TESTQUAL: Conceptualizing Software Testing as a Service

Yang Yang, First Tennessee Bank
Colin G. Onita, University of Memphis
Xihui Zhang, University of Memphis
Jasbir Dhaliwal, University of Memphis

Available at: https://works.bepress.com/colin-onita/10/
TESTQUAL: Conceptualizing Software Testing as a Service

Yang Yang
First Tennessee Bank, Memphis, Tennessee

Colin Onita
University of Memphis, Memphis, Tennessee

Xihui Zhang
University of North Alabama, Florence, Alabama

Jasbir Dhaliwal
University of Memphis, Memphis, Tennessee

ABSTRACT
Software testing has emerged as a distinct and critical component in software development. This paper argues that software testing should be conceptualized as a concurrent service throughout the software development process rather than being viewed as a sequential line of responsibility. Testing as a service has two key aspects: (1) a service to developers, and (2) a service to end users. This paper draws from the service quality and SERVQUAL literature to propose a structured measurement tool for testing as a service. Software quality is the most prominent outcome for TESTQUAL. The potential antecedents and outcomes of TESTQUAL are briefly discussed. The implications to research and practice of TESTQUAL are also discussed.

Keywords: Software testing, software development, service quality, software quality, SERVQUAL, TESTQUAL.

INTRODUCTION
Recent thinking about the nature of software architecture in organizations has been significantly influenced by models such as service-oriented architecture (SoA) and software-as-a-service (SaaS) (Goth, 2008). This is partly a result of the fact that it is often difficult for developers in disparate organizations to determine the interrelationships between or among software components because software is getting increasingly complex. At the
same time, today’s end users, who have become more technologically competent and sophisticated (Davis et al., 2009), have significantly higher expectations about software quality. Traditional testing practices, which have not changed significantly over the years, are unable to satisfy these emerging requirements to either software developers or end users. Therefore, it is imperative to reconsider the relationships between developers, testers, and end users from new perspectives.

One such new perspective we propose herein is that of “testing as a service,” which is in line with the general trend in information technology (IT) management to decompose the value of an IT unit in terms of a structured set of services rendered to both internal and external customers of an organization. In both academic and professional circles, this is often referred to as the “IT services” paradigm. This paradigm argues that organizations need to consider each value bundle provided by the corporate IT unit from a service provider and a service consumer relational perspective (Jia et al., 2008), which typically leads to a focus on measuring and managing the service quality in such relations.

Testing can be viewed as a service provided to software developers in relation to error minimization, validation, and verification (Bentley, 2005). Without the services provided by testers, the burden of error finding, verification, and validation would fall onto developers, who may not be equipped with needed training, capabilities, or technical resources to properly handle a comprehensive testing process. This is especially critical as it has been argued that developers are more technically oriented while testers are more process and business oriented (Cohen et al., 2004). Furthermore, a good developer will not necessarily make a good tester because distinct characteristics are needed (Pettichord, 2000). For instance, good testers are expected to have broad knowledge of many domains, whereas good developers are expected to have specialized knowledge of product internals (Pettichord, 2000). The role that testers play in software development is getting more and more critical because testers are the last arbiters who ensure that the software product is of high quality and also meets end users’ business needs.

This testing-as-a-service perspective is an important paradigm shift from the old perspective where testing is simply a technical quality assurance mechanism in software development. In this new paradigm, issues such as software usability, user satisfaction, mapping to business needs, and information systems coherence all become part of the testing agenda, leading to a more holistic role for testers beyond just technically testing of the code. To measure and manage the service rendered by testers, business and IT managers require a tool that allows for quantifying the quality of service provided. This paper attempts to fulfill this need by adapting an academic and practitioner validated service measurement tool - SERVQUAL - to the software testing context.
SOFTWARE DEVELOPMENT AND TESTING METHODOLOGY

Evolution of Software Development Methodology

In the 1950s, testing was not an indispensable activity in software development. The system development process then had only two steps - an analysis step followed by a coding step (Royce, 1970). There were no designated individuals that specialized in testing and professionally labeled as “testers.” The developers and end users often times carried out the testing activities to ensure the program could run as specified and also meet the business needs. As time went by, the life cycle models of software development emerged which divide the software development process into clear-cut phases, typically including phases such as analysis, design, coding, testing and implementation (McKeen, 1983). Depending on the workflows between the phases, the life cycle models can be categorized into sequential models (e.g., waterfall model, traditional V model, and PV model), progressive models (e.g., phased model), iterative models (e.g., spiral model), and agile models (e.g., eXtreme Programming) (Beck and Andres, 2005; Hill, 1996; Sommerville, 1996; Talby et al., 2006). In this paper, we use the waterfall model and agile model to illustrate the disparity of testing role in software development as they are respectively representative of the traditional and modern software development methods.

