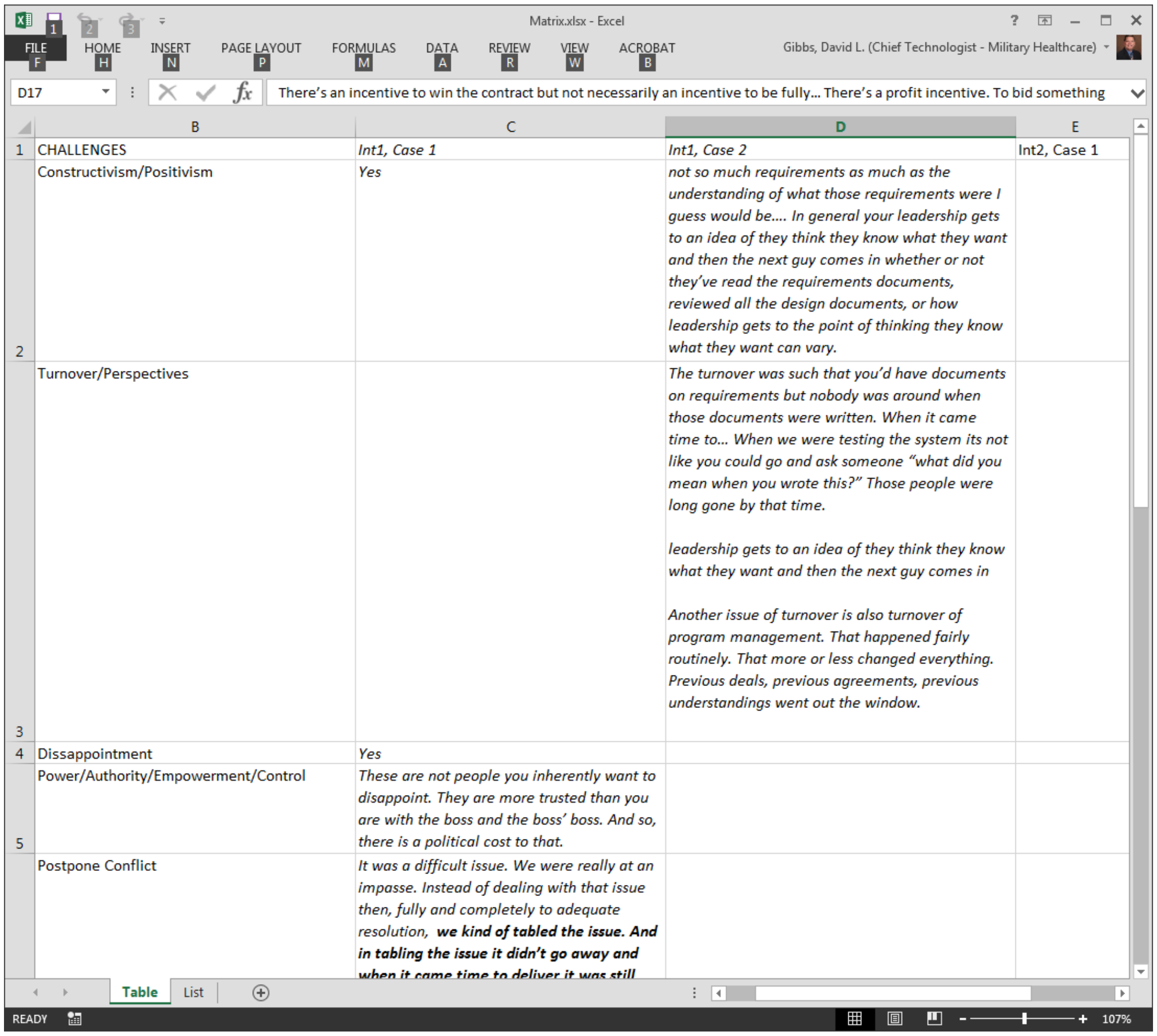
IRB Approval number: Application #EXP2012Z6144 granted IRB exempt status 12/03/2012)

________________________________	________________________________
Participant Signature & Date	Researcher Signature & Date

APPENDIX G

EXAMPLE MEMOS

Memo used to analyze data and initial codes


Matrix.xlsx - Excel

FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW ACROBAT

Gibbs, David L. (Chief Technologist - Military Healthcare)

