A METRICS PROPOSAL TO EVALUATE SOFTWARE INTERNAL QUALITY WITH SCENARIOS

Anna Grimán, María Pérez, Maryoly Ortega and Luis Mendoza
Processes and Systems Department – LISI
Universidad Simón Bolívar
Caracas, Venezuela

Keywords: Software quality assurance, Evaluation, Internal quality, Metrics.

Abstract: Software quality should be evaluated from different perspectives; we highlight the internal and external ones (ISO/IEC, 2002). Specially, internal quality evaluation depends on the software architecture (or design) and programming aspects rather than on the product behaviour. On the other hand, *architectural evaluation methods* tend to apply scenarios for assessing the architecture respect to quality requirements; however, mainly *scenarios* aren't effective enough to determine the level of satisfaction of the quality attributes. In practice, each scenario could need more than one measurement. Also, we need a quantitative way of comparing and reporting results. The main objective of this article is presenting a proposal of metrics grouped by quality characteristics and sub-characteristics, according to ISO 9126 standard, which can be applied to assess software quality based on architecture. Once selected the most important quality requirements, these metrics can be used directly, or in combination with quality scenarios, into an architectural evaluation method. Metrics proposed also consider some particular technologies, such as OO, distributed and web systems.

1 INTRODUCTION

According to Barbacci et al. (1997) the software quality is defined by the degree in which the software exhibits the correct combination of desired attributes. Such attributes encompass system requirements which are different from the functional requirements (Clements et al., 2002), and which refer to the specific characteristics that the software must meet. We know these characteristics or attributes as *quality attributes* and they are defined as properties inherent to a service rendered by a particular system to its users (Barbacci et al., 1997).

In this sense, most of the Architectural Evaluation Methods of Software Quality use scenarios to assess the degree of satisfaction of the quality attributes or requirements in a software. In reality, however, these scenarios cannot be measured directly; on the contrary they require an assigned metric or measure whose value can be represented in different scales and can be incorporated to the results from the whole of the quality attributes.

Thus, the objective of this article is to present a proposal of metrics organized into quality characteristics and sub-characteristics, according to the ISO 9126 standard, which can be applied to assess software quality based on its architecture. This study was inspired in a previous research (Grimán et al., 2006) which established the elements to be evaluated in software architecture in order to estimate software quality.

There are different works which related ISO 9126 to software measurement (Mavromoustakos and Andreou, 2007; Lee and Lee, 2006; Azuma, 1996; among others), however, they don't present the evaluation in an architectural level. In this investigation we have used two related works (Losavio and Levy, 2002) and (Ortega et al., 2003) which proposed some metrics for software evaluation considering internal (architectural) elements according to ISO 9126.

In section 2 we briefly outline the theoretical basis for this work; in sections 3 and 4 we propose a set of metrics aimed at internal quality evaluation and its application within a case study; finally in section 5 we introduce the conclusions of our work and our recommendations for future researches.

Grimán A., Pérez M., Ortega M. and Mendoza L. (2007).
A METRICS PROPOSAL TO EVALUATE SOFTWARE INTERNAL QUALITY WITH SCENARIOS.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - DISI*, pages 558-563
DOI: 10.5220/0002391905580563
Copyright © SciTePress

2 BACKGROUND

The ISO 9126 standard (ISO/IEC, 2002) does not define software quality incorporation; it simply defines a model with the characteristics to be taken into account when establishing quality requirements for product evaluation. This standard outlines a software quality model which includes internal and external quality and quality of use. It specifies six (6) characteristics for internal and external quality, with subsequent divisions (or sub-characteristics), which are the product of internal attributes and come into play externally whenever the software is used as a part of a computing system. The characteristics are applicable to any type of software.

The ISO/IEC 9126 model organizes quality attributes into six characteristics (Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability). Each characteristic is, in turn, subdivided into sub-characteristics that can be measured through internal and external metrics. The *Quality of Use* refers to the software's capability to allow users to attain specific goals effectively, productively, securely and satisfactorily within a particular context.