The waterfall model includes a set of sequential phases that flow downwards like a waterfall (Mahmood, 1987). Waterfall models have well-defined boundaries and responsibilities for the stakeholders, and are usually composed of consecutive stages where the outputs of each step become the inputs for the subsequent stages. However, with a waterfall model, testing can only get started after coding is complete. When defects are found, previous phases have to be revisited in order to fix them. This tends to cause a project to run over time and over budget (Tsai et al., 1997).

More modern agile software development model suggests an iterative and incremental approach to software development which requires full integration of testing and development (Talby et al., 2006). The intent is to produce high quality software in a timely manner, and also meeting the changing needs of the end users. Among many agile software development methods, eXtreme Programming (XP) is the most prominent one. XP shuffles through analysis, design, and implementation stages quickly (Dennis et al., 2005). A timebox that usually spans only a few weeks is used to ensure that enhanced software is ready to be delivered for the next iteration. Test-driven development (TDD), the notion of “writing test cases that then dictate or drive the further development of a class or piece of code” (Murphy, 2005, p. 3), is becoming an integral core practice of XP. TDD’s ability to change a class’s implementation and re-test it with very little effort and time makes it a powerful tool for meeting changing client requirements during the course of a project (Murphy, 2005). Beck and Andres (2005) note that XP encourages communication, simplicity, feedback, courage, and respect. The next two sections will elaborate
on how testers take a role in the traditional and modern software development practices, represented by waterfall models and agile models.

**Traditional Approaches of Testing**

In his 1979 book, *The Art of Software Testing*, Myers defined testing as “the process of executing a program with the intent of finding errors” (p. 16). The focus of software testing then was on “finding errors” by using some destruction-oriented means. Later, Hetzel (1984) defined testing as “any activity aimed at evaluating an attribute or capability of a program or system” and as “the measurement of software quality” (p. 7). Thus the focus of software testing had shifted to product evaluation and quality assurance. Gelperin and Hetzel (1988) claimed that testing is as old as coding, and its evolution is reflected in the changing definition of testing itself. Regardless what the focus of testing would be, testing had been long regarded as a post-hoc activity. In the traditional waterfall framework, testing usually focuses on technical issues to ensure that the code runs correctly and is compatible with other software components. In addition, testers will briefly check if the code complies with the user requirements and specifications. Typically, testing is performed by an independent group of testers after the software is completed by developers but before it is shipped to the end users. As a result, testing was separated from the development activities due to this sequential working mode of developers and testers (Whittaker, 2000). Figure 1 shows the workflow between developers, end users, and testers in the traditional life cycle of software development. There are two groups of arrows with opposite directions: the arrows pointing from left to right are the software products (or code) from Developers to Testers, and from Testers to End Users; the arrows in the other direction are feedback from End Users to Testers, and Testers to Developers.

This sequential development and testing process can lead to conflict between developers and testers (Cohen et al., 2004). Developers and testers have to constantly compete for scarce resources such as project time. Often times testing is postponed, and the planned testing time is reduced to meet the delivery schedule. The sequential software development process is shown in Figure 1.

![Figure 1. Traditional workflow among developers, testers, and end users](image-url)
development practice often results in compromises of product quality that are resulted from inadequate time being spent on testing the code, validating the requirements, or mapping specifications to business needs. Furthermore, in this out-of-date testing practice, testers are very much focused on the mechanical side of testing, which does not provide the level of testing quality required by today’s business environment. Therefore, a novel perspective of software testing that can encourage testers to take a more proactive and collaborative role in software development is very much needed.

**Testing as a Service**

Recent literature defines testing as a concurrent lifecycle process of engineering, using and maintaining testware in order to measure and improve the quality of the software being tested (Craig and Jaskiel, 2002). This definition of software testing emphasizes that testing should no longer be a post-hoc activity after coding and that testing should start taking preemptive actions to improve software quality. Pyhajarvi and Rautiainen (2004) recommend that testing should be included early in software development. Schach (1996) also suggests that testing should be performed throughout the software life cycle. Schach (1996) distinguishes two types of testing: execution-based testing and nonexecution-based testing. Execution-based testing techniques include black-box testing (i.e., testing to specification) and glass-box testing (i.e., testing to code). Nonexecution-based testing techniques include walkthroughs, inspections, and correctness proving (i.e., using a mathematical means to show a product meets its specification). He also predicts that “the future role of testing will be to prevent faults rather than to detect them” (p. 278). Vijay (2001) suggests that we should tap into the testing phase as early as possible if we want to cut down the overall cycle time of software development since the testing activity consumes approximately 40% of the whole project time and around 60–75% of the testing activities (test plan and test design) do not require any code. Empirical studies have shown that engaging testers earlier and throughout the software development process is beneficial to a project team's performance. Waligora and Coon (1996) demonstrated that project performance (cost and cycle time) is improved without sacrificing the quality of the end product by starting testing earlier in the development life cycle. A key argument for engaging testers earlier is to catch defects early in the software development process because fixing a defect detected in the later phases tends to be 10 to 100 times more costly (Tyran, 2006).

