SOFTWARE PROJECT MANAGEMENT: SOFTWARE METRICS CLASSIFICATION DECISION-MAKING

Ursula Y. Christian and Mohamad Saleh Hammoud
1Professor of Management and Doctoral Chairperson, Northcentral University
2Senior Manager Simulation & Systems Integration Laboratories (SimSIL) Fort Worth (FW), Lockheed Martin

ABSTRACT

The success of software projects is dependent upon useful software metrics, which create value by providing insight into the complex software development environments. The purpose of this qualitative interpretive phenomenological study was to explore how software project management lived experiences of software engineers may influence decision-making in the selection of software metrics for managing, planning, and controlling software development and maintenance tasks for project success. A qualitative interpretive phenomenological research method was employed to explore how lived experiences of software engineers can be used to classify software metrics to meet organization goals. A purposeful homogeneous sample of senior-level software engineers was selected. As a multi-disciplinary field, software engineering, which crosses many technological and social boundaries, was inclined to rely on social and cognitive processes surrounding the human decision-making experience. The findings in this exploratory study provided affirmation of the benefit of studying human experiences in software project management decision-making. The findings identified five major themes and three minor themes, which were supported by the current literature.

Keywords: Software Metrics Classification, Decision-Making for Software Development Tools, Software Project Management, Quality Control, Selection of Software Engineering Tools

INTRODUCTION

Poor management of software quality and processes predicates the continued failure of software projects (Selby, 2009). Volatile requirements, poor planning, unrealistic schedules and budgets, inadequate software metrics, undercapitalization, difference exploitation, and lack of focus on quality are indicators of a poorly managed software project (Silveira, Becker, & Ruiz, 2010). The practice of software engineering requires massive changes to address the inability of software engineers to meet customer needs, as evidenced by historically poor performance in the software engineering field (Chen, 2011). Whether the focus is on technical, programmatic, products and services, processes, organizational, or enterprise management, measurement and analysis identify what is actually happening so that appropriate management could take suitable actions (Zwikael, 2008). Starting from the basic management mantra, plan, act, measure, and control, the measure component supports each of the other three components (Meneely, Smith, & Williams, 2012; Singh, Singh, & Singh, 2011). Additionally, using software metrics provided information for decision-making in control (Mussbacher, Araújo, Moreira, & Amyot, 2012).

Decision-making was the central construct of this interpretive phenomenological study. The decision-making processes associated with selecting useful software metrics were diverse and dynamic (Burge, 2008), and the
absence of useful software metrics to manage planning and control of software development and maintenance tasks, effectively contributed to cost overruns reported for 70% of software projects (Chen, 2011; Gholami et al., 2011). Considering overall failures, which included cancelled projects and delivered projects with unsuccessful performance, 40% to 60% of software projects failed (Conboy, 2010). Unsuccessful projects can negatively affect an organization's financial position by at least 50% of project cost due to risks such as rework, delayed schedules, cost overruns, and scope deficiencies (Suda & Rani, 2010).

A significant number of software engineering researchers have endeavored to understand and improve software performance in organizations by applying pragmatic theories and quantitative methods to research to advance the practice of software engineering (Easterbrook, Singer, Storey, & Damian, 2008; Mahaney & Lederer, 2011). Conversely, the primary problem in this phenomenological study involved real-world complexities in software engineering, which were better explored through qualitative research inquiries that provided deeper insight into dynamic experiences from the perspective of those who lived those experiences (Smith, Flowers, & Larkin, 2010).

RESEARCH PROBLEM AND PURPOSE

The general problem addressed in this study is that when using traditional software metric selection methods, software engineers have been unable to meet customers' needs, as evidenced by historically poor performance in the software engineering field (Silveira et al., 2010; Suda & Rani, 2010). The specific problem is that experiences of software engineers in software metric classification decision-making were not always recognized and valued in the selection of useful software metrics for effective software project management and software project success (Chen, 2011; Hays & Wood, 2011; Nilsson & Wilson, 2012). Project management professionals in the software industry may consent that software metrics significantly improve project success or creates value, but deciding what selection method to use for software metrics is challenging (Mahaney & Lederer, 2011). Continued use of quantitative selection methods to determine useful software metrics has not yielded the desired software project results (Chen, 2011; Gholami et al., 2011). As the dependence on software by major industries grows, late deliveries, total project failures, inadequate software infrastructures, and defective software resulting in litigation and staggering loss in operational income warrant examination of how software engineers can classify software metrics that manage planning and control of software development and maintenance tasks effectively (McManus & Wood-Harper, 2007).

The purpose of this qualitative interpretive phenomenological study was to explore how software project management lived experiences of software engineers may add value to software classification decision-making to select software metrics for managing, planning, and controlling software development and maintenance tasks for project success. Interpretive Phenomenological Analysis (IPA) aims to understand participant perceptions or experiences of their personal world while recognizing the complexities of the analytic process and, therefore, appropriate for this study (Smith et al., 2010).

RESEARCH QUESTIONS AND METHOD

The central phenomenon in the study was the software project management experience of senior-level software engineers (postulated attribute) as reflected by experience in decision-making regarding software metrics classification as critical, essential, or redundant (test performance) (Cronbach & Meehl, 1955). The following research questions guided the inquiry:

Q1. How do lived experiences of software engineers add value to decision-making in the selection of software metrics for planning, managing, and controlling software development and maintenance tasks?

Q2. How do lived experiences of software engineers add value to software metric
classification decision-making for effective project success?