D17

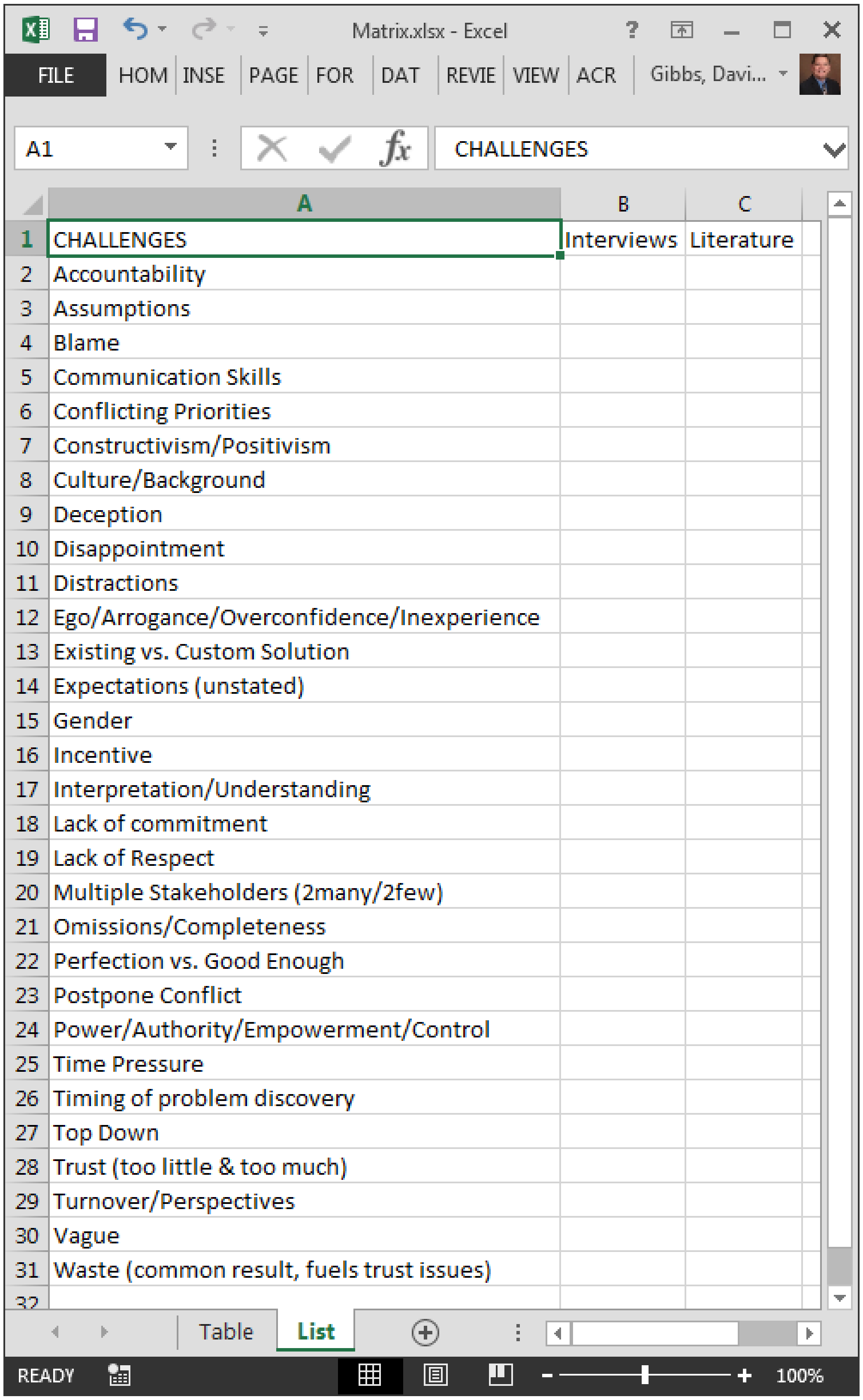
There's an incentive to win the contract but not necessarily an incentive to be fully... There's a profit incentive. To bid something

	B	C	D	E
1	CHALLENGES	*Int1, Case 1*	*Int1, Case 2*	Int2, Case 1
2	Constructivism/Positivism	*Yes*	*not so much requirements as much as the understanding of what those requirements were I guess would be.... In general your leadership gets to an idea of they think they know what they want and then the next guy comes in whether or not they've read the requirements documents, reviewed all the design documents, or how leadership gets to the point of thinking they know what they want can vary.*	
3	Turnover/Perspectives		*The turnover was such that you'd have documents on requirements but nobody was around when those documents were written. When it came time to... When we were testing the system its not like you could go and ask someone "what did you mean when you wrote this?" Those people were long gone by that time.* *leadership gets to an idea of they think they know what they want and then the next guy comes in* *Another issue of turnover is also turnover of program management. That happened fairly routinely. That more or less changed everything. Previous deals, previous agreements, previous understandings went out the window.*	
4	Dissappointment	*Yes*		
5	Power/Authority/Empowerment/Control	*These are not people you inherently want to disappoint. They are more trusted than you are with the boss and the boss' boss. And so, there is a political cost to that.*		
	Postpone Conflict	*It was a difficult issue. We were really at an impasse. Instead of dealing with that issue then, fully and completely to adequate resolution,* ***we kind of tabled the issue. And in tabling the issue it didn't go away and when it came time to deliver it was still***		

