DYNAMIC SOFTWARE QUALITY ASSURANCE*

D. Kardoš

Czech University of Life Sciences, Faculty of Economics and Management, Department of Information Engineering, Prague, Czech Republic

Basic principles of the dynamic software quality assurance are explained in this paper. The current assume the quality testing in discrete time intervals. After quality evaluation the next time of quality test shall be planned. Continuous quality monitoring between two tests is not required. Author therefore suggests implementation sub-characteristic "users monitoring and cooperation" defined as the capability of the of-the-shelf software product to provide the level of the ability of the software to be centrally monitored by the maintenance center and communicate with other copies of the same software product on private network or Internet. Thereby it creates a dynamic model from primarily static one.

dynamic software quality assurance; software product quality; measures for users monitoring and cooperation

INTRODUCTION

The software product quality is defined as level of satisfaction of specified or implied requirements by the inherent set of inherent attributes of the software. This concept of quality has a static character. After a product quality evaluation ends, the new quality evaluation after changes can be planned, but continuous quality monitoring during the period of usage between two quality testing points of the software is not expected. Gilles (1992) states, "The quality is transparent when presented, but easily recognized in its absence". This assertion is the consequence of the simple notion that it is not possible to foresee all circumstances and therefore the specification of needs and requirements for all quality characteristics can be not appropriate. Hence, tomorrow's quality view can be different from the today's one. For these reasons author suggests the implementation of a new subcharacteristic of software quality characteristic functionality. This subcharacteristic has the draft name "Users monitoring and cooperation" and describes the level of the ability of the software to cooperate with the related software and communicate with software product on Internet. After the adding the new subcharacteristic of functionality the step toward the transformation of the static software product quality model to the dynamical one can be accomplished. The benefit of this replenishment can be the minimalization of the period between the time when the quality problem occurs and the time when users are informed about the problem and instructed about precautions, which can be proposed. This possibility can lead to decrease in user's losses.

MATERIAL AND METHODS

In this part of the paper we shall describe in short principles of the current state of international standardization of software product quality. For details see Vaníček (2000), ISO/IEC 9126, ISO/IEC 14598, ISO/IEC 2500n, ISO/IEC 25051.

APPROACHES TO QUALITY

User quality needs can be specified as quality requirements by quality in use measures, by external measures, and sometimes by internal measures. These requirements specified by measures should be used as criteria when a product is validated. Achieving a product that satisfies the user's needs normally requires an iterative approach to software development with continual feedback from a user perspective.

External Quality Requirements specify the required level of quality from the external view. They include requirements derived from user quality needs, including quality in use requirements. External quality requirements are used as the target for validation at various stages of development.

Internal Quality Requirements specify the level of required quality from the internal view of the product. Internal quality requirements are used to specify properties of interim products. These can include static and dynamic models, other documents and source code. Internal quality requirements can be used as targets for validation at various stages of development. They can also be used for defining strategies of development and criteria for evaluation and verification during development. This may include the use of additional measures. Specific internal quality requirements should be specified quantitatively using internal measures.

Internal quality is the totality of characteristics of the software product from an internal view. Internal quality is measured and evaluated against the internal quality requirements. Details of software product quality can be improved during code implementation, reviewing and testing, but the fundamental nature of the software product quality represented by internal quality remains unchanged unless redesigned.

* Supported by the Ministry of Education, Youth and Sports of the Czech Republic (Grant No. MSM 6046070904 – Information and knowledge support of strategic control and No. 2C06004 – Information and knowledge management – IZMAN).

Estimated (or Predicted) External Quality is the quality that is estimated or predicted for the end software product at each stage of development for each quality characteristic, based on knowledge of the internal quality.

External Quality is the totality of characteristics of the software product from an external view. It is the quality when the software is executed, which is typically measured and evaluated while testing in a simulated environment with simulated data using external measures. During testing, most faults should be discovered and eliminated. However, some faults may still remain after testing. As it is difficult to correct the software architecture or other fundamental design aspects of the software, the fundamental design usually remains unchanged throughout testing.