Bentley (2005) defines software testing as a process of verifying and validating so that a software application or program meets the business and technical requirements that guide its design and development, and works as expected. According to Bentley (2005), software testing has three main purposes: verification, validation, and defect finding. Verification is to confirm that the software meets its technical specifications, validation is to confirm that the software meets the business requirements, and defect finding is to find
and then to assist (developers) in fixing important defects. The “testing as a service” paradigm has the capability of conceptualizing all its activities as distinct services to pertinent constituencies—developers and end users, manifesting the three main purpose of testing explained by Bentley (2005). Software testing encompasses a wide spectrum of different activities, from testing a small piece of code (unit testing), validating a system (acceptance testing), to monitoring a network-centric, service-oriented application at runtime. There are different sets of objectives for testing at different levels (e.g., unit test, integration test, and system test) and at different stages of the software production cycle. It is increasingly becoming a prerequisite that testers work closely with both developers and end users throughout the software production life stages to fulfill these main purposes.

Figure 2 shows that testers provide key services to two distinct constituencies: service to developers (Service I) and service to end users (Service II). Service I is provided to developers and it pertains to error detection, verification, and validation of the code. For example, unit testing is a service provided by testers to developers to ensure that individual modules work right. Another aspect of this service is exemplified by code review and functional testing services which are to ensure that code written by developers is consistent with business requirements and software specifications. A third aspect of this service is provided by boundary testing, stress testing, and load testing that focus on ensuring that the code will work in the intended real world business context. Service II captures all categories of testing services that testers provide to ensure that end users will receive software that meets their needs. Specific examples of these services include requirements validation at early stages, alpha testing, beta testing, and usability testing at later stages of software development. In requirements validation, testers need to understand end users’ expectations about the software so that appropriate testing plans and strategies can be implemented to guard the users’ interests. Testers will analyze user requirements to construct test cases so that linkages are established between the two. User requirements validation is akin to services provided by testers in regard to business needs validation (functional requirements), software performance (performance requirements), and security (security requirements). Different applications with varying sets of requirements often require a diverse set of testing services. With Alpha testing service, testers engage users directly in system validation by allowing them to interact with the system in controlled testing environments. With Beta testing service, users are engaged through controlled software distribution processes tied to feedback loops that enable users to test software in real world conditions. Testers herein often act as facilitators in dynamic feedback loops designed to customize and localize software to specific user environments. Services I and II, as illustrated in Figure 2 and explained above, form the foundation of the TESTQUAL tool proposed in this paper as a holistic measurement of the testing service quality.

In Figure 2, the other three dotted lines illustrate the importance of the three stakeholders’ (developers, testers, and end users) combined and concurrent contributions.
The latest trend in software development methodologies is towards those that require testing personnel to be engaged much earlier in the development process, and testing is no longer the culminating activity of software development, leading to more frequent and sophisticated interaction among developers, testers, and end users (Cusumano and Selby, 1997). Although developers and testers are largely held accountable for software quality, more recent literature has demonstrated that involving end users in early stages of software development life cycle is beneficial to reduce uncertainty and to improve overall software quality (Hsu et al., 2008). This suggests an early interaction between or among developers, testers, and end users. In this new dynamics of requirement acquisition and confirmation, we argue that there is an urgent need to elevate testers to a more strategically centralized position in software development as they are the information integrators of end users’ high-level business requirements, the arbiters of developers’ intermediate software modules, and ultimately, the insurers of software products quality.

The conventional testing metrics primarily measure testers’ productivity and the outcome they deliver in technical terms (e.g., numbers of bugs found). As a result, the social aspect of testing service has largely been ignored. We advocate that testers take a service-provider role from two distinct yet complementary perspectives (to developers and to end users) and that testing as a service includes both technical and social aspects of testing service quality. The next section reviews the literature on SERVQUAL to provide a foundation for developing a structured scale for measuring the service quality of software testing.

