5-2008

Improving Software Quality Through the Use of Statistics: An Initial Approach

Juan M. Gómez Reynoso
Universidad Autónoma de Aguascalientes, jmgr@correo.uaa.mx

Mónica R. Brizuela Sandoval
Universidad Autónoma de Aguascalientes, mbrizuel@correo.uaa.mx

51F. Improving Software Quality Through the Use of Statistics: An Initial Approach

Juan M. Gómez Reynoso,
Universidad Autónoma de Aguascalientes
jmgr@correo.uaa.mx

Mónica R. Brizuela Sandoval
Universidad Autónoma de Aguascalientes
mbrizuel@correo.uaa.mx

Abstract

Information systems development is a very important activity that is performed continuously in Information Systems departments. We can say that quality is a complex measurement of a product or service that people demands. However, quality is a measurement that is composed by a set of aspects. Quality measurement can be performed in concrete or abstract form. Software quality is a very important issue that developers must address properly, but a lot has to do with abstract aspects of it nonetheless. We proposed an approach that could reduce the abstractness of software quality measurement. In order to prove it, we conducted a study with encouraging results. We found that end-user participation in the evaluation IS quality can be improved.

Keywords

Information Systems quality, systems development, end-user involvement.

1. Software Development

Information Systems (IS) development is one of the most important activity that is performed continuously in information systems departments. Organizations face continued pressure from the environment to stay or become competitive. Pressure comes from sources such as: government, industry, and competitors. The development and maintenance of IS helps organizations to become -or continue being- competitive in their industry. Researchers study several issues that affects the IS field. Walls et al. (1992) believe that one of the concerns for researchers in the IS area is the design of systems. Nevertheless, the effectively development of systems is a research topic in IS (Markus, Majchrzak, & Gasser, 2002). In addition, "the development of information systems is a creative effort that involves the expertise, insights, and skills on many individuals" (Tiwana & McLean, 2005, p. 14). A very important part of the every day effort in IT departments is to development and maintenance of information systems.

The increasing demands for new software developments and/or modify the existing systems puts pressure for developers. In order to attend such demand, several approaches for software development have been created. One of them is end-user development. However, the dependability of these software developments is suspected (Burnett, Cook, & Rothermel, 2004). No user is willing to use any software that has poor quality because this might results in losing and/or damaging very important information. Therefore, IS development efforts have to be performed with quality in mind.

Zayaraz et al. (2005) argue that software quality has several views but the overall quality can be expressed by a combination of the different views. But, quality software should include the

measurement of attributes that are perceived as indispensable by users. Such measurement has to be in concordance with high standards and error-free as possible.

It is known that most of a software cost is due to maintenance activities (Pressman, 2005; Sommerville, 2006). One way to reduce maintenance is to have high-quality software. Software quality assurance helps to reduce the cost of software maintenance and satisfy users' requirements (Amasaki, Yoshitomi, Mizuno, Takagi, & Kikuno, 2005). However, software quality is improved by reducing the number of faults by testing sufficiently (Amasaki et al., 2005). Thus, in order to increase quality, software has to be tested both during development efforts and after finishing the product.

The majority of software projects failures are because system engineering shortfalls such as lack of user input, incomplete requirements, among others (Boehm, 2006). Most of software testing is performed by developers without end-user participation. Further, often developers do not understand or clarify completely users' requirements. Thus, it is extremely important that IS's users participate in software evaluation through a structured approach.

In conclusion, it is important to not only develop software that has been tested using a structured testing strategy but also testing it by using an empirical approach.

2. Information Systems Testing

There are many risks of IS failures. Some failures are creditable to developers, some because end-users did not provide accurate or complete requirements, and some because of poor interaction between both. One of the most important risks of IS projects failures is because developers are not familiar with the business application that is being developed (Dennis, 2002). Who are really familiar to it are end-users. Thus, it is important that developers acquire the knowledge about the application or involve end-users in all phases of the project so that a successful IS can be developed.

Ebert and Baisc (2001) argue that "in order to achieve software quality, it must be developed in an organized form by using defined methods and techniques and applying them consistently. In order to achieve an indication of software quality, software must be subjected to measurement. This is accomplished through the use of metrics and statistical evaluation techniques that relate specific quantified product requirements to some attributes of quality" (p. 4,5).

Lincke and Löwe (2006) mention that software quality is defined in the ISO/IEC 9126 standard, which describes internal and external software qualities and their connection to attributes of software. Such attributes are evaluated by people, but not based on a specific metric or scale. Most of the time, people assign a specific quality value to an attribute based on their own preferences, not in a particular scale.