Quality in Use is the user's view of the quality of the software product when it is used in a specific environment and a specific context of use. It measures the extent to which users can achieve their goals in a particular environment, rather than measuring the properties of the software itself.

Model for software product quality have two-part:

a) Internal quality and external quality, and
b) Quality in use.

The first part of the model specifies six characteristics for internal and external quality, which are further subdivided into subcharacteristics. These subcharacteristics are manifested externally when the software is used as a part of a computer system, and are a result of internal software attributes. The second part of the model specifies four quality in use characteristics, but does not elaborate the model for quality in use below the level of characteristics. Quality in use is the combined effect for the user of the six software product quality characteristics. The characteristics defined are applicable to every kind of software, including computer programs and data contained in firmware. The characteristics and subcharacteristics provide consistent terminology for software product quality. They also provide a framework for specifying quality requirements for software, and making trade-offs between software product capabilities.

USING A QUALITY MODEL

Software product quality should be evaluated using a defined quality model. The quality model should be used when setting quality goals for software products and intermediate products. Software product quality should be hierarchically decomposed into a quality model composed of characteristics and subcharacteristics, which can be used as a checklist of issues related to quality. It is not practically possible to measure all internal and external subcharacteristics for all parts of a large software product. Similarly it is not usually practical to measure quality in use for all possible user-task scenarios. Resources for evaluation need to be allocated between the different types of measurement dependent on the business objectives and the nature of the product and design processes. Their needs are to achieve specified goals with effectiveness, productivity, safety and satisfaction.

QUALITY MODEL FOR QUALITY IN USE

This clause defines the quality model for quality in use. The attributes of quality in use are categorized into four characteristics: effectiveness, productivity, safety and satisfaction. Quality in use is the user's view of quality. Achieving quality in use is dependent on achieving the necessary external quality, which in turn is dependent on achieving the necessary internal quality. Measures are normally required at all three levels, as meeting criteria for internal measures is not usually sufficient to ensure achievement of criteria for external measures, and meeting criteria for external measures of subcharacteristics is not usually sufficient to ensure achieving criteria for quality in use.

QUALITY IN USE

The capability of the software product to enable specified users to achieve specified goals with effectiveness, productivity, safety and satisfaction in specified contexts of use. Quality in use has following four characteristics:

Effectiveness

The capability of the software product to enable users to achieve specified goals with accuracy and completeness in a specified context of use.

Productivity

The capability of the software product to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved in a specified context of use.

Safety

The capability of the software product to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use.

Satisfaction

The capability of the software product to satisfy users in a specified context of use.

QUALITY MODEL FOR EXTERNAL AND INTERNAL QUALITY

This clause defines the quality model for external and internal quality. It categorized software quality attributes into the following six characteristics:

Functionality

The capability of the software product to provide functions, which meet, stated and implied needs when the software is used under specified conditions.

Reliability

The capability of the software product to maintain a specified level of performance when used under specified conditions.

Usability

The capability of the software product to be understood, learned, used and attractive to the user, when used under specified conditions.

Efficiency

The capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions.

Maintainability

The capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.

Portability

The capability of the software product to be transferred from one environment to another.

Each characteristic is divided into several sub-characteristics. Let us list the functionality sub-characteristics according the standard (ISO/IEC 9126) and at a rough some subsets of possible measures for each characteristic.

SUBCHARACTERISTIC FOR FUNCTIONALITY

This characteristic is concerned with what the software does to fulfill needs, whereas the other characteristics are mainly concerned with when and how it fulfils needs. For a system, which is operated by a user, the combination of functionality, reliability, usability and efficiency can be measured externally by quality in use.

Suitability

The capability of the software product to provide an ppropriate set of functions for specified tasks and user objectives.

Accuracy

The capability of the software product to provide the right or agreed results or effects with the needed degree of precision.

Interoperability

The capability of the software product to interact with one or more specified systems.

Security

The capability of the software product to protect information and data so that unauthorised persons or systems cannot read or modify them and authorised persons or systems are not denied access to them.

Functionality compliance

The capability of the software product to adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality.

MEASURES FOR FUNCTIONALITY

Following is the decomposition of the characteristic FUNCTIONALITY in sub-characteristics and for each sub-characteristic are indicated the measures to be used for the evaluation.