**SERVICE QUALITY AND SERVQUAL**

There are two primary perspectives of service quality in the literature. One is the Nordic perspective (Grönroos, 1982, 1984), which uses global terms to define service quality in terms of functional quality and technical quality. The alternative perspective, promi-
ently used in North America, emphasizes service encounter characteristics that are per-
ceived by the service recipient, including tangibility, reliability, responsiveness, empathy,
and assurance (Parasuraman et al., 1988, 1991). SERVQUAL includes a widely-utilized
measurement instrument for assessing service recipients’ expectations and perceptions of
service deliveries. Nyeck et al. (2002) note that the SERVQUAL measurement tool re-
mains the most complete attempt to conceptualize and measure service quality. Our
present exploration of service quality in the software testing context is primarily based on
this SERVQUAL perspective.

Parasuraman et al. (1988, 1991) initially developed the SERVQUAL instrument to
measure the gap between customer expectations and services received, based on inter-
views and focus group meetings with managers and customers from large service compa-

dies in various industries. They found that customers basically used the same criteria in
evaluating service quality and that these criteria spanned virtually all types of service in
the consumer and retail sector. This multi-item instrument quantifies customers’ global
(as against transaction-specific) assessment of a supplier company’s service quality. The
scale measures the service quality delivered by the service personnel along five dimen-
sions: tangibles, reliability, responsiveness, assurance, and empathy. They define these
five dimensions as: (1) tangibles - physical facilities, equipment, and appearance of ser-
vice personnel, (2) reliability - ability of service personnel to perform the promised ser-
vice dependably and accurately, (3) responsiveness - willingness to help and provide
prompt service, (4) assurance - knowledge and courtesy of service personnel and their
ability to inspire trust and confidence, and (5) empathy - a caring orientation and the
provision of individualized attention to service recipients.

There has been some debate over the usefulness of the SERVQUAL’s gap measure
(perceived service quality minus expected service quality) pertaining to the conceptual
and empirical relevance of SERVQUAL versus performance only service scores (SERV-
PERF). Parasuraman et al. (1991) noted that the SERVPERF scores produced higher
adjusted $R^2$ values when compared to SERVQUAL’s gap scores for each of the five di-

densions. Van Dyke et al. (1997) also questioned the interpretation and operationaliza-
tion of the SERVQUAL expectation construct as well as the reliability and validity of
SERVQUAL dimensionality. In contrast, Pitt et al. (1995, 1997) demonstrated that the
problems of reliability of difference score calculations in SERVQUAL are not nearly as
serious as those described by Van Dyke et al. (1997), and suggested that the marginal
empirical benefit of a performance-based (SERVPERF) service quality measure does not
justify the loss of managerial diagnostic capabilities found in the SERVQUAL gap mea-
sure. Moreover, there are studies that advocate dual levels of service expectations. Para-
suraman et al. (1994) recommended two different comparison norms for service quality
assessment: one is the level of service a customer believes that can and should be deliv-
ered, which is “desired service,” and the other is the level of service that the customer
considers acceptable, which is “adequate service” or “minimum service.” The reason for separating the expected service into these two levels is that customer service expectations are characterized by a range of levels between desired service and minimum service, rather than by a single point. The band between desired service and minimum service is termed as the Zone of Tolerance (ZOT). Kettinger and Lee (2005) argued that the ZOT service approach can help provide a more valid basis for diagnosis and judgment concerning service deficiencies and service quality management.

SERVQUAL has been introduced to assess the quality of electronic services as well. Zeithaml et al. (2002) developed an e-SERVQUAL for measuring e-service quality through a three-stage process using exploratory focus groups and two phases of empirical data collection and analysis. The resulting e-SERVQUAL dimensions include efficiency, reliability, fulfillment, privacy, responsiveness, compensation, and contact. Gefen (2002) found that the five standard SERVQUAL dimensions could be reduced to three when conceptualized for the context of online service quality, which are (1) tangible, (2) a combined dimension of responsiveness, reliability, and assurance, and (3) empathy. Tangibles were found to be the most important dimension in increasing customer loyalty, while the combined “responsiveness, reliability, and assurance” dimension was considered most critical in increasing customer trust. Gefen (2002) also advocated that the e-SERVQUAL construct requires scale development that extends beyond merely adapting offline scales. Variations of the SERVQUAL scale such as WebQual (Liu and Arnett, 2000; Loiacono et al., 2007), SiteQual (Yoo and Donthu, 2001), and eTailQ (Wolfnbarge and Gilly, 2002) have also been developed and applied generally for the contexts of e-retailers providing service to online consumers.