Most of IS testing is conducted while developing efforts are being conducted (Pressman, 2005; Sommerville, 2006). Further, literature advice a set of tests to be conducted such as: unit testing, black box testing, white box testing, system testing, among others. However, such tests are mainly conducted by developers and very few – or none at all – are performed involving end-users.

We believe that end-users participation in testing increases IS' quality. At doing so, deviations or incomplete requirements can be discovered; overseen errors can be discovered and corrected. Thus, this would increase the chances of end-users acceptance.

3. Methodology

We believe that following a specific set of steps as a structured approach increases the overall quality of an IS. We conducted a study to explore whether such approach would increase IS's quality. The study was conducted as follows: select a group of participants that have experience, knowledge and abilities for IS development assigned as developers or testers; select a set of requirements for a particular IS application, develop two versions of the IS, evaluate each version using descriptive statistics and refine as needed.

3.1 Participants

A total of twenty-three students in their ninth semester of a Computer Systems bachelor program enrolled in a Software Engineering course participated in the study as Developers. They received the specifications for a software IS project from the researchers, while twenty two students their eight semester from the same program enrolled in a Systems Development Methodologies course evaluated all the IS developed (Testers). Developers and Testers were free to drop from the study at any time. However, all of them completed the study, which lasted a full semester.

3.2 Questionnaire Development

In order to evaluate systems' quality, a set of seven attributes were defined. These attributes were defined through a focus group conducted with four people that are in charge of customer service in a local software development organization. They argued that these are the most common issues that end-users report problems in their systems. Such attributes are as follow: easy use for data entering, system's ability for detecting and handling unexpected errors, effectiveness of on line help, data validation, interfaces' attractiveness and aesthetics, font use throughout the system, and colors used in the system.

The questionnaire used at least three questions for each attribute. Questions were answered using a 7 point Likert-type scale, where 1 was the highest quality assigned. In addition, Testers were free to add comments and/or suggestions in any question they would think was necessary. Nevertheless, it was required that Testers report errors and faults detected using the free-text space provided.

3.3. Pilot Test

In order to create a good evaluation instrument, a pilot test was conducted. Twenty-four students in their eighth semester of a bachelor degree in Computer Systems participated in the pilot test. Time was recorded from the beginning of the exam so that we intended to assure that they run the system provided and read the questionnaire. The first person finished thirty-one minutes after the starting time, and the last person finished forty-five minutes after the starting time. Figure 1 shows participants' evaluations distribution for each question.

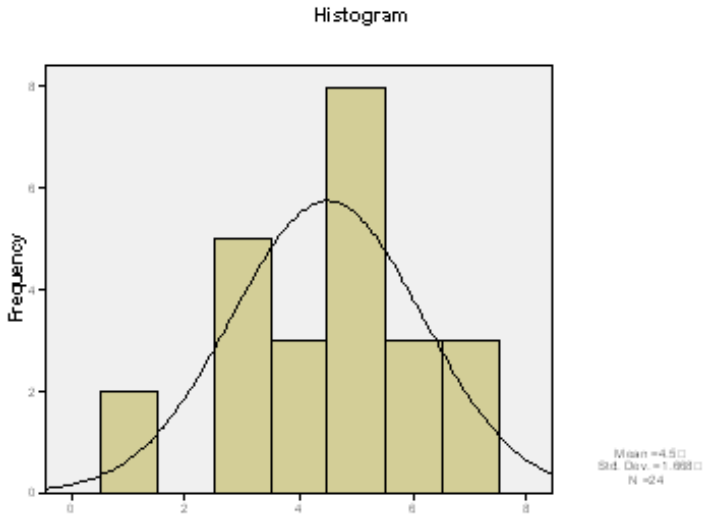

Figure 1. Instrument pilot test histogram

Table 1 shows descriptive statistics for the pilot test. Tabachnik and Fidell (Tabachnick, 1996) recommend that a sample's skewness value should not be beyond two times standard errors for skewness (SES). The calculated skewness value (-.489) for the answer is below the SES (two times ±0.472). Thus, it is assumed that the skewness is within the expected range of chance fluctuations; thus, has no significant skewness problem. In addition, a sample's kurtosis value should not exceed two times the standard errors of kurtosis (SEK). The calculated kurtosis value (-.091) is below SEK (two times ±0.918). Hence, it is assumed that the kurtosis is within the expected range of chance fluctuations. Based on both evaluations, it can be said that the instrument results exhibit a normal distribution. Thus, the results of the pilot test indicate that the instrument is suitable for the research.