Measures for SUITABILITY

Functions available ratio
Functional specification change ratio
Precision of Input-Output definition ratio
Project documentation ratio
Product documentation ratio

Measures for ACCURACY

Significant digits ratio
Volume of code ratio
Correctness ratio

Measures for INTEROPERABILITY

Communicate-ability ratio
Matched data format ratio
Matched character ratio

Measures for SECURITY

Software access control ratio
Data access control ratio
Ciphered data ratio
Access history ratio
Data corruption ratio
Detected abnormal operation ratio

Measures for COMPLIANCE

Compliance with (project) software development standards ratio
Compliance with (project) documentation standards ratio
Standardised data format ratio
Standardised character ratio

RESULTS AND DISCUSSION

NEW SUBCHARACTERISTIC PROPOSAL

One of the phenomena of mass expansion of information technology is the development and expansion of software product, which is commonly used by huge user community. Examples of such products are commercial-of-the-self software products. Maintaining and servicing such products is often supported by special software which uses a private network or the Internet to inform the team responsible for maintenance about all problems and failures during the product usage in the whole users community. These messages are usually send in the client server mode from the users towards the maintenance team. This team or developers send users information, such as warnings about possible problems, ways to avoid or temporarily solve possible problems and about future plans of error fixes. Such information is extremely useful for users.

The service described in the paragraph above can be regarded as a special functionality feature. It is reasonable to regard it as a special subcharacteristic of functionality, with a close direct influence to the reliability and maintainability quality characteristic.

This discussion leads to a proposal of a new functionality subcharacteristic with the draft name **”Users monitoring and cooperation”.**

The proposed definition of this subcharacteristic is the following:

Users monitoring and cooperation

The capability of the of-the-self software product to provide the level of the ability of the software to be centrally monitored by the maintenance centre and communicate with other copies of the same software product on private network or Internet.

SOME ATTRIBUTES AND MEASUREES PROPOSAL

In this paragraph we try to outline the same possible external and internal attributes for a new subcharacteristic and suggest some possible new measures.

Proposed external attributes and measures

Communicated failures ratio

The ratio of the number of problems (failures) during the specified time period which are automatically communicated by a special monitoring software to a central maintenance point, to a number of all problems that occurred during the specified time period.

This measurement is an absolute scale type, and can be evaluated during the exploitation stage of the product life cycle. The alternative measure of such attribute can be suggested when we do not count individual failures but only the number of different type of failures (due to the same fault).

User and maintainer information sharing ratio

The ratio of the of the number of failure report from different users, which are published in the open real time accessible document (for example www page) and the number of all failure reports monitored by maintainer.

This measurement is an absolute scale type, and can be evaluated during the exploitation stage of the product life cycle. The alternative measure of such attribute can be suggested when we do not count individual failures but only the number of different type of failures (due to the same fault).

The central warning density

The number of warnings sent from a central maintenance point to all registered product users divided by the time duration of monitoring.

This measurement is a ratio scale type and can be evaluated from the monitoring protocol recording on the central maintenance point. The alternative measure is the same number divided by the software size, which is measured by the appropriate software complexity measure (for example LOC).

Mean supplier reaction time

The average time between the message about a problem sending from the user to central maintenance point and the time when the message with appropriate warning and information how to avoid problem is send to all product users, counted in the monitoring time period.

This measurement is a ratio scale type and can be evaluated from the monitoring protocol recording on the central maintenance point. The alternative measure is the same number divided by the software size, measured by the appropriate software complexity measure (for example LOC). Another alternatives can be obtained by replacing the arithmetic mean by the median or some other aggregation operator.

Proposed internal attributes and measures

Implementation of failure communication ratio

The ratio between the number of points in the program code, where the automatic report about the failure is implemented to all points in the program code, where failure can occur.

This measurement is an absolute scale type, and can be evaluated during the design stage of a software life cycle. The source of data can be design documents and source code of software. The measure value can be used as a predictor for the Communicated failures ratio external measure in the variant when only different types of failures with the same source faults are counted.