TESTQUAL: ASPECTS, DEFINITIONS, AND SCALES

Modern system development methods have put testers under increasing pressure to demonstrate that their services are customer-focused and that continuous performance improvement is being delivered. Traditional software testing metrics have become inadequate for assessing the developers’ and end users’ expectations on both testers and software applications. TESTQUAL, as we proposed herein, is a structured scale to measure the quality of the service provided by testers in the context of software testing. It consists of two segments, each intending to measure service quality to developers and service quality to end users, respectively.

Service Quality to Developers

Even though many tools and automated methods can facilitate much of the testing tasks, the overall testing result ultimately depends on the interpersonal interactions of people producing the software (Cohen et al., 2004). The social interaction between developers
Table 1. TESTQUAL—Service quality to developers

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Item</th>
</tr>
</thead>
</table>
| Tangibles| • I believe the testers are equipped with the right hardware and software to perform the testing service.  
• The testing cases and results are well documented. |
| Reliability| • I believe the testing service performed by the tester(s) is reliable.  
• I believe the tester(s) can perform software testing correctly.  
• I trust the tester(s) to deliver the kind of service I need. |
| Responsiveness| • The tester responds to my needs.  
• In case of any problem, the tester will give me prompt service.  
• The tester will address any of my concerns regarding to the software product. |
| Assurance| • I feel confident about the testing service provided by the tester(s).  
• The tester(s) have answers to all my questions about the testing. |
| Rapport| • The tester(s) can address my various and specific needs.  
• I built a good relationship with the tester(s) through effective communication.  
• The tester(s) and I can resolve conflicts easily. |

and testers becomes evidently necessary in modern testing practice. The measurement for service quality to developers mainly draws upon the SERVQUAL literature, which primarily taps the human aspect of service quality. Our proposed TESTQUAL adopted five aspects of SERVQUAL to measure the testers’ service to developers, including tangibles, reliability, responsiveness, assurance, and rapport, as listed in Table 1.

The definition of Tangibles, distinctive from the one by Parasurman et al. (1988, 1991), refers to a set of attributes about the physical testing environment, testing deliverables, and documentation. Definitions for Reliability, Responsiveness, and Assurance are directly adapted from the original SERVQUAL. Reliability refers to the ability of testers to perform the promised service dependably and accurately. Responsiveness refers to testers’ willingness to help the developers in a timely manner. Assurance refers to knowledge and skills of testers and their ability to inspire trust and confidence of the developers.

The Rapport aspect, similar to Empathy in the original SERVQUAL, refers to caring and individualized attention to the developers. Since developers are focused on the technical nuances of software design and testers are focused on user requirements (Cohen et al., 2004), they can be at odds with each other, and conflict between developers and testers can arise. Conflict can adversely affect work process, quality of work, and quality of work life. Therefore, rapport-building is key to the quality of service rendered to developers.
**Service Quality to End Users**

This aspect of service relates to perception and/or confirming the expectations of users regarding software quality. This can be called an “indirect” testing service since it is performed through the medium of the software product with which the users interact. The dimensions that are investigated under this section pertain to three individual characteristics of service quality: tangibles, agency oversight, and user satisfaction. The items are listed in Table 2.

Tangibles refer to the set of attributes about the appearance of software that users come in direct contact with, as well as the actual software deliverables and documentation.

Testers are agents of end users to control software quality. They need to guarantee that software perform consistently, accurately, and securely according to the expectation of end users. Agency oversight pertains to the testers’ overall responsibility to ensure that all attributes related to and characteristics of the software and its use are of high quality, conform to the requirements of the users, and meet the business needs.

User satisfaction is the users’ holistic view of software quality that is assured by testers. Alpha testing and Beta testing put the users in direct contact with the software and allow their satisfaction to be measured. This is the ultimate test of software before it is released to production. The testers have a paramount role during this stage, and the service provided by them deals with final error checking and any concluding validation and verification issues.