Statistics

Value asigned

N	Valid	24
	Missing	0
Mean		4.50
Median		5.00
Mode		5
Std. Deviation		1.668
Variance		2.783
Skewness		-.460
Std. Error of Skewness		.472
Kurtosis		-.091
Std. Error of Kurtosis		.918
Range		6
Minimum		1
Maximum		7

Table 1. Instrument pilot test statistics

3.4. Research Design

We conducted our study as Figure 2 shows. Participants involvement for each step is noted in the same figure. Developers received the IS's specifications from the researchers in written form. The IS developed consisted in a database system that allows a bachelor degree coordinator to

evaluate their personnel in six different aspects related to their work each week. In addition, the IS included data analysis and charts construction features. Developers studied the specifications for a week. After that, several group meetings were conducted so that any issues regarding the IS to-be-developed were resolved. No private discussions or sessions were allowed so that all developers have the exact same information. One special requirement was that the software cannot be linked in any way to the Developers so that testers can make an unbiased evaluation. Only the researchers knew who was responsible for each software solution. After all the issues regarding IS's specifications were resolved, Developers got six weeks to develop and deliver the system to the researcher.

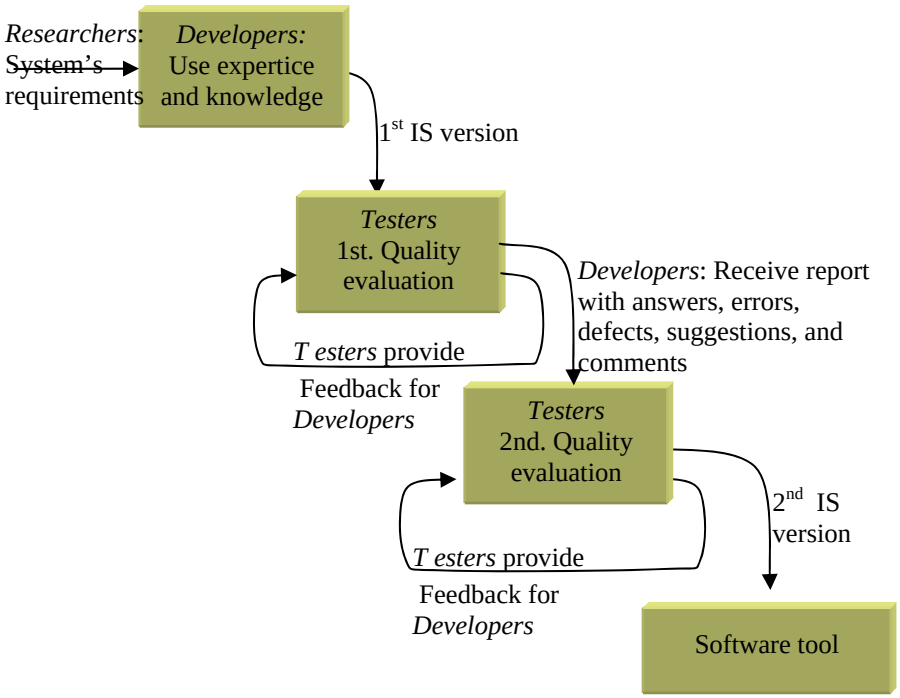

Figure 2. Development and Evaluation Process Proposed

Testers received a CD that has a copy of the first version of each system and kept them for three weeks. They evaluated each system and answered a questionnaire for each system.

After Testers evaluated the IS developed, they turned in their reports to the researchers. Then, the researchers created a package for each Developer contained all the evaluations for their own IS. With these reports, each Developer addressed all errors and faults reported by Testers as well as addressed all comments and suggestions received. Then, they performed an descriptive statistical analysis of the answers in the questionnaires. These analyses allowed them to assess the quality achieved to this point so that Developers were aware of the IS' overall quality as perceived by end-users.

4. Software Quality Measurement Results

4.1. Descriptive statistics

All Developers evaluations were collected and analyzed using descriptive statistics. Figure 2 shows two examples of histograms for two questions. Both example histograms show that evaluations were less disperse (comparing left side histograms with right side histograms), quality mean was improved and evaluations were more concentrated in the highest values from 2nd evaluation comparing with 1st evaluation. In addition, some low-quality values assigned by users in the 1st evaluation were not present anymore in 2nd evaluation, which means that Testers considered that quality for that particular IS attribute was improved. We believe that quality was improved because of the effects of how quality evaluation was approached.

