Measuring Critical Factors of Software Quality Management: Development and Validation of an Instrument

Padmal Vitharana, Syracuse University, USA
Mark A. Mone, University Wisconsin - Milwaukee, USA

ABSTRACT

Literature presents attributes of effective quality management in building software systems. The value of the capability maturity model (CMM), ISO 9000, and total quality management (TQM) concepts in managing software quality has been widely acknowledged. However, shortcomings of these approaches have also been recognized. For instance, scholars have acknowledged CMM implementation difficulties because of its inherent complexity. This research synthesizes existing literature base in CMM, ISO 9000, TQM, among others, to identify six critical factors of software quality management (SQM) and then develops an instrument that can be used to measure critical factors of SQM. Validity and reliability are established by reviewing extant literature, testing a preliminary version of the instrument among a group of researchers and industry experts, and empirically testing a revised version of the instrument among a group of IS professionals. The instrument is relatively easy to implement and requires minimal resources. Implications for research and practice are discussed.

Keywords: instrument; management; quality; software; validation

INTRODUCTION

Recently, there has been an increasing emphasis on quality in developing software (Duggan, 2004; Haag, Raja, & Schkade, 1996; Harter & Slaughter, 2003; Prajogo & Sohal, 2006). The quality of a software system is widely accepted as its conformance to customer requirements (Kan, Basili, & Shapiro, 1994). The interest in quality is heightened as more system failures are attributed to issues in software quality that often lead to higher maintenance costs, longer cycle times, customer dissatisfaction, lower profits, and loss of market share (Arthur, 1993; Gopal, Krishnan, Mukhopadhyay, & Goldenson, 2002; Kan, Basili, & Shapiro, 1994). For instance, a software problem with one of the switches left AT&T customers nationwide without service for 26 hours, costing the company about $1 billion and 6.25 cents per share loss (New York Times, 1998). In a more recent report, AT&T claims
that it spends roughly $1 million a month to
patch defective software built by others (Wall
Street Journal, 1998). Although the importance
of quality is acknowledged, managing quality
efforts remains a major challenge in software
development. In this context, software quality
management refers to the process of carrying
out key management practices necessary for
achieving software quality.

The problems associated with inadequate
software quality belie the amount of research on
how software quality should be managed. Such
normative work typically reports software qual-
ity management practices of successful compa-
nies by consultants, researchers, and managers.
For example, Kan et al. (1994) discuss software
quality in the context of total quality manage-
ment (TQM). Although the TQM philosophy
in general emphasizes continuous improvement
in quality, various TQM advocates prescribe a
diverse array of techniques for quality manage-
ment. Several authors present evidence of using
TQM to improve software quality (e.g., Arthur
1993; Dunn & Ullman, 1994; Prajogo & Sohal,
2006; Manz & Stewart 1997; Victor, Boynton,
& Stephens-Jahng, 2000). For instance, Ravi-
chandan and Rai (2000) apply TQM principles
to information systems development and derive
a set of scales for assessing quality constructs.
Others have acknowledged differences between
soft (behavioral) TQM factors such as employee
commitment and hard (technical) TQM such
as statistical testing and their implications for
quality performance (e.g., Rahman & Bullock,
2005). Still others have examined the mediat-
ing role of TQM on the relationship between
firm strategy and firm performance (Prajogo
& Sohal, 2006). TQM, however, represents
only one stream of research applicable to the
management of software quality.

Software quality management is also dis-
cussed in the context of ISO 9001 and ISO
9000-3 certification, and more recently, SPICE
(Jenner, 1995; Kehoe & Jarvis, 1996; Jung,
2005; Yoo, Yoon, Lee, & Lee, 2006). By achiev-
ing ISO certification, an organization is able to
conduct business with those (e.g., customers or
vendors) who require that their partners adhere
to accepted quality standards. However, because
of possible disruption of normal operations and
long duration of the auditing process, the costs
involved in ISO certification could be consider-
able to some organizations. Moreover, because
ISO 9001 only defines minimum qualifications a
firm needs to achieve for certification (Bamford
& Deibler, 1993), it lacks support for continu-
ous improvement in software quality (Coallier,
1994). In addition, ISO 9001’s high level of
abstraction has caused auditors to interpret it
in different ways (Paulk, 1995).

The capability maturity model (CMM²)
developed by Software Engineering Institute
(SEI) details a well-defined approach to soft-
ware process improvement (Harter, Krishna,
While ISO 9000 highlights a more inter-orga-
nizational approach (e.g., vendor relationships)
to managing quality, CMM takes a more intra-
organizational strategy to quality management.
Although CMM has been used in the industry as
a means to assessing software process maturity,
there is some criticism of its 5-level maturity
model. For example, Saiedian and Kuzara
(1995) claim that CMM is not an exhaustive
model in that it does not address several software
management and engineering practices crucial
for project success and that because of CMM’s
roots in the governmental and defense-oriented
software arena, its assumptions may not neces-
sarily hold true in the commercial sector. Fur-
thermore, Stelzer, Mellis, and Herzwurm (1997)
highlight that CMM lacks a solid understanding
and an explicit model for explaining the causes
and effects of software quality management. It
may be ok, but please consider. More recently,
Niazi, Wilson, and Zowghi (2005a) observe that
because of complexity of CMM, little attention
has been paid to their implementation resulting
in limited success in many software process
improvement efforts. Notwithstanding their
shortcomings, ISO 9000 and CMM have helped
several organizations improve their ability to
develop quality software (Harter et al., 2000).
More recently, the emergence of the capabil-
ity maturity model integration (CMMI) and
ISO’s SPICE typify the continued importance