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Item</th>
</tr>
</thead>
</table>
| **Tangibles**     | • The interface was well designed so that it is easy to use.  
• The software was well documented so that I can pursue help on my own easily.  
• The software’s interface is pleasing aesthetically. |
| **Agency Oversight** | • The testers did a good job in terms of catching bugs.  
• The testers ensured that the overall quality of the software was improved and that it meets my needs.  
• Testing ensured that the software meets the business needs of the organization. |
| **User Satisfaction** | • I am satisfied with performance of the software that is warranted by testers.  
• Testers assure that the software can meet my needs.  
• Overall, I am happy about quality of the testing service. |

Table 2. TESTQUAL—Service quality to end users
SOFTWARE QUALITY AND SCALES

Software Quality

The quality of software developed is a principal concern of every stakeholder in a software development project. Software quality should be interpreted in light of the concept of “purpose of use,” considering both internal attributes (product characteristics) and the external attributes (aim of use) (Issac et al., 2003). Therefore, assessment of software quality should take into account multiple viewpoints. From the developers’ perspective, software quality can be defined by the conformity to functional and performance requirements that are explicitly fixed, conformity to development standards that are explicitly documented, and conformity to implied characteristics expected of all software developed in a professional manner (Pressman, 1988). From the users’ perspective, software quality can be defined with respect to all the properties that satisfy correctly and efficiently the present and future needs of software buyers and users (Issac et al., 2003). These perspectives can be economic as represented by managers, be both technical and social as represented by developers, and be utility-like as represented by users.

The importance of customer-perceived quality is recognized in many fields, especially in manufacturing, marketing, and service organizations; however, there exists no rigorous framework for measuring customer perceptions of software quality. Issac et al. (2003) proposed a conceptual framework for customer-perceived software quality, which is based on the Total Quality Management (TQM) framework and consists of six dimensions: product quality characteristics, operational effectiveness, client focus, process quality, infrastructure and facilities, and employee competence.

Software quality does not improve unless it is measured and managed (Reichheld and Sasser, 1990). Measures of software quality are often difficult to articulate and no single measure is adequate by itself. ISO 9126, developed by International Standards Organization (ISO), is an international standard for the evaluation of software quality. Empirical studies on software quality, although limited in number, almost all agree that functionality, reliability, usability, efficiency, maintainability, and portability are the six major dimensions of software quality (Barki and Hartwick, 2001; Issac et al., 2003; Ortega et al., 2003).

Software Quality: Dimensions and Scales

Software products are expected to demonstrate all the attributes that can measure up to pre-established standards, and testers have a key service responsibility in this regard. Many of these attributes pertain to technical standards that have to be met for certification to be awarded (e.g., ISO 9126). In specific industries where software failure may mean life or death consequences, certified testers often times have to confirm that imple-
mented software products meet strict thresholds for reliability, security, and stability. As aforementioned, the six major dimensions of software quality include functionality, reliability, usability, efficiency, maintainability, and portability (Barki and Hartwick, 2001; Issac et al. 2003; Ortega et al., 2003), as listed in Table 3.

Functionality is a direct reflection of the ability of the software to perform the intended tasks for which it is developed. Testers ensure that the capabilities of the software are adequate to perform the intended tasks that the software is supposed to support.

Reliability investigates the capability of the software to maintain its performance level over different conditions and at different times. Functionality looks at performance, whereas reliability looks at sustainability of performance.

Usability is the degree of effort required for learning and using the software by its intended users. The software is checked to see if typical users are able to properly operate the software. The service provided here has two parts because it looks at the software itself, and also takes into account the users’ interaction with the software.

Efficiency refers to the performance and resource use of the software in a given certain circumstance. Testers verify the responsiveness of the software when answering requests from the users, as well as the responsiveness of the interface. They also check the new system’s resource requirements and load footprint. This ensures that the new system will function smoothly and efficiently in various operating environments.

Table 3. Software quality

<table>
<thead>
<tr>
<th>Aspect</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>• It is easy to tell whether the software is functioning correctly.</td>
</tr>
<tr>
<td></td>
<td>• The software can recover from errors, accidents, and intrusions.</td>
</tr>
<tr>
<td></td>
<td>• The software can maintain data security and integrity.</td>
</tr>
<tr>
<td>Reliability</td>
<td>• The software is reliable (it is always up and running, runs without</td>
</tr>
<tr>
<td></td>
<td>errors, and does what it is supposed to do).</td>
</tr>
<tr>
<td></td>
<td>• The software has the ability to prevent (or minimize) damage,</td>
</tr>
<tr>
<td></td>
<td>when an error is encountered during the execution of a program.</td>
</tr>
<tr>
<td>Usability</td>
<td>• The software is easy to use.</td>
</tr>
<tr>
<td></td>
<td>• The software is easy to learn.</td>
</tr>
<tr>
<td>Efficiency</td>
<td>• The software performs its functions quickly.</td>
</tr>
<tr>
<td></td>
<td>• The software does not overtax system resources during its operation.</td>
</tr>
<tr>
<td>Maintainability</td>
<td>• The software can easily be modified to meet changing user</td>
</tr>
<tr>
<td></td>
<td>requirements.</td>
</tr>
<tr>
<td></td>
<td>• The software can easily be adapted to a new technical or organiz-</td>
</tr>
<tr>
<td></td>
<td>tional environment.</td>
</tr>
<tr>
<td></td>
<td>• The software is easy to maintain.</td>
</tr>
<tr>
<td>Portability</td>
<td>• The software can easily be operated on different operating systems.</td>
</tr>
<tr>
<td></td>
<td>• The software can easily be adapted to organizational environment.</td>
</tr>
</tbody>
</table>
Maintainability pertains to the modularity of software design and the effort required to modify or upgrade the software with additions, changes, or improvements.