Table List

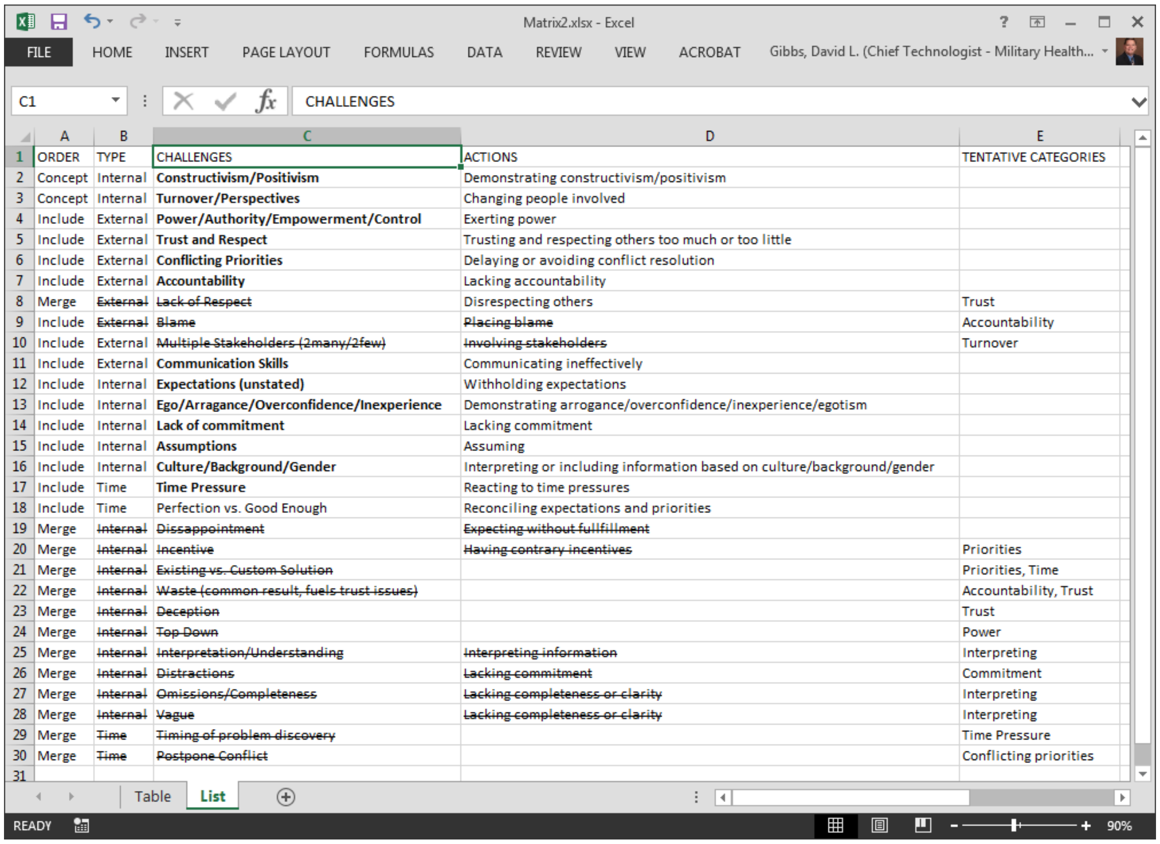
READY 107%

Memo used to analyze initial codes



	A	B	C
1	CHALLENGES	Interviews	Literature
2	Accountability		
3	Assumptions		
4	Blame		
5	Communication Skills		
6	Conflicting Priorities		
7	Constructivism/Positivism		
8	Culture/Background		
9	Deception		
10	Disappointment		
11	Distractions		
12	Ego/Arrogance/Overconfidence/Inexperience		
13	Existing vs. Custom Solution		
14	Expectations (unstated)		
15	Gender		
16	Incentive		
17	Interpretation/Understanding		
18	Lack of commitment		
19	Lack of Respect		
20	Multiple Stakeholders (2many/2few)		
21	Omissions/Completeness		
22	Perfection vs. Good Enough		
23	Postpone Conflict		
24	Power/Authority/Empowerment/Control		
25	Time Pressure		
26	Timing of problem discovery		
27	Top Down		
28	Trust (too little & too much)		
29	Turnover/Perspectives		
30	Vague		
31	Waste (common result, fuels trust issues)		
32			

Memo used to analyze codes and tentative categories



ORDER	TYPE	CHALLENGES	ACTIONS	TENTATIVE CATEGORIES
Concept	Internal	**Constructivism/Positivism**	Demonstrating constructivism/positivism	
Concept	Internal	**Turnover/Perspectives**	Changing people involved	
Include	External	**Power/Authority/Empowerment/Control**	Exerting power	
Include	External	**Trust and Respect**	Trusting and respecting others too much or too little	
Include	External	**Conflicting Priorities**	Delaying or avoiding conflict resolution	
Include	External	**Accountability**	Lacking accountability	
Merge	~~External~~	~~Lack of Respect~~	Disrespecting others	Trust
Include	~~External~~	~~Blame~~	~~Placing blame~~	Accountability
Include	External	~~Multiple Stakeholders (2many/2few)~~	~~Involving stakeholders~~	Turnover
Include	External	**Communication Skills**	Communicating ineffectively	
Include	Internal	**Expectations (unstated)**	Withholding expectations	
Include	Internal	**Ego/Arragance/Overconfidence/Inexperience**	Demonstrating arrogance/overconfidence/inexperience/egotism	
Include	Internal	**Lack of commitment**	Lacking commitment	
Include	Internal	**Assumptions**	Assuming	
Include	Internal	**Culture/Background/Gender**	Interpreting or including information based on culture/background/gender	
Include	Time	**Time Pressure**	Reacting to time pressures	
Include	Time	Perfection vs. Good Enough	Reconciling expectations and priorities	
Merge	~~Internal~~	~~Dissappointment~~	~~Expecting without fullfillment~~	
Merge	~~Internal~~	~~Incentive~~	~~Having contrary incentives~~	Priorities
Merge	~~Internal~~	~~Existing vs. Custom Solution~~		Priorities, Time
Merge	~~Internal~~	~~Waste (common result, fuels trust issues)~~		Accountability, Trust
Merge	~~Internal~~	~~Deception~~		Trust
Merge	~~Internal~~	~~Top Down~~		Power
Merge	~~Internal~~	~~Interpretation/Understanding~~	~~Interpreting information~~	Interpreting
Merge	~~Internal~~	~~Distractions~~	~~Lacking commitment~~	Commitment
Merge	~~Internal~~	~~Omissions/Completeness~~	~~Lacking completeness or clarity~~	Interpreting
Merge	~~Internal~~	~~Vague~~	~~Lacking completeness or clarity~~	Interpreting
Merge	~~Time~~	~~Timing of problem discovery~~		Time Pressure
Merge	~~Time~~	~~Postpone Conflict~~		Conflicting priorities

Memo in the form of an early abstract for possible presentation:

Session Title
Constructing Requirements: Evolving beyond hunting and gathering to improve HIT project outcomes

Session Description (400 characters or less):
This session uses vignettes from real-world projects to identify common challenges encountered when eliciting requirements for information systems and suggests a constructive, collaborative approach to improve project outcomes. The information presented applies regardless of whether agile, waterfall, or another lifecycle model is followed.

Session Abstract (1500 characters or less):
Contrary to common expression, requirements for successful information systems are not *gathered*, they are *learned*. More specifically, requirements are *constructed* knowledge formed through active collaboration among stakeholders. Using vignettes from real-world projects, this session identifies common challenges encountered when eliciting information systems requirements and explains why evolving from gathering to constructing requirements will improve success. Recommendations are made for how to address the common challenges through collaboration.

Data were collected and analyzed using qualitative methods revealing patterns. Some common challenges that emerged include perpetuating assumptions, avoiding conflict, misplacing trust, misusing power, unrealistic time pressure, personnel turnover, lacking commitment, and ignoring differences in people. These are social challenges, not technology challenges, but their negative impacts on success are very real and can be mitigated with a change of approach.

A core recommendation is to approach requirements as knowledge to be actively constructed via collaboration, not passively gathered. Other recommendations include ensuring all stakeholders are represented during collaboration, are committed to success, and are empowered to fully engage. Power dynamics, egos, and cultural differences must be proactively mitigated to achieve common understanding. It is also recommended to verify requirements when stakeholder turnover occurs.