In phenomenological studies, research questions serve as a guide to the exploration of how participants experience the phenomena under study (Moustakas, 1994, Smith et al., 2010). A qualitative interpretive phenomenological study method was chosen to align with the exploratory nature of the study, which required in-depth and detailed participant and phenomena inquiries regarding lived experiences of software engineers with software metrics to foster successful software projects and with software metrics classification decisions (Patton, 2002; Smith et al., 2010). A phenomenological research design is appropriate to address the study problem as phenomenological research allows for exploration of experience to gather comprehensive descriptions, which subsequently can provide the basis for a reflective, inductive structural analysis to analyze the essence of the lived experience (Moustakas, 1994). The focus of this study was to explore how software engineers perceive their project management experiences (Zwikael, 2008) and describe how software engineers perceive software metric classification decision experiences (Boehm & Jain, 2006; Sureshchandar & Leisten, 2006). Furthermore, phenomenology attempts to eliminate pre-judgment or pre-supposition, and requires overt exploration, undisturbed by the natural setting (Moustakas, 1994; Smith et al., 2010). Phenomenology returns to experience in order to obtain comprehensive descriptions as the interviewer enters into participants' perspectives (Moustakas, 1994; Smith et al., 2010; Turner, 2010).

Consequently, the design of the study was also premised on an interpretive approach that assumed participants gave meaning to their experiences through interactions with others (Muller, 2008; Smith et al., 2010). The decision-making processes associated with selecting useful software metrics are diverse and dynamic. An interpretative phenomenological design was adopted to capture information about the software project management experiences of stakeholders involved in the decision-making process surrounding software metric classification (Hays & Wood, 2011). Furthermore, this specific research design allows for an in-depth understanding to the meaning and value of the software project management experiences of research participants regarding decision making through their own voice (Bullen & Love, 2011; Smith et al., 2010). Finally, it helps in gaining an understanding of the issues that software engineers confront when considering software metrics (Hays & Wood, 2011). The interpretivist perspective leads to theory for understanding (Smith et al., 2010). Practitioners of interpretive research design seek to provide understanding of the individual and collective internal experiences for a phenomenon of interest, how participants intentionally and consciously think about their lived experience, valuing subjective experience, and the connection between self and world (Hays & Wood, 2011; Moustakas, 1994).

A purposeful homogeneous sample of 10 experienced, senior-level software engineers was selected from a population of 175 software engineers employed at a local company in a major city in southwest U.S. The local company selected is a global leader in the design, implementation, and support of innovative software solutions, a reputable source for highly skilled software engineers, and a recognized software project management authority. A purposeful homogeneous sample was achieved by selecting participants from a single profession, one organization, and one geographic location. The criterion sampling was used to identify those software engineers who viewed themselves to possess the following requirements (a) BS degree in engineering or computer science field, (b) at least two years of employment at local company, (c) at least five years of software project management experience, (d) involved in at least one failed project, (e) involved in at least one successful project, and (f) advanced understanding of software metrics. Additional characteristics were also imposed since software project management experience is shaped by the cognitive assimilation of software quality management (SQM), software project failure experiences,
software project success experiences, and performance measurement knowledge (El Emam & Koru, 2008; McManus & Wood-Harper, 2007; Moody, 2009; Vitharana & Mone, 2008; Zhou, Vasconcelos, & Nunes; 2011). Software engineers selected for this study had to have the following characteristics:

- planned and managed an effective project transition from software estimating through customer delivery;
- established and maintained an integrated project management plan ensuring the application of earned value (EV) management;
- ensured timely and thorough requirements development; communication, traceability, baselining, configuration control, validation and verification involving all project SCSs;
- achieved project performance, cost, schedule, quality, revenue, and profit objectives;
- established and maintained approved technical, cost, schedule, and resource baselines;
- implemented a risk and opportunity management plan and process;
- measured performance with forward-looking performance metrics;
- pursued a continuous improvement process consistent with operating excellence objectives;
- implemented an appropriate software quality system.

Purposeful homogeneous sampling facilitates in-depth interviews, simplifies analysis, and reduces variation (Smith et al., 2010). There are no conventions for sample size in phenomenological inquiry, but the focus is seeking depth with the available resources and time (Patton, 2002; Smith et al., 2010). Nevertheless, data saturation was achieved with the small sample size because of the aptness of the study participants to provide in-depth, rich data for extracting themes (Mason, 2010; Patton, 2002; Smith et al., 2010). Hence, the sample size was not adjusted based on the richness of the data and the ability to extract themes (Mason, Patton, 2002; Smith et al., 2010).

Semi-structured interview questions were used as the data collection instrument and a computer-assisted qualitative data analysis software (CAQDAS) program was used to augment IPA to increase reliability. The study instrumentation included 13 semi-structured interview questions to explore participant software metric decision-making. The research instrument ensured that the desired results were achieved as expected from the research questions. To ensure credibility, subjectivity, and quality of the data used in the qualitative study, a small panel of experts, consisting a software product manager, software manager, and software quality manager from the local company, validated interview questions prior to data collection to elicit accurate and reliable responses from participants. MAXQDA 11, a CAQDAS software application, was used to collect, organize, and analyze content from interview data, participant observation, analysis of field notes, and digital audio recordings.

A field test was conducted to assess the data collection instrument prior to administration by verifying that (a) interview questions would be implicit, (b) responses to interview questions would explicitly address research questions, and (c) the procedures allowed for effective qualitative interviews (Denzin & Lincoln, 2011; Pritchard & Whiting, 2012). A draft of the interview questions was sent to a software product manager, software manager, and a software quality manager as a field test to determine readiness to proceed with data collection. Feedback from the field test panel was used to refine the questions prior to study administration. Respondents meeting the criteria were purposefully selected to participate in the study to enhance the study's credibility (Patton, 2002; Smith et al., 2010).

Data collection consisted of semi-structured, in-depth interviews with senior-level software engineers in the participants' natural setting for a period of one month. The objective of the interviews and discussions in this study was to facilitate natural conversations, while probing for details and expansion of ideas to understand the project management experiences
of participants. A face-to-face, in-depth interview was conducted with each study participant.