A metric of software quality refers to a quantitative method and scale used to determine the value of a particular quality characteristic in a software product. According to ISO 9126 the evaluation of software quality can be achieved through internal and external metrics (ISO/IEC, 2002).

In this study we are particularly interested on **internal metrics** which can be applied to a software product without executing (to specifications or source code) during the design or codification. When a software product is developed the intermediate product can be evaluated using internal metrics to measure intrinsic properties.

In order to generate the internal metrics we used the Goal Question Metric (GQM) approach (Basili, 1992) (sometimes called the GQM paradigm) which supports a top-down approach to define the goals behind measuring software processes and products, and using these goals to decide precisely what to measure (choosing metrics). It additionally supports a bottom-up approach to interpreting data based on the previously defined goals and questions.

In our case, the expected quality attributes, as outlined by the ISO 9126 standard, were established as goals, where the steps previously mentioned were applied to each attribute. Due to space constraints we have omitted the questions that emerged during the research, focusing on the metrics generated which are shown in the next section.

3 METRICS FOR EVALUATING INTERNAL QUALITY OF SOFTWARE

As previously stated, in this work we have considered our previous findings about architectural evaluation (Grimán et al., 2006). According to that study an evaluation method include a quality model to estimate the satisfaction of quality requirements. Metrics can be used to compose a quality model and evaluate software architecture. In this sense they must assess architectural mechanisms (patterns and styles) and models (composed by components, relationships, and views). Grimán et al. (2006) also established that ISO 9126 is a convenient framework to organize the quality model needed in the evaluation. This way, in this investigation the organization of the proposed metrics followed the ISO 9126 structure.

Based on this study we oriented the metrics proposal towards the evaluation of: elements (presence/absence, quantity), relationships (presence/absence, type), views (consistency, layers, and balance), models (consistency, layers, and balance), and mechanisms (presence/absence of architectural patterns or styles). These key architectural aspects constituted the goal of our assessment; the next step was proposing one or more metrics to achieve each goal.

Respect to the metrics proposal, two other works were considered as antecedents: (1) Ortega et al. (2003), which, in turn, was inspired on ISO 9126 and proposed a seminal set of metrics to measure mainly the external quality of the product, factoring in efficiency and effectiveness for each one of the characteristics. After analyzing, we found they included 10 internal metrics (related to Maintainability), which were generalized and incorporated in our proposal; (2) Losavio and Levy (2002) who proposed definitions and parameters to measure and evaluate the quality characteristics and sub-characteristics at the architectural level. Four metrics proposed by (Losavio and Levy, 2002) were incorporated in our proposal, they were related to Functionality, Reliability and Portability.

As a result of this study we obtained a model of 117 internal metrics as follows: Functionality: 24, Reliability: 30, Maintainability: 31, Efficiency: 15, Portability: 4 and Usability: 13. We had also to determine if all the sub-characteristics were architectural in nature, as a result of this analysis we were able to organize the 117 metrics into characteristics and sub-characteristics.

These metrics were validated through the application of different architectural evaluations onto different case studies (Grimán et al., 2003;

Grimán et al., 2003b; Grimán et al, 2004; Grimán et al., 2005; Grimán et al., 2006) where Collaborative Systems, Enterprise Information Systems, Financial Systems, CASE tools, and Geographical Information Systems were evaluated. In these validations 22 stakeholders (developers, architects, project managers, etc.) were involved to evaluate the proposal of metrics according to the following features: appropriateness, feasibility, depth, and scale. In general, we obtained values higher than the minimal level of acceptance (75%) for each evaluated metric.

Because of space restrictions, we only present a sample of the 117 metrics. Figure 1 presents the proposed metrics for Functionality which were considered pertinent to the software's architecture, based on the definitions by Bass et al. (2003) and Losavio and Levy (2002): Adequacy, Precision, Security and Interoperability. Note that 2 metrics in this set were marked (^{1}) because they were proposed by Losavio and Levy (2002). For each metric, we have established a corresponding scale within the range of 1 through 5, where 5 always represents the highest value (see Table 1).

Table 1: Sample of ordinal scales.