Learning Objectives:
1. List common challenges faced when eliciting requirements for HIT solutions.
2. Recognize circumstances that indicate potential failure.
3. Contrast the positivist and constructivist approach to requirements.
4. Classify requirements approaches by reflecting on previous projects.

Speaker BIO (400 characters)
David Gibbs, MS, CPHIMS, has 30 years of professional experience with information systems requirements, the past decade in federal healthcare with Hewlett Packard. David is a doctoral candidate at Texas State University completing a Ph.D. in Adult, Professional and Community Education. His research involves addressing challenges encountered while eliciting requirements for information systems.

Session Background:
After decades of evolving processes, information systems projects continue to experience low success rates due to issues with requirements. There is debate whether the success rate of projects is around 30%, as reported in the annual survey from the Standish Group. Among practitioners, there is clear frustration about the challenges experienced, especially on large complex projects. Proponents of the agile model have demonstrated improvements to success rates as compared to the waterfall model, but even the improved 40% success rate remains disappointingly low.

Data were collected about real-world projects and analyzed using qualitative methods to identify emerging patterns of challenges related to requirements elicitation. The study covers both software development and system integration projects, including many related to health information technology (HIT). Sources of the collected data include semi-structured interviews with experienced practitioners as well as published case studies.

Common challenges that emerged from the study include perpetuating assumptions, avoiding conflict, misplacing trust, misusing power, exerting unrealistic time pressure, experiencing personnel turnover, lacking commitment, and ignoring differences in people. These are social challenges, not technology challenges, but their impacts are very real. The Standish Group report lists challenges that are more general yet related to those above: lack of input from users, incomplete requirements, and changing requirements.

This session will include recommendations for addressing the common challenges that, when ignored, have led projects to fail. A foundational recommendation is to approach requirements for information systems as knowledge to be proactively constructed via collaboration. Other recommendations are to ensure all current stakeholders are represented during the collaboration and that all are committed and empowered to fully engage. Power dynamics, egos, and cultural differences must be proactively mitigated to achieve full engagement from all stakeholders. It is also recommended to verify requirements anytime there is a turnover among the stakeholders.

The paragraphs below provide foundation for this session as well as example cases from the data collected. As a teaching tool and for some humor, I will relate the evolution of requirements approaches to the evolution of society from hunter/gatherer to the information age.

Contrary to common expression, requirements for successful information systems are not *gathered* or *discovered*, they are *learned*. More specifically, requirements are *constructed* knowledge formed through active collaboration of stakeholders. Too often, eliciting requirements is regarded as the administrivia before commencement of the real, challenging work. Such an attitude has doomed projects to fail. I suggest that constructing requirements is an integral part of the real work for any information system being developed or integrated.

Sociologist Gerhard Lenski's typology shows the evolution of humans from hunters and gatherers through agricultural and industrial societies. With each advancement, we improved our ability to influence the future outcomes of our endeavors. Now we are in the information age, yet we continue to *gather* requirements for information systems. We need to evolve by changing our approach to recognize requirements as knowledge that we must proactively construct.

Philosopher Auguste Compte developed the idea in the 1830s that a society engaged in the quest for knowledge, or truth, progresses through three stages: theological, metaphysical, and eventually the positive stage. *Positivism* promotes that rational, justifiable assertions can be verified through math, science, or logic. In other words, positivism holds that "the truth is out there" and can be gathered. Many information systems professionals approach requirements with a positivist attitude, and projects fail. I propose that requirements for information systems are not "out there", but instead originate in the minds of stakeholders as expectations.

Constructivism and *Social Constructionism* are more recently developed approaches to truth and knowledge. Promoted by developmental psychologist Jean Piaget in the 1960s, constructivism advocates that the physical world exists separately from our understanding of it and that knowledge about the world is constructed by each individual based on previous experiences. At about the same time, Peter Berger and Thomas Luckmann wrote that knowledge is constructed by social interactions, popularizing *social constructionism*. Together, these approaches can be applied to information systems projects to help drive the evolution from gathering requirements to constructing requirements. This simple change of attitude may have significant impact when combined with other contemporary methods such as requirements engineering and agile. In fact, constructivism helps explain why agile is effective.

Information system lifecycle models evolved from the waterfall model, through iterative models, to the agile model popular today. While these models prescribe requirements elicitation at different points in each respective lifecycle, the approach used to elicit the requirements remain discretional. Many practitioners continue to gather requirements with the same positivist approaches used for decades. Lists of requirements are often collected via email or by requesting stakeholders fill out forms. These positivist approaches have proven to yield limited success.

Constructing requirements adjusts the approach to acknowledge requirements do not exist on their own to be gathered, they must be constructed through proactive collaboration and reconciliation among the stakeholders. The requirements may indeed differ depending on which stakeholders participate, and often change when stakeholders change.

The waterfall model elicits requirements only at the beginning and does not accommodate change to requirements from learning. Frameworks that follow the agile model, such as Scrum, address requirements continuously. In Scrum parlance the requirements are represented in the Product Backlog which is constantly updated by the Product Owner as learning occurs. The Product Owner has responsibility for managing the requirements, but is not the source of the requirements.

The source of requirements for an information system is the set of people who are impacted by the outcome of the project, the stakeholders. Users, executive sponsors, funding sources, and even the team members responsible for delivering the solution are stakeholders. All stakeholders are potential sources of valuable knowledge of requirements and constraints which can influence whether a project is ultimately perceived as successful. Requirements are stakeholder expectations which must be understood and unified before they can be managed or engineered.

Achieving a common understanding of requirements, and reconciling conflicts, requires effort and a constructivist approach. In Scrum, the responsibility for promoting common understanding of requirements is assigned to the Product Owner. In other models, this task may be accomplished by project manager or a solution architect.

The presentation will include several vignettes from the data collected. Two representative samples are included below.