Data analysis through IPA consisted of the following steps: (a) data immersion through reading and re-reading of interview transcripts, (b) detailed exploration through initial noting, developing emergent themes by mapping interrelationships between exploratory notes, (c) searching for associations across emergent themes, (d) moving to the next participant's lived experience, and (e) looking for patterns across participants' lived experiences to address the research questions (Smith et al., 2010). In conjunction with IPA, MAXQDA 11 was used for coding, organizing, and managing the study data to identify recurrent patterns and themes within the data set (Smith et al., 210). Categorization was based on Meneely et al. (2012) five criteria for software metrics: (a) actionability, (b) constructiveness, (c) economic productivity, (d) predictability, and (e) usability. The analysis and interpretation of the narrative contents provided an understanding of the perceptions and experiences from the participants (Smith et al., 2010). Analysis of the interviews was part of a dynamic, iterative, and reflexive process organized around emergent themes (Bullen & Love, 2011; Smith et al., 2010; Simmons, Conlon, Mukhopadhyay, & Yang, 2011). Saturation occurred after the seventh interview when no new themes emerged from additional interviews (Mason, 2010; Patton, 2002; Smith et al., 2010). However, all 10 interviews were completed.

ASSUMPTIONS, LIMITATIONS, AND DELIMITATIONS

We assumed that respondents would provide honest and unbiased responses during the interviews. To minimize the risk of anticipated reprisal or trepidation, participants were assured that all information relating to this study would remain private and confidential through a password protected pseudonym schema (Resta et al., 2010). It was also assumed that the sample size reflects those software engineers with sufficient software project management experience to describe the essence of the lived phenomenon (Patton, 2002). A final assumption was that the interview questions achieve the intent of the guiding research questions and elicit reliable and accurate responses from the participants.

To offset the data collection method limitation, a field test was conducted to assess the data collection instrument to assure clarity and credibility. The study sample derived from the purposive, homogeneous sampling technique may not be easily defensible as being representative of a general population (Patton, 2002); therefore, selection of participants of similar backgrounds and experiences to provide insight into a particular subgroup, such as software engineers, may reduce variation and simplify analysis (Smith et al., 2010). The sampling technique helped to ensure subject matter experts provide information-rich narratives, which provided in-depth insight into project management experiences (Patton, 2002; Smith et al., 2010).

The study was delimited to a single profession, one organization, one geographic location, and specific criteria for participant selection to provide an understanding to the phenomenon being explored in a natural setting (Patton, 2002). The target population consisted of software engineers employed at a local company located in a major city in southwest U.S. The sample was delimited to the following criteria to the target population: (a) BS degree in engineering or computer science field, (b) at least two years of employment at local company, (c) at least five years of software project management experience, (d) involved in at least one failed project, (e) involved in at least one successful project, and (f) basic understanding of software metrics. This qualitative interpretive phenomenological study focused on understanding the meanings and essences of software project management as experienced by software engineers because they could provide a particular perspective on software metric classification (Moustakas, 1994; Patton, 2002; Smith et al., 2010).

LITERATURE REVIEW
The literature review addressed the current research regarding software quality management, software project management experience, valued-based software engineering decision-making, software metrics, and software metric classification. The integration of SQM, software metrics, software project management experience, and value-based software engineering (VBSE) provided the basis of exploration of the research phenomenon.

SOFTWARE QUALITY MANAGEMENT

The interest in software performance measurement has increased as software quality issues attributed to more system failures, which frequently led to higher maintenance costs, longer cycle times, customer dissatisfaction, lower profits, and loss of market share (Vitharana & Mone, 2008). Managing quality efforts remains a major challenge in software development even with the wide use of quality frameworks, such as Total Quality Management (TQM), International Organization of Standardization (ISO) 9000, and Software Engineering Institute (SEI) Capability Maturity Model (CMM).

Software quality management includes all activities that software engineers carry out in an effort to implement a software quality policy. These activities include quality planning, quality control, quality assurance, and quality improvement outlined in a Software Quality Management Plan (SQMP) reflecting the quality management (QM) goals, strategies, and techniques used in managing the software development throughout a software development program. The QM process consists of measurements, such as software metrics, and analyses to aid software engineers in stabilizing the key process elements in order to determine project success. McManus and Wood-Harper (2007) surmised that the application of the QM process allows software engineers to predict future process performance and to establish and achieve software product quality goals based on customer, end user, and developing organization requirements. The experiences and perspective of software engineers through SQM of one software project should translate to better decision-making in the management of the next software project (McManus & Wood-Harper, 2007).

SOFTWARE PROJECT MANAGEMENT EXPERIENCE

Lavrishcheva (2008) defined software engineering as a system of methods and techniques of programming, engineering of project management planning and team process management of manufacturing computer software systems, and methods of measurement and estimation of the compatibility of their various characteristics as to their conformity with customer’s requests. Easterbrook et al. (2008) suggested that software engineers were included as part of the definition of a system, but the software engineer’s role in that system was not clearly defined, and the focus of the software development effort was on the techniques, tools, and technology components that can actually be quantified. What software engineers perceive can be substantially different from objective reality within the software domain (Moustakas, 1994; Patton, 2002). From a phenomenological perspective, interest lies in software engineers experience and how they perceive or interpret the software domain (Patton, 2002). Consequently, Elm et al. (2008) determined that it was critical for software engineers to be better integrated into multidisciplinary software development teams. This integration facilitated success in transforming the software engineer’s cognitive needs into a set of software engineering products by providing this critically needed input to the software development process (Elm et al., 2008). Accordingly, software engineers are more than just another software system component. The proper scope of a software system is the combination of software engineering perception, software project management experience and techniques, tools, and technology that must act as mutual agents to achieve goals and objectives in a complex software development domain (Easterbrook et al., 2008; Elm et al., 2008). In this study, the tool is software metrics and the mutual agents become an integrated human-
driven decision-making subsystem intended to achieve software project success.