Type 1	5**:** Yes; 1: No
Type 2	5 = All; 4= Nearly All; 3= Median.; 2=Few; 1=None
Type 3	1=Unacceptable; 2=Below Average; 3=Average; 4=Good; 5=Excellent
Type 4	5 = Completely satisf.; 4 = Almost Always satisf.; 3= Occasionally Satisf.; 2 = Rarely satif.; 1= Never satisf.
Type 5	5=Completely satisf.; 3=somewhat satisf.; 1= Not satisf.

In the case of Functionality, we must point out that, even though the software's functional requirements can be regarded as orthogonal to the architecture; there are specific architectural strategies to achieve the desired Security, Precision and Interoperability requirements.

3.1 Analysis of Metrics

Regarding Functionality we proposed metrics essentially oriented to evaluating the presence of elements or components within the architecture, which support the identified requirements. Some of the metrics proposed will have to be refined in order to be applied upon each actual evaluation e.g. *the existence of any component/function/method for each specific task within the requirements* would have to be specialized in as many metrics as requirements have been identified.

Usability is a quality characteristic which can be translated into Functionality; however in this work we have also considered other internal aspects that would have an impact on the software's interaction e.g. *modules or components which are responsible for only one task*. The complexity of the required interfaces are evaluated by a module or method i.e. the more specific the method is, the simpler its interface.

Regarding Reliability, the proposed metrics evaluate the presence or existence of mechanisms to prevent or handle exceptions. Some metrics can be applied to Distributed Systems e.g. *Presence of mechanisms which allow shared access to the resources within the system* or *Presence of mechanisms which allow stopping non-operative processes* are metrics aimed at evaluating the maturity of the architecture in order to prevent malfunctions, whereas the metric *Presence of mechanisms of failure notification* will evaluate aspects regarding the management of exceptions The metrics *Presence of mechanisms which allow restarting the system in degrade mode* and *Presence of any mechanism which allows the recovery of the previous status of the system* are designed to determine the probability for future software to remain operative after a malfunction has occurred.

Even though Efficiency is a quality characteristic that is difficult to estimate through the architecture –especially because it is hardware dependent, as well as dependent on the communication aspects,- we proposed a set of metrics aimed at evaluating resource management within the architecture, in order to optimize resource consumption within large or complex systems. This will translate into a better response time ratio. Thus, such metrics do not contemplate the platform where the software will be executed, but rather the performance inherent to a particular architectural organization, e.g. *Existence of unnecessary connections between classes* or *Presence of mechanisms which optimize broadband use*.

As far as Portability we found that it is fundamentally an architectural characteristic, even though it can be translated into some functionality especially regarding Installability e.g. *Presence of mechanisms which allow the system's installation*. We proposed a number of general metrics designed to evaluate the presence of mechanisms and strategies which render the software usable within multiple platforms e.g. *Presence of mechanisms (layer or subsystem) which encapsulates the restrictions of the environment*.

Functionability	Adequacy	Existence of any component /function/method for each specific task within users' requirement (e.g. enquiries, reports, transactions)	Type_1
		System functions adequately refined in the architecture	Type_2
		Completion of Use-case Model vs. Logical Model	Type_2
		Completion of Concepts vs. Database Components	Type_2
		Completion of the Logical Model vs. Component Diagrams	Type_2
		Existence of mechanisms which promote communication	Type_1
		Re-configurable architecture in run-time	Type_1
		Design adapted to the original problem	Type_3
		Class conceptual diagram adapted to software specifications	Type_1
	Precision	*Presence of software components responsible for the calculations needed by the client* [1]	Type_4
	Security	Presence of architecture components for detection of non-authorized system access	Type_1
		Presence of a mechanism/subsystem/kernel which determines user access when users need to perform a particular operation	Type_1
		Presence of a secure mechanism which allows users to remember their password	Type_1
		Presence of a mechanism which reminds users to change their password after some time	Type_1
		Presence of a mechanism within the software architecture which includes a function to evaluate the amount of access logs within the system	Type_1
		Presence of an access blocking mechanism after a number of failed attempts	Type_1
		Presence of mechanisms (methods, procedures or functions) which register user operations in a log	Type_1
		Compliance with security standards	Type_5
		Presence of mechanisms for recovery after failures due to attacks	Type_1
		Presence of mechanisms for data update used for authentication	Type_1
		Presence of mechanisms for data encryption	Type_1
		Presence of mechanisms which limit domain access in the network (applicable to intranet development	Type_1
	Interoperability	*Existence of elements which allow connections to other systems*[1] (e.g. middleware, COBRA etc.)	Type_1
		Components with clear communication interfaces	Type_1
		The Architecture uses certified components	Type_5
		Clear identification of architecture extension points (applicable to systems based on plug-in and components	Type_1