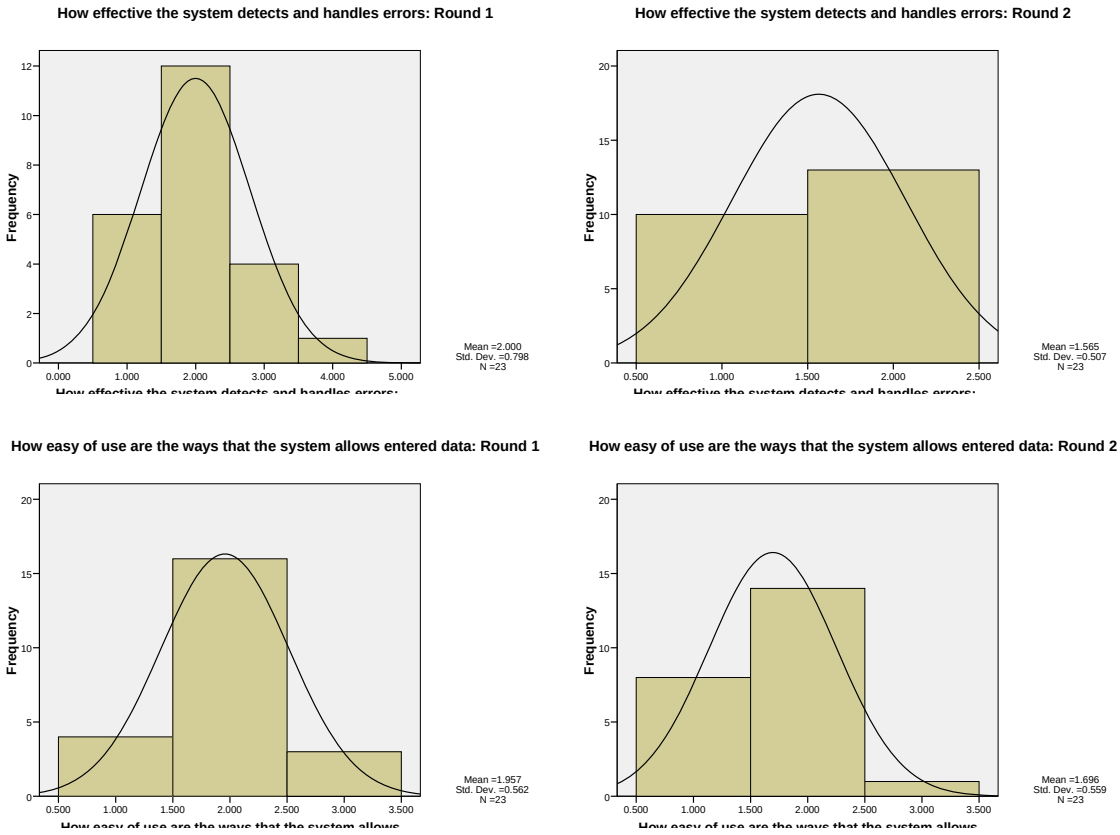

Figure 2. Example of quality evaluation by Testers

4.2. t-test Results

In order to evaluate whether quality was improved though the proposed methodology, a t-test was conducted. Results in Table 1 shows that, in all cases, the developed systems quality was improved through the evaluation method applied. In addition, all cases results are highly significant ($p<.01$).

Paired Samples Test

		Paired Differences					t	df	Sig. (2-tailed)
		Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference Lower	Upper			
Pair 1	How easy of use are the ways that the system allows entered data: Round 1 - How easy of use are the ways that the system allows entered data: Round 2	1.154899	.521215	.108681	.929509	1.380289	10.627	22	.000
Pair 2	How effective the system detects and handles errors: Round 1 - How effective the system detects and handles errors: Round 2	1.366571	.702984	.146582	1.062577	1.670564	9.323	22	.000
Pair 3	How the system provides help so that users are able to perform their tasks: Round 1 - How the system provides help so that users are able to perform their tasks: Round 2	1.496467	1.200273	.250274	.977430	2.015504	5.979	22	.000
Pair 4	How effective the system validates all the data input by users: Round 1 - How effective the system validates all the data input by users: Round 2	1.132122	.542516	.113122	.897520	1.366724	10.008	22	.000
Pair 5	How attractive and ahestetic are the interfaces of the system: Round1 - How attractive and ahestetic are the interfaces of the system: Round 2	.926728	.631593	.131696	.653606	1.199849	7.037	22	.000
Pair 6	How visible, consistent, and ahestetic are the fonts used in the system: Round 1 - How visible, consistent, and ahestetic are the fonts used in the system: Round 2	.963433	.773011	.161184	.629158	1.297708	5.977	22	.000
Pair 7	How effective and ahestetic are the color used in the system: Round 1 - How effective and ahestetic are the color used in the system: Round 2	.867102	.648936	.135313	.586481	1.147723	6.408	22	.000