**VALUED-BASED SOFTWARE ENGINEERING DECISION-MAKING**

Boehm and Jain (2006) addressed bringing value to an organization through project success with the 4+1 theory of VBSE, which includes Theory W, dependency theory, utility theory, decision theory, and control theory. The 4+1 theory of VBSE framework is based on an interrelated set of theories of planning and managing valuation of software decisions; therefore, project management is based on value realization rather than project completion (Boehm, 1996; Boehm & Jain, 2006; Nilsson & Wilson, 2012). The perspective addressed by Boehm and Jain (2006) encompassed both subjective meaning of experiences and a model of value realization, which was useful in achieving project success by providing a decision support system. Additionally, Boehm and Ross (1989) used 4+1 theory of VBSE as the theoretical lens to provide a conceptual model to view data and understand various emerging themes; and to provide a unifying framework for stakeholders to reason about software decisions in terms of value created through software. Burge (2008) asserted that value-based planning and control differed most significantly from traditional project planning and control in its emphasis on monitoring progress toward value realization rather than towards project completion. Based on Boehm’s win–win Theory W, a methodology at the center of 4+1 theory of VBSE, which also includes utility theory, dependency theory, decision theory, and control theory, Burge (2008) asserted that considering value when making software engineering decisions is good business acumen. The 4+1 theory of VBSE addresses all of the cogitations of computer science theory, including considerations involved in the managerial aspects of software engineering, personal, cultural, and economic values involved in developing and evolving successful software-intensive systems (Boehm & Jain, 2006; Burge, 2008).

**SOFTWARE METRICS**

Measurement and analysis, otherwise referred to as software metrics, are fundamental elements of project management (Gholami et al., 2011; Mahaney & Lederer, 2011). Zwikael (2008) asserted that whether the focus is on technical, programmatic, products and services, processes, organizational, or enterprise management, measurement and analysis identify what was actually happening so that appropriate management could take suitable actions. Starting from the basic management mantra, plan, act, measure, and control, the measure component supports each of the other three components (Gholami et al., 2011; Meneely et al., 2012; Singh et al., 2011). Additionally, using software metrics provide information for decision-making in control (Mussbacher et al., 2012). According to Mussbacher et al. (2012), using software metrics provided documented history for correlating data to generate leading indicators or developing sophisticated predictive models or forecasts to improve plan. Further, the use of software metrics provided assessments of process performance to improve the processes executed in act (Singh et al., 2011). Software metrics is fundamental to management independent of the class or level of management being discussed. Singh et al. (2011) asserted that software quality and project success can be achieved by making the collection and use of software metrics an integral part of the software development process to foster continuous improvement actions. Software metrics provide project and organizational management with quality information that facilitates early identification of problems and appropriate corrective actions.

An integrated set of systematic measurement activities designed to support management, technical, and process decision-makers provide a framework to ensure customers of mission success; to improve business execution; and to provide customers with best value results that will keep them coming back (Gholami et al., 2011; Mahaney & Lederer, 2011). An organization’s mission success
transcended disciplines and methods, requiring an appropriate selection, definition, and use of effective measures and quantitative techniques to support project performance (Ananatmula, 2010). However, Singh et al. (2011) recognized the limitations of software metrics and advised that it was also necessary to recognize that measurement processes are subject to random and systematic errors, that analyses depend on modeling relationships among measures and processes that are fuzzy logic, and that information and communication can be imprecise. Software metrics is a tool to aid in managerial and technical decision-making and not a panacea for project success (Singh et al., 2011).

Criteria for software metrics. A review of extant, academic literature revealed at least 47 unique validation criteria for software metrics (Meneely et al., 2012). The intended use of a software metric or its alignment with business goals directs SCSs toward specific properties of a metric that are more appropriate for that use (Gibler, Gibler, & Anderson, 2010). For the purpose of this study, these were considered advantages. Since a value-driven philosophy based on VBSE, which purports that the primary purpose of a metric is to use it in assessment, prediction, and improvement, was adopted, five advantages related to value-driven perspectives were selected (Nilsson & Wilson, 2012). The following five were selected for use in this study based on the works of Begier (2010), Ellis and Levy (2008), McManus and Wood-Harper (2007), and Sureshchandar and Leisten (2006): (a) practicality, (b) efficiency, (c) decision-informing, (d) quality-focused, and (e) theory-building. Of the 47 criteria, five criteria that provided the desired advantages were selected for this phenomenological study: (a) actionability, (b) constructiveness, (c) economic productivity, (d) predictability, (e) and usability (Meneely et al., 2012).

Software metric types. There are three categories of software metrics that most engineers apply to software projects to address organization, program, and project goals: product, project, and process metrics (Matalonga & San Feliu, 2011; Peterson, 2011; Silveira et al., 2010; Suda & Rani, 2010; Symons, 2010). Table 1 provides a summary of the product, project, and process metrics. Mussbacher et al. (2012) asserted that organizations use Key Performance Indicators (KPI) to define and measure progress toward business goals. Once the organization has performed a systematic analysis of its mission, identified all its SCSs, and defined its organization, program, or project goals, it needs a way to measure progress toward those goals and reflect the critical success factors of an organization (Gibler et al., 2010; Mussbacher et al., 2012); key performance indicators are those measurements. Mussbacher et al. (2012) contended that whatever KPIs were selected, they must reflect the organization, program, and project goals; they must be critical to its success; and they must be measurable. The goals for a particular KPI may change as the organization, program, or project goals change, or as the use of a KPI drives an organization closer to achieving a goal specified for that KPI (Gibler et al., 2010; Mussbacher et al., 2012).