One example of misplaced trust occurred when a customer awarded a contract to a vendor who previously performed well in the developing custom software. Based on the trust established between the companies previously, little time was spent eliciting requirements for the new system. There was an attitude of "they already know what we need". Unfortunately, the individuals involved with the new project were not the same as with the first project. Many assumptions were made by both customer and vendor. Conflicts were delayed rather than being addressed immediately. Not surprisingly, the project fell short of expectations. The result was a missed deadline impacting customer operations, the vendor absorbing a substantial financial penalty, and a loss of trust between the companies. One way to address misplaced trust is to "trust but verify", meaning the customer and vendor agree on periodic checkpoints to validate understanding. In Scrum, this would be the Sprint Review.

Having diverse perspectives on a creative team is a strength and is desirable. However, ignoring diversity among people is a recurring challenge. People are not interchangeable machines with identical capabilities and knowledge. Each individual has not only skills, but also unique perspective, experiences, cultural background, understanding, priorities, and aspirations. One example where ignoring diversity was a challenge occurred on a project to develop software for an insurance company. Requirements were gathered using the terminology and concepts of insurance in the United States (U.S.). These requirements were provided in writing to expert developers whose backgrounds were strong in software, but not the jargon of the U.S. insurance industry. The developers were not invited to participate during requirements elicitation and so did not have the benefit of exposure to the context of the requirements. The developers thought they understood the meaning of the terms used to specify the requirements, but their interpretation was not always correct for the unfamiliar context of U.S. insurance. When questions arose, cultural differences hindered the development team's effectiveness to clarify meaning of the requirements. The outcome was software that met the wording of the requirements, but not the intended meaning. Customers were disappointed. Had this diversity been identified and addressed earlier the project might have been perceived as more successful. Referring to a separate case study, Leviss wrote "a culture that inhibits direct and open communication leads to diminished stakeholder input in favor of conflict avoidance behaviors that often defeat the intent of HIT systems." One way to address diversity is to include representatives of the developer team along with all other stakeholders as requirements are constructed and to ensure everyone feels empowered to speak up and ask for clarification and address issues. Another recommendation is to use technology such as instant messaging and virtual rooms to enable collaboration among stakeholders in order to close gaps in common understanding.

REFERENCES

Cardinal, M. (2014). *Executable specifications with Scrum: A practical guide to agile requirements discovery*. Boston: Addison-Wesley.

Leviss, J. (2010). *H.I.T. or Miss.* Chicago: American Health Information Management Association.

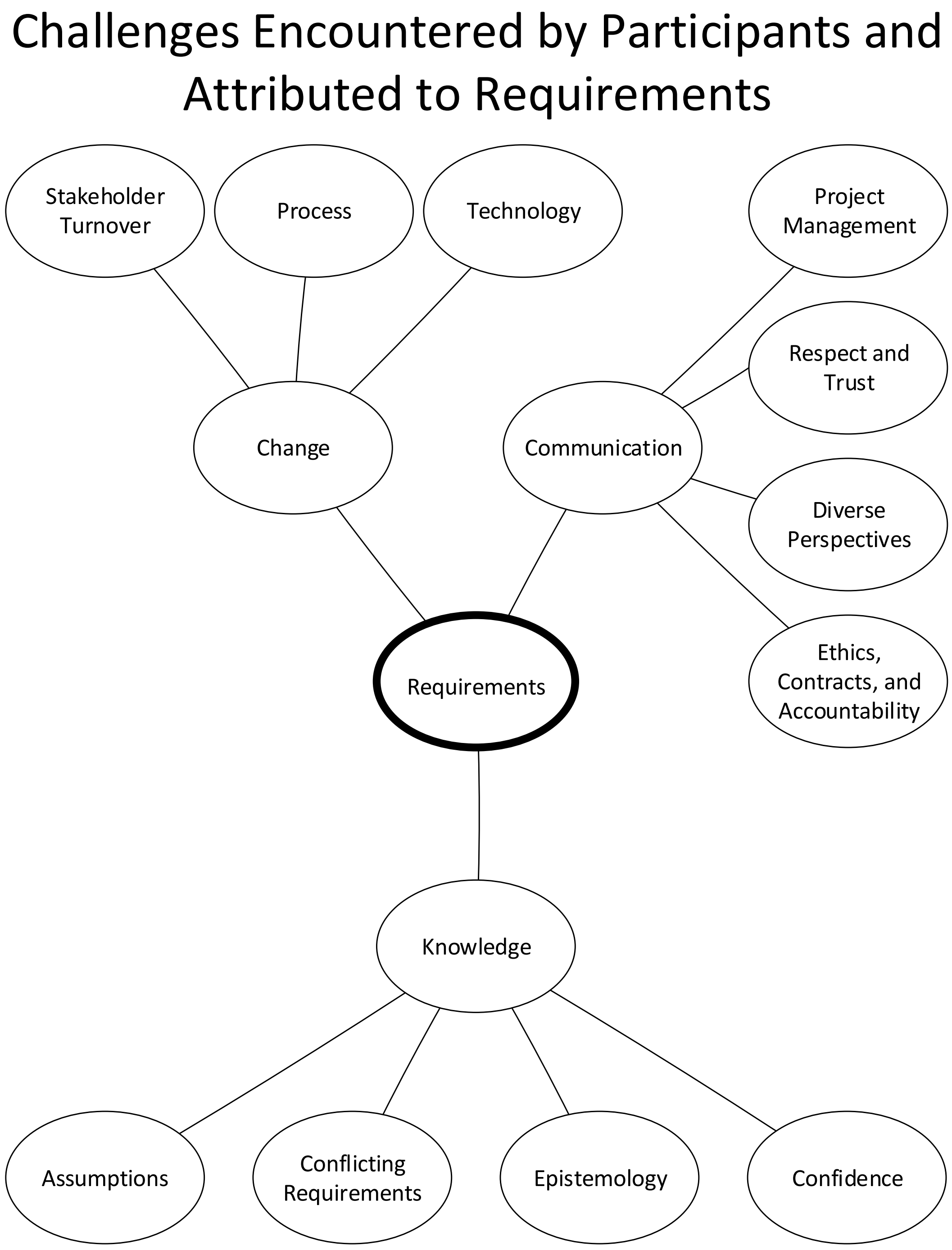
Leffingwell, D. (2011). *Agile software requirements: Lean requirements practices for teams, programs, and the enterprise.* Boston: Pearson.