Portability is the degree to which the software is transferable to multiple environments and configurations. Testers provide a service here that pertains to ensuring that, given future business and technological development, the software developed will allow easy installation/reinstallation, can be replaced or transferred with minimum effort, and can easily adapt to new environments.

TESTQUAL: ANTECEDENTS AND OUTCOMES

The focus of this paper is to propose the two key dimensions of TESTQUAL: testing service that is perceived by developers and testing service that is perceived by end users. The very next step ought to verify whether the proposed TESTQUAL scale can demonstrate reliability and validity by conducting the questionnaire consisting of the items that are listed in Tables 1 and 2. As prior research (eg., Devaraj et al., 2001) advocates that SERVQUAL is positively related to product quality, we argue that TESTQUAL is a significantly positive factor in determining software quality in the software development context. Linking TESTQUAL with other constructs is necessary to check the nomological validity of this newly adapted scale. Figure 3 illustrates the roadmap to expand the linkage between TESTQUAL and Software Quality, to other organizational level, proj-

Figure 3. TESTQUAL: Antecedents and outcomes
ect-level, and individual level factors that can affect the stakeholders’ perception on the provision of testers’ service, and the outcome that derived from TESTQUAL besides Software Quality.

**Antecedents of TESTQUAL**

The antecedent of TESTQUAL can be categorized into three groups: organizational level (service culture/climate and resource allocation), project level (methodology, scope of testing, communication and leadership), and individual level factors that include testers’ capabilities, expertise, and personalities.

**Organizational Level.** One of the themes in service quality suggests that organizations must create and maintain a service climate in order for employees to effectively deliver service (Schneider, 1998). Management can help promoting such a service-oriented culture/climate that emphasizes that testing organization is a service provider and developers and end users are their internal customers. Resources (e.g., time and budget) allocated for training testers and testing activities also impact the service quality that testers can deliver. Experienced testers who have received sufficient training for strengthening both testing expertise and business understanding can be more effective in provision of testing. Indeed, time is always a scarce resource in project management and critical for effective testing.

**Project Level.** System Development Life Cycle (SDLC) is the traditional way of software development, and execution-based testing is usually conducted with the primary objective of finding defects in the code after the design phase under this methodology. The problems that are found at this stage of software development are a lot more costly to fix than if the problems are found in the requirement phase. Therefore, involving testers as early as possible within iterative development life cycles is a prerequisite to conquer the disadvantages of traditional SDLC. Modern system development methodology based on prototyping such as RAD, JAD, and eXtreme Programming can provide the testers with the necessary working environment and job structure to provide quality testing service. Project manager should use a system development methodology that incorporates and considers testing from project inception through project completion. The Scope of testing comprises the responsibilities that testing has in relation to what, when, and who to test various parts of the software developed. Unit testing, integration testing, alpha testing, beta testing, requirement testing are some of the few types of tests that testers can run on the software. Depending on where tester responsibility starts and finishes, the scope of testing may be more or less extensive, and the service that testers need to provide varies. When the scope of testing enlarges, the testing service tends to be more of strategic effect on the whole processes of the system development. Lack of leadership has been a problem in system testing, resulting in an out-of-control testing environment.
(Frolick and Janz, 1998). With more management support and involvement in testing practice, the management’s leadership style is also an important antecedent to the quality of deliverables of the project constituencies. Participative or authoritarian leadership leads to different outcomes, where a participative style is more amenable to infusing a proactive service-oriented mindset to the testers, by allowing them to have an effective say in the various milestones of development processes. One of the key findings of the project management is that the communication flow structure of business processes was a particularly strong determinant of most of the quality and productivity problems, more so than either the activity flow or the material flow structure of the business processes (Kock, 2006). The determining role of communication flow applies to testing practice as well. The communication flow between developers and testers should be free of barriers, thus horizontal communication is preferred to vertical communication, which does not require the superiors’ approval. Communication media (e.g., face-to-face and email) are also important. As Cohen et al. (2004) suggested, the physical co-location of developers and testers can help the testing service be provided more effectively and efficiently.