Table 1

<table>
<thead>
<tr>
<th>Software Metric Type</th>
<th>Key Performance Indicators</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>Requirements Stability</td>
</tr>
<tr>
<td></td>
<td>Software Size Stability</td>
</tr>
<tr>
<td></td>
<td>Computer Resource Utilization</td>
</tr>
<tr>
<td></td>
<td>Defect Profile</td>
</tr>
<tr>
<td></td>
<td>Software Product Evaluation (SPE) Defect Density</td>
</tr>
</tbody>
</table>
SOFTWARE METRIC CLASSIFICATION

Several studies asserted that software metric classification should align with the goals of an organization to help communicate, measure progress towards, and eventually attain those goals (Desouza et al., 2009; Meneely et al., 2012). Well-designed metrics with documented selection criteria can help an organization obtain the information it needs to continue to improve its software products, processes, and services while maintaining a focus on creating value through project success (Symons, 2010). The discipline of software metrics involves identifying multifarious attributes to measure and determining how to measure those attributes in developing quality software (Vitharana & Mone, 2008). Bullen and Love (2011) revealed that managing the whole software development life cycle, which was (a) concept stage, (b) requirements stage, (c) design stage, (d) implementation stage, (e) test stage, (f) installation and checkout stage, (g) operation and maintenance stage, and (h) retirement stage, was a fundamental strategy to optimize project success and profitability of software projects. By utilizing useful software metrics, software engineers obtain the ability to make informed decisions and better choices. Importance of software metrics is stressed in TQM, ISO 9000, and CMM, widely used quality frameworks in the software industry (McManus & Wood-Harper, 2007). However, software engineers acknowledged difficulties in implementing these software quality frameworks (Vitharana & Mone, 2008). In many cases, software organizations embarked on costly adoption and implementation of TQM, ISO 9000, or CMM without aligning their use with organization goals or without providing value-based rationale for implementation (Burge, 2008; van Solingen, 2009; Vitharana & Mone, 2008). Sureshchandar and Leisten (2006) examined the relative importance of software metrics from the standpoint of their value towards improving business performance. Sureshchandar and Leisten (2006) sought to provide a benchmark for the evaluation of metrics from the perspective of measurement priorities or critical, essential, or redundant classification. Measures classified as critical, essential, or redundant could help decision makers decide what to focus on and the significant few metrics, instead of spending time, effort, and money on the insignificant many. Selecting software metrics based on a set of relevant criteria, from a business perspective, is especially useful in identifying key measures with respect to software metrics aligned with an organization’s business objectives (McManus & Wood-Harper, 2007; Vitharana & Mone, 2008).

Software metrics give software engineers the ability to make conversant and informed decisions. Understanding how particular software metrics, used to measure either product, process, or project performance, can assist in achieving business goals, whether organization, program, or project, is key to business success (Gibler et al., 2010; Singh et al. 2011).
RESULTS AND EVALUATION OF THE FINDINGS

Research question 1 focused on the study construct, decision-making, and the value of selecting metrics useful in planning, managing, and controlling software projects. Research question 1 had a dependency on research question 2. Research question 2 also focused on decision-making, but embedded the value of classifying software metrics to aid in the selection of useful metrics. As a result, five major themes and three minor themes emerged from data analysis. The five major themes associated with research questions 1 and 2 were: (a) intrinsic value of software metrics, (b) adaptive change management, (c) experience-based decision-making, (d) systemic reasons for software project failures, and (e) project size/type. Additionally, the three minor themes consistent with research question 1 were: (a) organizational processes, (b) data-driven decision-making, and (c) customer expectations. Figure 1 depicts the relationship between the research questions and the major and minor themes.

Figure 1. Mapping themes to research questions. This diagram shows how the major and minor themes map to the two research questions.
Major Theme 1: Intrinsic Value of Software Metrics

The findings indicated that companies evaluate software metrics based on the true value these metrics deliver to the organization and not on the actual value the company pays to buy and use the software metrics. Researchers have reported findings consistent with major theme 1, which was the intrinsic value of software metrics. Intrinsic value of software metrics was consistent with the value-driven philosophy based on VBSE, which has been historically purported the primary purpose of a metric for use in assessment, prediction, and improvement (Burge, 2008; Gholami et al., 2011; Mahaney & Lederer, 2011; Zwikael, 2008). Similarly, Burge (2008) argued that the emphasis of value-based planning and control was on monitoring progress toward value realization rather than schedule completion. Likewise, Gholami et al. (2011) asserted that measurement of software had no meaning without software metrics. Thus, the findings of Burge and Gholami et al. were comparable to major theme 1 as to the value of software metrics as a tool in monitoring software project progress, guiding decisions of change, and ensuring project success instead of marking time. In addition, Mahaney and Lederer (2011) concluded that the application of software metrics led directly to project success, which was comparable to major theme 1 as to the predictive power of software metrics. Finally, Zwikael (2008) identified software metrics as one of the six critical processes and practices for project success, as in major theme 1 for the application of software metrics to projects to ensure success.

Conversely, past literature contrasted with major theme 1 for the unconditional intrinsic value of software metrics (Pfleeger, 2009; Selby, 2009). Pfleeger’s (2009) supposition was that the value or utility of any software metric depended on the extent to which it is able to explain how it helps an organization to improve. Similarly, Selby (2009) asserted that if there was no justification for using a particular metric, there was no justification for applying it. Thus, the findings of Pfleeger and Selby contrasted with major theme 1 as to the unassailable value of software metrics.

During data analysis, advantages of software metrics study participants used when making software metric classification and selection decisions were identified. The three major advantages of using software metrics found in this study were decision-informing, practicality, and quality-focused. Neither efficiency nor theory building was identified in this study as advantages of using software metrics. Participants in this study placed more value on metrics that allowed them to (a) make informed decisions while developing or maintaining software (decision-informing), (b) improve software development processes and practices (practicality), and (c) improve the customer relations through high quality software (quality-focused); these findings were in line with those of Meneely et al. (2012). Less value was placed on (a) saving time and effort by choosing a single metric for a given measurement goal (efficiency) and (b) making general statements about the nature of software and its development (theory building) (Meneely et al., 2012). This was in line with Jamieson and Lohman’s (2009) assertion that mission and goals of engineering academia differed from those of practicing engineers. Based on the mapping of software metric criteria to advantages, participants in this study favored usability, actionability, economic productivity, predictability, and constructiveness when classifying and selecting software metrics.