Schwaber, K., & Sutherland, J. (2013). The scrum guide. *Scrum.org.*

The Standish Group. (2013). *The CHAOS manifesto*.

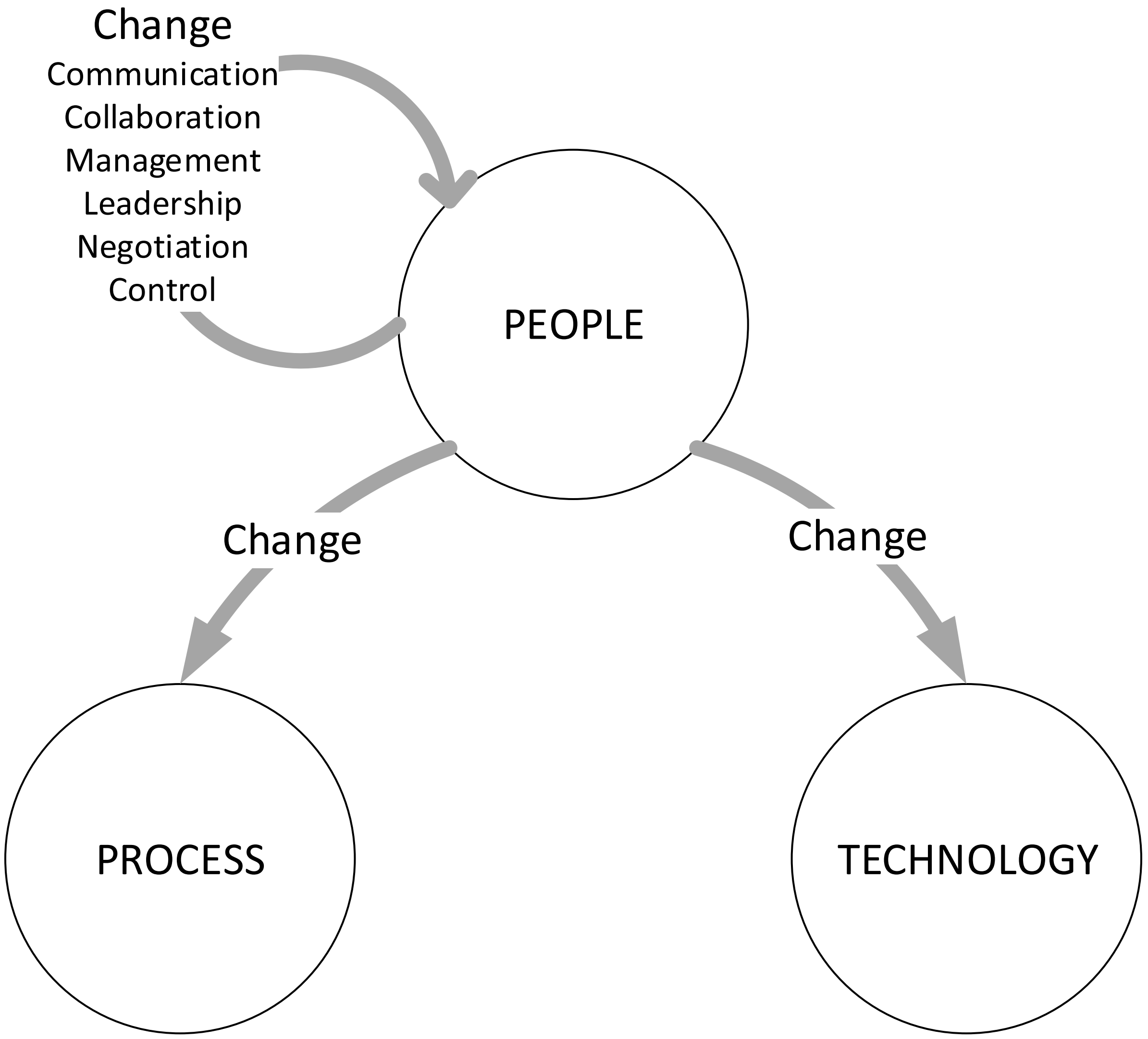
APPENDIX H

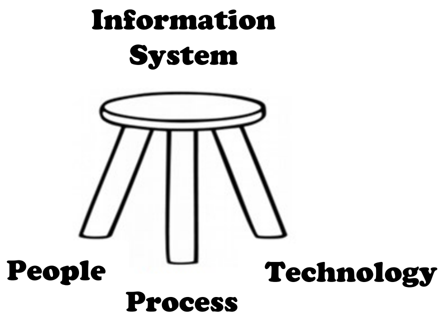
RELATIONSHIP DIAGRAM



APPENDIX I

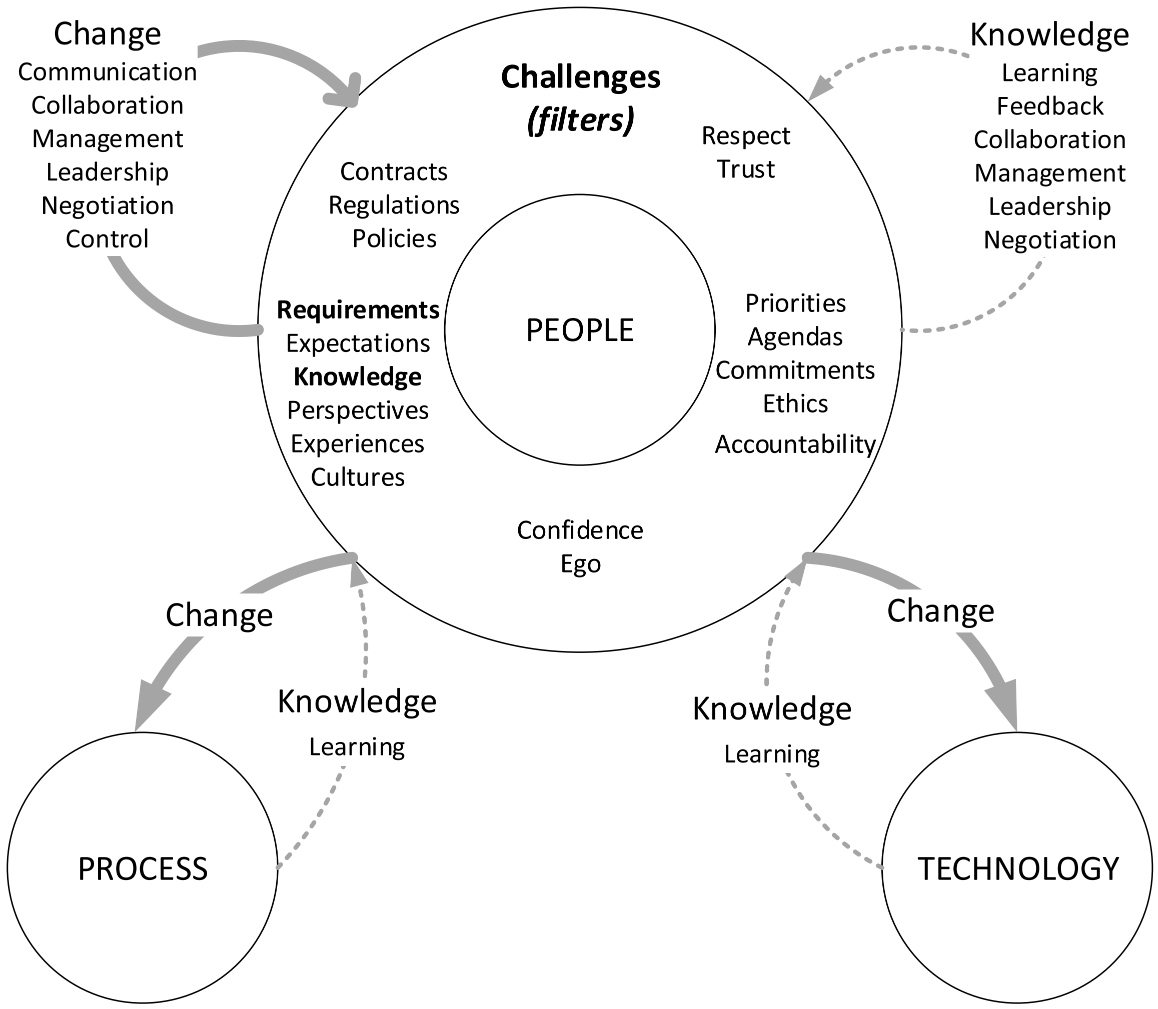
TRADITIONAL MODEL

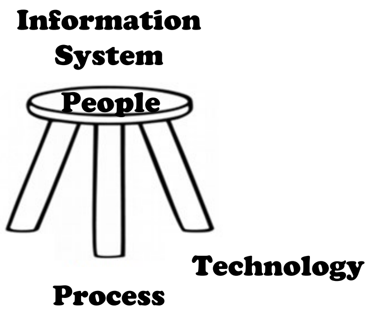




APPENDIX J

CONSTRUCTIVIST MODEL





REFERENCES

Agile Alliance. (2001). *Agile Manifesto*. Retrieved from http://agilealliance.org/the-alliance/the-agile-manifesto

American Psychological Association. (2014, May 4). APA Style Quick Answers. Retrieved from http://www.apastyle.org/learn/quick-guide-on-references.aspx

Berg, M. (2001). Implementing information systems in health care organizations: myths and challenges. *International Journal of Medical Informatics, 64*, 143-156.

Berger, P., & Luckmann, T. (1967). *The social construction of reality: A treatise in the sociology of knowledge.* New York: Anchor.

Bloomberg, L. D., & Volpe, M. (2008). *Completing your qualitative dissertation: A roadmap from beginning to end.* Thousand Oaks, CA: SAGE Publications.

Brooks, A. K., & Edwards, K. (2014). *Consulting in uncertainty: The power of inquiry.* New York: Routledge.

Brooks, F. P. (1987). No silver bullet: Essence and accidents of software engineering. *Computer, 20*(4), 10-19.

Burns, T. E., & Stalker, G. M. (1961). The management of innovation. *University of Illinois at Urbana-Champaign's Academy for Entrepreneurial Leadership Historical Research Reference in Entrepreneurship.*