**Individual Level.** Testers’ personal characteristics that include capabilities, experience, and personalities can also determine the service they provide. Capabilities and experience of testers are obvious determinants for superior testing service, so herein we will focus on personalities of testers in relation to the TESTQUAL. Chilton et al. (2005) note that the personalities types are an important factor in successful team performance and suggest that complex task in software development needs a balanced team opposite personality types. Particularly, the fit between the developers’ preferred cognitive style and the developers’ perception of the cognitive style required for the existing job environment is instrumental in increasing their job performance (Chilton et al. 2005). Accordingly, we expect that the existence of fit for developers and end users’ preferred cognitive style of testers and their perception of cognitive style of testers can positively impact TESTQUAL as well.

**Outcomes of TESTQUAL**

Software quality will be the prominent outcome that derives from higher quality testing service that is provided to the development team. Testing is just like Quality Assurance (QA) service that monitors the whole software development process. The functionality, reliability, usability, efficiency, maintainability, and portability of software can be greatly enhanced by better testing service.

Development time will be dramatically reduced when testers can coordinate with developers and end users, and provide adequate and even superior service at proper time during the software development life cycle. As noted by many testing researcher such as Frolick and Janz (1998), high quality software testing service, if provided from project
inception, can not only increase software quality, but also decrease the overall system development cycle time.

Based on the Job Characteristic Model (JCM) (Hackman and Lawler, 1971; Hackman and Oldman, 1975), feedback in the job that a worker performs and felt meaningfulness of the job can be effective in increasing his/her job satisfaction. Testing activity as a service provided to the developers can be viewed as constant feedback from testers. Testers’ focus on the user requirements can reinforce the developers’ felt meaningfulness of their software development job tasks, as developers often times are heavily oriented toward the technical side of software development, and the fundamental goal of satisfying users’ need and being aligned with organizational objectives is ignored. So developers’ job satisfaction can be improved because of the intensive feedback from testers and enhanced felt meaningfulness toward the job they perform. On the other hand, users’ requests on the desired features of a new application are fulfilled, which is facilitated through the user-centered testing service. Therefore, end users’ appraisal of their job experience can be elevated to a more pleasurable emotional state. End users’ job satisfaction could also be partially derived from better software product quality and reduced software development cycle time.

DISCUSSION AND CONCLUSION

In this paper, we proposed a new concept, i.e., testing as a service, and we also provided a measurement tool, i.e., TESTQUAL. The new concept and the measurement tool have important implications to both research and practice. In terms of research, the new “testing as a service” perspective will lead to better research focus that is aligned better with the modern software development methodologies. For instance, prior research on software testing has been mainly focused on “bug counts,” with this new perspective, future research on software testing should focus on the service quality to developers and end users delivered by testers. The proposed scale, TESTQUAL, will equip interested researchers and practitioners with a tool to quantify the quality of such services delivered by testers. In terms of practice, the new “testing as a service” perspective has three major implications. First, the new perspective posits testers as the service providers, which is different from the paradigm where testing is a simple technical verification and validation function that does not tie business needs with technical implementations. This new “testing as a service” perspective, in our contention, has the potential to improve software development processes by allowing structured and implementable input from all development stakeholders. It helps developers, testers, and end users recognize that they are highly interdependent on one another and can really be in a win-win-win situation. Second, the new perspective enables testers to have a clear sense of expectations and perceptions of other stakeholders such as developers and end users. As a result, testers are more likely to provide adequate and even superior services to their “customers,” thus the project team as a whole is one step closer to successfully completing a software development project. Third, the new perspec-
tive will remind IT managers and technical leads that they must not only support software testing with resources such as funding, but also make efforts to build and promote a “testers-as-service-providers” climate in software development project teams.

TESTQUAL is an implementable tool that will allow IT and business managers to quantify the level of service quality offered by their testing function. It can be administered to the receivers of testing services at multiple intervals in the development processes. Once the perceived level of service gap, or ZOT, of service quality is quantified, managers can identify areas of deficiency. In particular, the proposed TESTQUAL items can provide richer information than perception-only scores, as it could be expectation-minus-perception or ZOT scores. Accordingly, managerial corrective measurements can be taken to address the testing service deficiencies or failures. TESTQUAL can also assist testers in identifying cost-effective ways of closing service quality gaps and in prioritizing which gaps to focus on, given the constraint of scarce resources. It can also serve as a structured guideline for training newly hired testers in terms of the technical aspects of delivering quality testing service and also make them aware of the social aspects of interaction with other important stakeholders such as developers and end users in software development projects.

REFERENCES