Major Theme 2: Adaptive Change Management

Adaptive change management reflected the perceptions of study participants regarding project management decisions based on what KPIs reveal about the software product, project, or process to resolve software issues. Researchers reported the significance of adaptive change management to predict project performance, guide strategic decision-making, and control software project risks (Chen, 2011; Nilsson & Wilson, 2012; Symons, 2010). Chen (2011) affirmed that adaptive change had a strong, stable, discriminatory power to predict
success and failure, which was analogous with this theme regarding the value of applying adaptive change to software projects. Symons (2010) asserted that adaptive change management should be fully integrated into the project design and decision-making to enable strategic project management direction, and Symons’ findings were comparable to major theme 2 in the regard that situational awareness and foresight were essential to changing the course of a project to ensure project success. Furthermore, Suda and Rani (2010) acknowledged that deferred decisions regarding software adaptive changes to right the ship posed risks to software project success as the software project progressed, which substantiated major theme 2 regarding addressing change at the appropriate time in a project to reduce risk of project failure. Likewise, Nilsson and Wilson (2012) confirmed that project managers should forego being irrational by assessing risk only, but should actively manage risk through adaptive change, which was also comparable to major theme 2 regarding perceptions in this study regarding deferred decisions on adaptive change that may lead to project failure.

Bareil’s findings contrasted with major theme 2 regarding the perceptions of leadership resistance to adaptive change as a negligent behavior and the ineffectiveness of leaders as change agents. Bareil (2013) posited that in many cases leaders act responsibly by resisting change given the high failure rate of change implementation. Further, Bareil (2013) asserted that resistance to change, from a traditional paradigm perspective, was perceived to hinder organizational progress, but in the modern paradigm perspective resistance to change was used to elicit clarity and understanding of the change framework to ensure organizational success. Thus, Bareil’s finding contrasted with major theme 2 for leadership resistance to adaptive change.

**Major Theme 3: Experience-Based Decision-Making**

Experience-based decision-making characterized participant perceptions for benefits derived from project management experience to guide decisions, which included classifying software metrics as critical, essential, or redundant to guide software metric selection decisions. Past literature was analogous with major theme 3 regarding decision dependency on human interaction for adapting change and controlling progress (Anantatmula, 2010; Baba & Hakem-Zadeh, 2012; Bryde & Robinson, 2007; Burge, 2008; El Emam & Koru, 2008; Vareman, 2008). Burge (2008) affirmed that dependency theory elucidated the underlying implications of diverse human interactions on decisions when integrating software engineers with software processes, and control theory explicated that the necessary conditions for successful project control were observability, predictability, controllability, and stability. Similarly, Bryde and Robinson (2007) affirmed that the application of the necessary conditions, which Burge identified, to people-intensive software projects does not permit the use of observability and controllability equations as precise as the equations do in other engineering disciplines, but they capture most of the wisdom provided by software management thought leaders. Thus, the findings of Burge (2008) and Bryde and Robinson (2007) were comparable to major theme 3 for lived experience guiding decisions. Likewise, Baba and Hakem-Zadeh (2012) acknowledged that the process of decision-making was not a purely rational process and decision-makers perceived and utilized evidence differently based on their experience and judgment, and Vareman (2008) affirmed that decisions could be made with certainty because decision-makers were fully informed. Baba and Hakem-Zadeh and Vareman’s findings were comparable with major theme 3 in that extensive project management experience provided a cogent framework of certainty for classifying software metrics as critical, essential, or redundant. Additionally, Anantatmula (2010) asserted that successful decision-making benefits from previous experience regardless if that experience resulted in a successful or failed software project was also comparable to major theme 3 for perceptions of lived experiences that guide decisions and lead to project success. Finally, El Emam and Koru (2008) concluded that project management experience affected
project success and admonished software organizations to document project lessons learned to avoid repeating organizational behavior that leads to project failures.

Selby (2009), and Silveira et al. (2010) brought merit to the software metric selection process in their respective studies, but did not emphasize the key role of experience and knowledge in that selection process. The narrow focus on the right metrics results is one-dimensional aspects of software development instead of creating optimal criteria based on the knowledge and experience of software engineers to facilitate selection of measurement alternatives for multiple phases of software development.

Major Theme 4: Systemic Reasons for Software Project Failure

Systemic reasons for software project failure embodied participant perceptions that there were common reasons for software project failures that persisted within software organizations. The systemic reasons were related to inadequate performance measures. The fourth major theme was a reverberation of past literature for the recurring reasons for software project failures (Gholami et al., 2011; Kahura, 2013; Nilsson & Wilson, 2012; Yang, Hu, & Jia, 2008). For example, Nilsson and Wilson (2012) cited personnel shortfalls, performance shortfalls, unrealistic planning and schedules, and continuing stream of requirements change as risk factors to project success. These findings were similar to major theme 4 for risks to project success. Yang et al. (2008) acknowledged changing scope or volatile requirements as a precursor to software project failure, which was comparable to major theme 4 by reiterating the requirements of volatility affect, also articulated as baseline control, on project success. The lack of adequate software metrics for monitoring and measurement of software life cycle phases, which caused low quality and usefulness of software products was cited as reasons software projects failed by Gholami et al. (2011), and accordingly, the findings of Gholami et al. was also comparable to major theme 4. Additionally, Symons (2010) asserted that poor software quality was a result of change management shortcomings, which was an inhibitor to the success of software projects. Thus, Symons' findings were similar to the thematic expression of major theme 4. Finally, Kahura (2013) cited the importance of competition as a forced means for organizations to remain relevant in their respective fields. As a result, software quality improved and the risk of software project failure decreased; thus, Kahura's findings were an analogous with major theme 4.