Figure 1: Internal quality metrics for Functionality.

The last characteristic we found was Maintainability, which originates a number of important general metrics, which can be further refined. These metrics are designed to evaluate the effortlessness with which the architecture can be understood and modified, focusing mainly on its organization and distribution, e.g. *Responsibility of each object within the Interaction Diagrams*. Our proposal did not contemplate semiotic-related metrics, such as the ideality of any particular symbol.

Finally, the proposed metrics include measurements at various levels which correspond to different architectural views, in order to apply some estimates even in cases where no refined design is available.

4 CASE STUDY: GIS EVALUATION

With the objective of showing a practical application of the metrics proposed in this work, we present the study of a particular case in which the architecture of a Geographical Information System (GIS) for an electrical distribution company's was evaluated. The system was very demanding in terms of quality, and it was developmentally complex. The system's purpose was to address the optimization needs of those processes inherent to electric energy distribution while meeting the legal dispositions outlined in the quality regulations of the National Electrical Distribution System (GISEN).

The evaluation of this architecture was done following the Architectural Trade-offs Analysis Method – ATAM (Clements et al., 2002), which required establishing the architectural drivers and the utility tree. Given that the former is based on quality scenarios, we included some of the metrics proposed in our work in order to facilitate the quantitative analysis of such scenarios.

The quality requirements as well as the expected attributes of the system were: precision in the calculations (Accuracy); communication with other systems (Interoperability); access control (Security); quantification of the permitted malfunctions, management of internal and external errors (Reliability); requirements of interface and communication with users (Usability); speed, response-timeframe and consumption of resources (Efficiency).

Once we identified the characteristics we designed a utility tree based on the quality scenarios and on their weight and frequency. One or more of the proposed metrics were assigned to each scenario in order to allow an evaluation by the evaluating group; two examples are shown next:

Scenario: We require communication with the Incidence System. Metrics Proposed: *Existence of elements which allow connections with other* *systems (e.g. middleware CORBA, etc), Components with clear communication interfaces.*

Scenario: A transmission of sensitive is preformed through a public network. Metrics Proposed: *Presence of mechanisms for data encryption.*

In order to evaluate the architecture quality, we coordinated a one-day session with different stakeholders to assign values to each scenario/metric. After evaluating, the results obtained were quantitative (see Figure 2) and showed the required architectural improvements upon those characteristics which are inhibited. In this case, this was represented by those characteristics whose values were less than 75%.

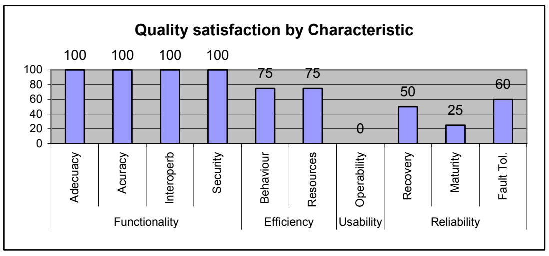

Figure 2: Percentages of coverage of quality characteristics.

Based on these results, we were also able to provide practical recommendations regarding the architecture improvements. Some examples are: *use of concurrent processing and of dispatch policies*, *mechanisms to manage the workload*, *helps for users (including tool tips)*, *use of mechanisms of data updating for authentication*, among others.

After illustrating the practical application of the proposed metrics in a real context, we present the conclusions and opened questions of this research in the next section.

5 CONCLUSIONS

Through the results achieved in this work, it becomes evident that some of the quality characteristics or attributes (especially the non-observable during execute time), are more easily evaluated through the architecture (e.g. Maintainability).