Table 1. t-test results

These results show that quality was improved, which means that using a structured approach really helps developers to reduce issues that were overseen or ignored by them during early stages of the project. The most expensive IS is the one that is bought and not used by end-users, which results in a cost rather than an investment. Thus, this could result in an IS that complies with end-users expectations and needs. Moreover, user involvement in the final development stages helps developers to minimize any problems related to user-acceptance and IS usage.

5. Conclusions

Evidence shows that most of the time evaluated attributes' quality was improved through the approach we used. This could be because end-users were included as a very important part of the overall project. Thus, they knew exactly what were the IS requirements and evaluate what was offered by developers. Then, issues that were discovered by users were feedback to Developers so that can be addressed properly.

We argue that end-user involvement in IS development efforts has to be included during testing phases, not only during information gathering and requirements validation phases. Such involvement helps developers to create more user-oriented IS, which could reduce IS maintenance and their associated costs.

5.1. Limitations of the Proposed Study

Results show that our proposed structured approach for quality improvement through a structured testing delivers very good results. However, results has to be taken with caution since this is an ongoing research and this is the initial study that allowed us to understand the problem,

thus findings might be only true for this initial study. In addition, each developer created a "version" of the same system. This could affect the results maybe some less advanced developers requested help from other more skilled and affecting the IS evaluated. This could introduce an effect that we did not consider, and in consequence, did not measure or controlled. We believe that outcomes cannot be generalized.

In addition, since we applied the study with students that were enrolled in courses that a project was required this could lead that Developers and Testers put special interest and effort in their participation. With a group of professional developers results could be different.

There might be additional effects that we did not identify during our study that could affect outcomes.

5.2 Areas for Additional Research

The same study could be conducted but Developers should be free to develop their own IS as well as to create their own measuring instrument. This could provide more insights about the effectiveness of our way of IS quality improvement approach proposed. Conduct the same study with a group of professional developers could provide more insights about our research. Thus, this could result in an improved approach.

References

Amasaki, S., Yoshitomi, T., Mizuno, O., Takagi, Y., & Kikuno, T. (2005). A New Challenge for Applying Time Series Metrics Data to Software Quality Estimation. *Software Quality Journal, 13*(2), 177-193.

Boehm, B. (2006). *A View of 21st Century Software Engineering.* Paper presented at the ICSE'06, Shangai, China.

Burnett, M., Cook, C., & Rothermel, G. (2004). End-user software engineering. *Communications of the ACM, 47*(9), 53-58.

Dennis, A., Wixom, B. H., & Tegarden, D. (2002). *Systems Analysis and Design. An Object-Oriented Approach with UML*. New York: John Wiley and Sons.

Ebert, C., & Baisch, E. (2001). Metrics for identifying critical components in software projects. In *Handbook of Software Engineering and Knowledge Engineering*.

Lincke, R., & Löwe, L. (2006, July 3rd. 2006). *Validation of a Standard- and Metric-Based Software Quality Model.* Paper presented at the 10th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, Nantes, France.

Markus, L., Majchrzak, A., & Gasser, L. (2002). A Design Theory for Systems that Support Emergent Knowledge Processes. *MIS Quarterly, 26*(3), 179-212.

Pressman, R. S. (2005). *Software Engineering: A Practitioner's Approach, 6/e*. New York, N.Y.: Mc Graw-Hill.

Sommerville, I. (2006). *Software Engineering. 8/e*. London, U.K.,: Addison-Wesley.

Tabachnick, B. G., & Fidell, L.S. (1996). *Using multivariate statistics (3rd ed.)*. New York, N.Y.: Harper Collins College Publishers.

Tiwana, A., & McLean, E. R. (2005). Expertise Integration in Information Systems Development. *Journal of Management Information Systems, 22*(1), 13-43.

Walls, J. G., Widmeyer, G. R., & El Sawy, O. A. (1992). Building an Information System Design Theory for Vigilant EIS. *Information Systems Research, 3*, 36-58.

Zayaraz, G., Thambidurai, P., Srinivasan, M., & Rodrigues, P. (2005). Software quality assurance through COSMIC FFP. *ACM SIGSOFT Software Engineering Notes, 30*(5), 5 pages.