Implementation of failure publicity service ratio

The ratio between the number of points in program code when the failure reports from individual users are draft on the open real time accessible document to all points where the reports about failures are collected for maintainer.

This measurement is an absolute scale type, and can be evaluated during the design stage of a software life cycle. The source of data can be design documents and source code of software. The measure value can be used as a predictor for the external measure value User and maintainer information sharing ratio.

Failure monitoring implementation density

The number of program points in the source code, where the automatic collection of possible failures is implemented, divided by the software size, measured by the appropriate software complexity measure (for example LOC). This measurement is a ratio scale type and can be evaluated from design documents and source code of software.

Of course the list of possible attributes for the suggested subcharacteristic is not comprehensive.

CONCLUSION

The possible adding of a new subcharacteristic "Users monitoring and cooperation" to a software quality model can contribute to more complex evaluation of modern software product with the broad implications. Evaluation of such quality aspects can help to the development of software with a lower risk to the damages due to software failures.

REFERENCES

GILLES, A. C.: Software Quality, Theory and Management. Chapman & Hall 1992.

ISO/IEC 9126: Information technology – Software product quality.

ISO/IEC 14598: Information technology – Software product evaluation.

ISO/IEC 2500n: Quality Management Division.

ISO/IEC 25051: Software Engineering – Software product evaluation – Requirements for quality of Commercial Off The Shelf software product (COTS) and instructions for testing.

VANÍČEK, J.: Měření a hodnocení jakosti informačních systémů. Praha, ČZÚ 2000.

Received for publication on January 14, 2008
Accepted for publication on March 3, 2008

KARDOŠ, D. (Česká zemědělská univerzita, Fakulta provozně ekonomická, katedra informačního inženýrství, Praha, Česká republika):

Dynamické zajištění kvality softwaru.

Scientia Agric. Bohem., *39*, 2008: 92–96.

Článek popisuje princip navrhovaného dynamického zajištění kvality softwaru pomocí zavedení hodnocení doposud nehodnocené vlastnosti softwarového produktu. Stávající pohled na kvalitu softwaru je definován zejména dosažením specifikovaných vhodných charakteristik jakosti s tím, že se bere v úvahu účel používání softwarového produktu. Tento koncept kvality má statický charakter. Při použití stávajícího modelu kvality se sice vyžaduje po provedení hodnocení naplánovat následující termín zkoušky, ale průběžně kvalita v období mezi dvěma zkouškami není sledována. G i l l e s (1992) uvádí, že kvalita je „přehlédnutelná, když ji navrhujeme, ale snadno rozpoznáme, když chybí". Tento závěr vychází z jednoduché myšlenky, že není možné správně předvídat všechny okolnosti, a proto specifikace všech potřeb a požadavků na charakteristiky kvality nemusí být vhodná. Z tohoto důvodu může být zítřejší pohled na kvalitu softwarového produktu jiný než dnešní. Autor proto navrhuje zavedení nové subcharakteristiky v rámci charakteristiky jakosti funkčnost. Tato subcharakteristika byla pojmenována „Sledování a spolupráce s uživateli" a je definována jako úroveň schopnosti masově rozšířeného komerčního softwarového produktu být centrálně sledován centrem údržby a spolupracovat s dalšími kopiemi téhož produktu prostřednictvím privátních sítí nebo internetu. Doplněním nové subcharakteristiky funkčnosti vytvoříme z původně statického modelu model dynamický. Přínosem tohoto doplnění může být minimalizace doby mezi časem, kdy problém s kvalitou nastane, a časem, kdy uživatelé jsou informováni o problému nebo navrhovaných opatřeních. To může vést ke snížení případných škod u uživatelů.

dynamické zajištění kvality softwaru; kvalita softwarového produktu; míry pro sledování a spolupráci s uživateli

Contact Address:

Ing. Daniel K a r d o š , Česká zemědělská univerzita v Praze, Fakulta provozně ekonomická, katedra informačního inženýrství, Kamýcká 1076, 165 21 Praha 6-Suchdol, Česká republika, tel.: +420 774 061 028, e-mail: kardos@pef.czu.cz