It is also evident, however, that those quality attributes which are observable during execute time are directly determined by the architecture. It is possible then to establish objective measurements about the architecture's specifications which would allow us to estimate the levels of product quality at an early stage.

These metrics can be used in combination with different architectural evaluating techniques, (e.g. scenarios and simulations), in order to obtain quantitative measures of quality. Similarly, the metrics can be redefined so as to obtain more precise evaluations of specific architectural mechanisms or technologies. This way we have proposed a compendium of metrics aimed at estimating internal quality (based on standard ISO 9126) and which, at the same time, functions as a model for specification. This work provides important orientation for the application of the different methods of architectural evaluation which claim for quantitative measurements.

In the future and as a complement to our proposal, we intend to include aspects of Consistency, Conceptual Integrity, Simplicity of Construction and Comprehension of the Architecture, which are regarded as inherent to the quality of the architecture more so than to the software quality.

REFERENCES

Azuma, M. 1996. Software products evaluation system: Quality models, metrics and processes -international standards and Japanese practice. Information and Software Technology, 38 (3 SPEC. ISS.):145-154.

Barbacci, M, Klein, M., Weinstock, C., 1997. Principles for evaluating the quality Attributes of a Software Architecture. Technical Report CMU/SEI-96-TR-036.

Basili, V., 1992. Software modeling and measurement: The Goal/Question/Metric paradigm. Technical Report CS-TR-2956, Department of Computer Science, University of Maryland, College Park, MD 20742.

Bass, L., John, B., Kates, J., 2001. Achieving Usability through Software Architecture. CMU/SEI-TR-2001-005. Software Engineering Institute.

Bass, L., Clements, P. and Kazman, R., 2003. Software arquitecture in practice. Addison Wesley Longman Inc., USA.

Clements, P., Kazman, R., Klein M., 2002. Evaluating Software Architectures: Methods and Case Studies. The SEI Series in Software Engineering.

Grimán, A., Pérez, M., Mendoza, L, 2003. Estrategia de pruebas para software OO que garanticen requerimientos no funcionales. In III Workshop de Ingeniería del Software, Jornadas Chilenas de Computación.

Grimán, A., Lucena, E., Pérez, M., Mendoza, L., 2003. Quality-oriented Architectural Approaches for Enterprise Systems. In 6th Annual Conference of the Southern Association for Information Systems.

Grimán. A., Pérez, M., Mendoza, L, Hidalgo, I., 2004. Evaluación de la calidad de patrones arquitectónicos a través de experimentos cuantitativos. In Jornadas

Iberoamericanas de Ingeniería del Software e Ingeniería del Conocimiento.

Griman, A., Valdosera, L., Mendoza, L., Pérez, M., Méndez, E., 2005. Issues for Evaluating Reliability in Software Architecture. In 8th Americas Conference on Information Systems.

Griman, A.; Chávez, L.; Pérez, M.; Mendoza, L.; Dominguez, K. 2006. Towards a Maintainability Evaluation in Software Architectures. In 8th International Conference on Enterprise Information Systems - ICEIS . Vol. 1: 555 - 558.

ISO/IEC, 2002. Software Engineering – Software quality – General overview, reference models and guide to Software Product Quality Requirements and Evaluation (SQuaRE). Reporte. JTC1/SC7/WG6.

Lee, K. and Lee, S. 2006. A quantitative evaluation model using the ISO/IEC 9126 quality model in the component based development process. Lecture Notes in Computer Science: 917-926.

Losavio, F., Levy, N., 2002. Putting ISO standards into practice for Architecture evaluation with the Unified Process. Journal of Object Technology, Vol. 2, Nº 2.

Mavromoustakos, S. and Andreou, A. 2007. WAQE: A Web Application Quality Evaluation model. In International Journal of Web Engineering and Technology, 3 (1): 96-120.

Ortega, M, Pérez, M., Rojas, T., 2003. Construction of a Systemic Quality Model for evaluating a Software Product. Software Quality Journal. Kluwer Academic Publishers, 11 (3): 219-242.