Major Theme 5: Project Size/Type

Project size/type characterized perceptions that project size and/or project type was a factor for selecting software metrics, classifying software metrics as critical, essential, or redundant, predicting project success or failure, or accepting accountability. Major theme 5 was comparable in the current research for project size/type as a factor in value realization (Conboy, 2010; Di Tullio & Bahli, 2013; Lagerstrom, von Würtemberg, Holm, & Luczak, 2012; Silveira et al., 2010). Silveira et al. (2010) determined that project size had distinct project management implications that influence decisions regarding software metric classification, software metric selection, and software metric usage in the project environment. The findings of Silveira et al. were comparable with theme 5 in that project/size was considered when making software metric classification and selection decisions. Project management professionals in the software industry consented that software metrics significantly improved project success or created value (Chen, 2011; Gholami et al., 2011; Sureshchandar & Leisten, 2006; Zwikael, 2008), but project size was an added risk to the decision-making process regarding which software metrics for a given project would be useful to abet project success (Di Tullio & Bahli, 2013). Thus, the finding of Di Tullio and Bahli was also comparable with this theme regarding project size as a factor in selecting software metrics. Finally, Lagerstrom et al. (2012) found project type significantly affected software productivity, and maintenance projects were
twice as productive as development projects, which was comparable to major theme 5.

Minor Theme 1: Organizational Processes

Organizational processes embodied perceptions for organizational culture, mode of operation, and expected behaviors as a framework for decision-making. Minor theme 1 was comparable to findings in the current research, which referred to organizational processes as critical top management support processes, a factor of project success when embedded in project management decisions (Conboy, 2010; Di Tullio & Bahli, 2012; Mahaney & Lederer, 2011; Zwikael, 2008). Zwikael (2008) indicated that managers were not aware of, or preferred to ignore, the impact various supporting processes had on project success, which was comparable to minor theme 1. Likewise, Di Tullio and Bahli (2012) concurred that organizational processes were dependencies to project success and identified organizational support as one of the risks to software development; thus, this finding of Di Tullio and Bahli was also comparable to minor theme 1 by reflecting perceptions of leadership support at the right levels. Additionally, Mahaney and Lederer (2011) indicated that top management should focus on organizational processes, such as managing software metrics, to meet customer expectations and improve project outcomes. Likewise, Conboy (2010) affirmed that organizational culture was one of the factors affecting cost control, which is measured using the cost performance metric. Thus, the findings of Mahaney and Lederer and Conboy were similar to minor theme 1.

Findings in the current literature contrasted with minor theme 1 on goal alignment between program, project, and organization. Mahaney and Lederer (2011) concluded that even when goals were in conflict within an organization, project success could still be achieved because organizational commitment was a greater motivator; thus, Mahaney and Lederer’s findings contrasted with minor theme 1 as to goal congruity.

Minor Theme 2: Data-driven Decision-Making

Data-driven decision-making characterizes the interpretation of participant perceptions in this study about using prescriptive data from software metrics to guide decisions. Minor theme 2 was comparable to findings in current literature as the best decision to make based on the presupposition that the decision-maker was fully informed and rational and had the ability to compute with perfect accuracy (Baba and HakemZadeh, 2012; Suda & Rani, 2010; Vareman, 2008). Baba and HakemZadeh (2012) concluded that managers need reliable evidence in order to be able to make solid and effective decisions. Likewise, Vareman (2008) affirmed that decisions occur under certainty when the consequences of those decisions were sure. Thus, the findings of Baba and HakemZadeh and Vareman were comparable to minor theme 2 in that all decisions should be influenced by data. Furthermore, decisions made under uncertainty because decision-makers were not fully informed or data was not available posed a threat to project success because issues were not resolved as the software project progressed (Suda & Rani, 2010). Thus, the findings of Suda and Rani were analogous to minor theme 2 from the perspective that ignoring data from software metrics in favor of schedule-driven decisions posed a risk to project success.

Additionally, past researchers also indicated that software metrics provided project and organizational management with quality information that facilitated early identification of problems and appropriate corrective actions (Brothers et al., 2009; El Emam & Koru, 2008; Gholami et al., 2011; Meneely et al., 2012). Gholami et al. (2011) asserted that the data obtained from software metrics as feedback should be given to the software manager in order to find existing faults, provide solution for those faults, and prevent further rising of faults. Similarly, Meneely et al. (2012) identified one of the advantages to using software metrics as decision informing, which promoted actionability, constructiveness, and economic productivity, three of the five criteria mentioned in this study for selecting software metrics. Thus, the findings of Gholami et al., Meneely et al.,
and Brothers et al. were comparable with the thematic assertion of minor theme 2 that data obtained from software metrics should be used to guide decisions and may improve project outcomes. Finally, El Emam and Koru (2008) admonished software organizations to document project lessons learned to establish a data repository of knowledge base to facilitate data-driven decision-making, which was also comparable with minor theme 2 on capturing best practices to effect data-driven decision-making.

**Minor Theme 3: Customer Expectations**

Customer expectations characterized participant perceptions in the current study that satisfying customer requirements resulted in project success and have on program management decision-making. Minor theme 3 was comparable to past literature as past researchers found when the desired schedule, cost, scope, or quality constraints of a software project were realized, a software project was deemed successful; thereby, customer expectations were met (El Emam & Koru, 2008). Likewise, Chen (2011) affirmed customer benefit as a factor in assessing project success, and Vitharana and Mone (2008) asserted that compliance to customer requirements resulted in project success. Thus, the findings of El Emam and Koru, Chen, and Vitharana and Mone were comparable to minor theme 3 in that customer expectations should be realized to affirm project success.

In contrast, past researchers did not acknowledge the interdependency between customer expectations and project success, which was the negative impact of customer expectations on schedule, cost, scope, or quality constraints through requirement or scope. Past researchers acknowledged the effect of requirements instability on project success (Suda & Rani, 2010; Symons, 2010; Yang et al., 2008), but did not explicitly link customer expectations with a resultant negative impact on project success. Suda and Rani (2010) revealed that increases in project requirements during software development, beyond those originally foreseen, occurred when the scope of a project was not adequately defined, documented, or controlled. Subsequently, Symons (2010) asserted that requirement changes should be accounted for in the early phases of the software lifecycle because changes adversely affected schedule, cost, and quality. Likewise, Yang et al. (2008) identified customers as the originators of requirements creep, a term referred to by software practitioners to explain inadequately defined, documented, or controlled requirements. However, neither the findings of Suda and Rani, Symons, nor of Yang et al. implicated customers as a source of requirements instability.