Butterfield, L. D., Borgen, W. A., Amundson, N. E., & Maglio, A. T. (2005). Fifty years of the critical incident technique: 1954-2004 and beyond. *Qualitative Research, 5*(4), 475-497.

Cardinal, M. (2014). *Executable specifications with Scrum: A practical guide to agile requirements discovery*. Boston: Addison-Wesley.

Charmaz, K. (2006). *Constructing grounded theory: A practical guide through qualitative analysis.* Thousand Oaks, CA: SAGE Publications.

Charmaz, K. (2014) *Constructing grounded theory* (2nd ed.). Thousand Oaks, CA: SAGE Publications.

Cleland-Huang, J. (2014). Don't Fire the Architect! Where Were the Requirements? *Software*, IEEE, 31(2), 27-29.

Davey, B., & Cope, C. (2008). *Requirements Elicitation - What's Missing? Issues in Informing Science and Information Technology, 5*. Retrieved from http://proceedings.informingscience.org/InSITE2008/IISITv5p543-551Davey466.pdf

Davis. A. M. (1993). *Software requirements: Objects, functions, & states*. Englewood Cliffs, NJ: Prentice Hall.

Davis, A. M. (2005). *Just enough requirements: Where software development meets marketing.* New York: Dorset House Publishing.

Davis, C. J., Fuller, R. M., Tremblay, M. C., & Berndt, D. J. (2006). Communication challenges in requirements elicitation and the use of the repertory grid technique. *Journal of Computer Information Systems, 47*, 78-86.

Flanagan, J. C. (1954). The critical incident technique. *Psychological Bulletin, 5*(4), 327-358).

Galal, G. H., & Paul, R. J. (1999). A qualitative scenario approach to managing evolving requirements. *Requirements Engineering, 4*(2), 92-102.