**RECOMMENDATIONS**

**Recommendations for Practice**

As supported by major theme 4 and minor theme 3, the first recommendation is for software organization decision-makers to include system requirements and acceptance criteria as part of the statement or work or contract. By working with customers prior to the software development phases to define the system requirements and acceptance tests of the final product, software organization decision-makers can ensure scope boundary conditions throughout the software development cycle. Any changes to those boundary conditions would constitute a contract change instead of adversely affecting existing schedule, cost, and quality (Symons, 2010; White, 2013). In successful programs, system requirements were properly prioritized with the customer and change was managed at the system level (Anantatmula, 2010; Chen, 2011).

The second recommendation for practice is for software leaders to place more emphasis on managing, monitoring, and controlling software products and processes instead of focusing primarily on project performance, as supported by major themes 1, 2, 3, and 4, and minor themes 1 and 2. Software executives and senior-level software engineers should ensure that they are working toward the same overarching, not necessarily specific, goals within the organization. Software executives have a tendency to focus on earned value (EV) or projects metrics, such as cost performance and schedule performance metrics, which are more likely to be tied to their employee incentive
awards. Whereas, software engineers focus on product and process metrics, which they believe will result in positive cost and schedule performance. Employee incentive awards of software executives should truly reflect the importance of product and process management by including stipulations to that regard to change behavior toward judicious adaptive change. It would be naïve to think that such a change in behavior would occur without some type of monetary incentive. Linking software product and process improvements to profitability by modifying the behavior of software executives is one of many steps to a continuum of change necessary to improve software project failure rates.

As supported by major themes 2 and 3 and minor themes 1 and 2, the third recommendation is to strengthen relationships between software executives and senior-level software engineers to facilitate trust between these decision-makers. From a software executive perspective, diminishing tenures, mixing of diverse organizational cultures, condescension, cronyism, and leadership selection homogeneity results in organizational mistrust and disarray, and increasing collateral damage amongst an intellectual workforce. Software executives should return to strategic planning, hold software engineering managers accountable for the tactical execution of software projects, and rely on their project management experience and understanding of organizational processes. However, in their quest for the appropriate level of decision-making, software engineering managers should be cognizant that trust is built on evidence not supposition or emotion. Software engineers and managers should provide reliable data, such as software metrics applied to the software system, to clearly define and communicate project risks to software executives to develop trust and reduce resistance to requests for adaptive change.

The fourth and final recommendation for practice is that project lessons learned by software engineers be documented and used to support decision-making in risk management (Zhou et al., 2011) and contingency planning to alleviate uncertainty and turmoil around decisions regarding software corrective, perfective, or adaptive changes (Suda & Rani, 2010). Acting on this recommendation would also allow future project teams to leverage the learning experience of others in previously executed projects in order to avoid the some of the same consequences. As supported by major theme 3, experienced-based decision-making, successful decision-making benefits from previous experience (Anantatmula, 2010) and project management experience affects project success (El Emam & Koru, 2008).

RECOMMENDATIONS FOR FUTURE RESEARCH

This qualitative interpretive phenomenological study was delimited to a single profession, one organization, one geographic location, and specific criteria for participant, which resulted in a homogeneous sample. The first recommendation for future research is to expand and replicate the qualitative phenomenological study to explore the perspectives and lived experiences of software engineers in multifarious organizations, geographic locations, years of project management experience, and administrators. The benefit would be to compare the perspectives of software engineers from different organizations, geographic locations, or experience levels to provide more insight into software project management decision-making.

A second recommendation for future research is to conduct a qualitative multiple-case study (Conboy, 2010) to further explore major theme 3, experienced-based decision-making. This theme illustrated that there was a dependency on human interaction for adapting change and controlling progress, and future research could explore the perceptions of experienced software executives (directors and above), less experienced software engineers, or software engineers from multifarious organizations to provide insight into software metric selection and classification decision-making.

Similarly, the final recommendation for future research is to conduct qualitative single-case studies (Lagerstrom et al., 2012) to further explore major theme 4, systemic reasons for
software project failures. Major theme 4 illustrated the multiple reasons software project failures failed: inadequate technical baseline control (Silveira et al., 2010; Symons, 2010), performance shortfalls (Jovel & Jain, 2009; Wang, Kundu, Limaye, Ganai, & Gupta, 2011), deficient concurrency management (Stankovic & Tillo, 2009), and inadequate performance measurement (Gholami et al., 2011).

COMPLIANCE WITH ETHICAL STANDARDS

Since the research involved human participants, care was exercised when collecting, handling, recording, and storing data to ensure ethics and integrity in the study (Erlen, 2010). Data collection did not commence until the Northcentral University Institutional Review Board approved the qualitative interpretive phenomenological study. In one-on-one briefings, each participant was informed of (a) the study purpose, (b) estimated interview duration, (c) research points of contact, (d) potential risks associated with the study, (e) any latent benefits of the study, (f) the procedures for maintaining confidentiality, (g) and the anticipated consequences of declining or withdrawing from the study and refusal to answer specific questions. Informed consent forms were used to document voluntary consent and permission prior to data collection (Erlen, 2010). Each study participant signed an informed consent form prior to commencement of interviews.

Privacy and confidentiality were protected through a pseudonym schema to encourage forthrightness to increases trustworthiness and integrity (Resta et al., 2010). A coding list of pseudonyms was password-protected during the study, and all audio-recorded sessions and notes taken in a bounded field journal was secured. To facilitate future research and ensure the privacy and confidentiality of participants, the research plans and activities were recorded in an electronic, password-protected research journal. All records related to the study will be secured for five years for research posterity.

REFERENCES

immobilizing radioactive liquid process waste stored in Saluggia, Italy. Decision Analysis, 6(2), 98–114. doi:10.1287/deca.1090.0140