Gale, S. F. (2011). The buzz: Failure rates finally drop. *PM Network, 25*(8), 10-11.

Gottesdiener, E. B. (2002). *Requirements by collaboration: Workshops for defining needs.* New York: Addison-Wesley.

Graham, I., (2003). The compleat requirements analyste. *IEEE Software, 20*(6), 99-101.

Hazzan, O., & Dubinsky, Y. (2008). *Agile Software Engineering.* London: Springer.

Hanson, J. H., & Brophy, P. D. (2012). The critical incident technique: An effective tool for gathering experience from practicing engineers. *Advances in engineering education, 3*(1).

Heaven, W., & Finkelstein A. (2004). UML profile to support requirements engineering with KAOS. *IEE Proceedings-Software, 151*(1), 10-27.

Healthcare Information and Management Systems Society. (2008). *Preparing for success in healthcare information and management systems: the CPHIMS review guide*. Chicago: Author.

International Organization for Standardization. (2011). *ISO/IEC/IEEE 29148:2011 - Systems and software engineering — Life cycle processes — Requirements engineering.* Geneva: Author.

Hazzan, O., & Dubinsky, Y. (2009). *Agile software engineering*. Springer.

Keirsey, D. (1998). *Please understand me II: Temperament, character, intelligence.* Del Mar, CA: Prometheus.

Knowles, M S. (1980). *The modern practice of adult education: From pedagogy to andragogy* (2nd ed.). New York: Cambridge Books.

Knowles, M. S. (1984). *The adult learner: A neglected species* (3rd ed.). Houston: Gulf.

Korpela, M., Mursu, A., & Soriyan, H. A. (2002). Information systems development as an activity. *Computer supported cooperative work, 11*(1-2), 111-128.

Lang, M., & Duggan, J. (2001). A tool to support collaborative software requirements management. *Requirements Engineering, 6*(3), 161-172.

Laplante, P. A. (2014). *Requirements engineering for software and systems*. CRC Press.

Leffingwell, D. (2011). *Agile software requirements: Lean requirements practices for teams, programs, and the enterprise.* Boston: Pearson.

Lenski, G. E. (1970). *Human societies: A macrolevel introduction to society.* New York: McGraw-Hill.

Leviss, J. (2010). *H.I.T. or miss: Lessons learned from health information technology implementations.* Chicago: AHIMA Press.

Lewin, K. (1946). Action research and minority problems. *Journal of Social Issues. 2*(4), 34-46.

Maiden, N. (2005). What has requirements research ever done for us? *IEEE Software, 22*(4), 104-105.

Mason, J. (2002). *Qualitative researching* (2nd ed.). Thousand Oaks, CA: SAGE Publications.

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

McKinsey & Co. (2013). Red team: Discussion document. Retrieved from http://www.npr.org/blogs/alltechconsidered/2013/11/19/246132770/this-slide-shows-why-healthcare-gov-wouldnt-work-at-launch

Munassar, N. M. A., & Govardhan, A. (2010). A comparison between five models of software engineering. *International Journal of Computer Science Issues, 5,* 95-101.

Muise, K., & Wakkary, R. (2010, August). Bridging designers' intentions to outcomes with constructivism. In *Proceedings of the 8th ACM Conference on Designing Interactive Systems* (pp. 320-329). ACM.

Neill, C. J., & Laplante, P. A. (2003). Requirements engineering: The state of the practice. *IEEE Software, 20*(6), 40-45.

Oaklief, C. R. (1976). The critical incident technique: Research applications in the administration of adult and continuing education. Paper presented at the adult education research conference, Toronto, Ontario, April 7-9, 1976.

Orr, K. (2004). Agile requirements: Opportunity or oxymoron? *IEEE Software, 21*(3), 71-73.

Patton, M. Q. (2002). *Qualitative research and evaluation methods* (3rd ed.). Thousand Oaks, CA: SAGE Publications.

Piaget, J. (1970). *Genetic Epistemology* (E. Duckworth, Trans.). New York: Columbia University Press. Retrieved from http://www.marxists.org/reference/subject/philosophy/works/fr/piaget.htm

Pine, V. W., & Barrett, M. L. (2005). What kinds of communications are required on the job? *Journal of Computing Sciences in Colleges, 21*(2), 313-321.

Power, N., & Moynihan, T. (2003). A theory of requirements documentation situated in practice. *Proceedings of the 21st annual international conference on documentation, San Francisco, CA, USA,* 86-92.

Royce, W. (1970). Managing the development of large software systems. *Proceedings of IEEE WESCON 26*(8), 1-9.

Sajid, A., Nayyar, A., & Mohsin, A. (2010). Modern trends towards requirement elicitation. In *Proceedings of the 2010 National Software Engineering Conference* (p. 9). ACM.

Saldaña, J. (2009). *The coding manual for qualitative researchers*. Thousand Oaks, CA: SAGE Publications.

Schwaber, K., & Sutherland, J. (2013). The Scrum guide: The definitive guide to Scrum: The rules of the game. Retrieved from http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

Sinha, V., Sengupta, B., & Chandra, S. (2006). Enabling collaboration in distributed requirements management. *Software, IEEE, 23*(5), 52-61.

Sutcliffe, A., & Sawyer, P. (2013). Requirements elicitation: Towards the unknown unknowns. In *Requirements Engineering Conference (RE), 2013 21st IEEE International* (pp. 92-104). IEEE.

Thomas, D., & Hunt, A. (2004). Nurturing requirements. *IEEE Software, 21*(2), 13-15.

Von Glaserfeld, E. (1989). Cognition, construction of knowledge, and teaching. *Synthese*, 80, 121-140.

Wang, B., Zhao, H., Zhang, W., Jin, Z., & Mei, H. (2010). A problem-driven collaborative approach to eliciting requirements of internetwares. In *Proceedings of the Second Asia-Pacific Symposium on Internetware* (p. 22). ACM.

Wang, T., & Biedermann, S. (2012). Adoption and utilization of electronic health record systems by long-term care facilities in Texas. *Perspectives in Health Information Management/AHIMA, American Health Information Management Association, 9*(Spring).