Evaluation of Software Quality

Krzysztof Sacha

Warsaw University of Technology, Nowowiejska 15/19
00-665 Warszawa, Poland
k.sacha@ia.pw.edu.pl

Abstract. The paper describes a method, which we used to evaluate the expected quality of software that was developed for a huge governmental system. The evaluation lasted nearly two years and was performed along with the software development process. The output that was expected by our customer consisted of a quality assessment accompanied by a set of recommendations on what to do in order to enhance the quality of the product.

1 Introduction

The ultimate goal of software engineering is to find methods for developing high quality software products at reasonable cost. As computers are being used in more and more critical areas of the industry, the quality of software becomes a key factor of business success and human safety.

Two approaches can be followed to ensure software quality. One is focused on a direct specification and evaluation of the quality of software product, while the other is focused on assuring high quality of the process by which the product is developed.

The software industry is currently entering a period of maturity, in which particular informal approaches are specified more precisely and are supported by the appropriate standards. Quality characteristics of software products are defined in ISO/IEC 9126 [1]. For each characteristic, a set of attributes which can be measured is determined. Such a definition helps in evaluating the quality of software, but gives no guidance on how to construct a high quality software product.

The requirements for a quality management system are defined in ISO 9001 [2]. All the requirements are intended for application within a software process in order to enhance the customer satisfaction, which is considered the primary measure of the software product quality. The quality management system, as defined by the standard, can be subject to a certification.

This paper describes a method, which we used to evaluate the expected as well as the actual quality of a huge software system that was developed in the years of 2003-2004 to support Common Agriculture Policy of European Union in Poland (IACS – Integrated Administration and Control System). Both of the two approaches, mentioned above, appeared to be too abstract for a direct application. Moreover, the quality of a software product cannot be evaluated unless the development of this particular product has been finished, and high quality of a software process does not necessarily guarantee high quality of the product. Therefore we were pushed to develop an original method. A general framework of this method has been described in [3].

The paper is organized as follows. Section 2 provides the reader with a short overview of the approach represented by ISO 9126, and Section 3 summarizes the approach of ISO 9001. The method that was used to evaluate the quality of the development of the IACS software is presented in Section 4. Final remarks are gathered in Conclusions.

2 ISO/IEC 9126 Overview

ISO 9126 [1] is concerned primarily with the definition of a quality model, which can be used to specify the required product quality, both for software development and software evaluation. The model consists of six quality characteristics, which are intended to be exhaustive. Each quality characteristic is very broad and therefore it is subdivided into a set of sub-characteristics or attributes. This quality model can be applied in industry through the use of measurement techniques and related metrics.

All the six quality characteristics, defined in ISO 9126, are recapitulated below, along with some comments.

Functionality is defined as the ability of the software product to provide functions which meet stated or implied needs of the user. This is a very basic characteristic, which is semantically close to the property of correctness, as defined in other quality models [4]. If software does not provide the required functionality, then it may be reliable, portable etc., but no one will use it.

Efficiency is a characteristic that captures the ability of a correct software product to provide appropriate performance in relation to the amount of resources used. Efficiency can be considered an indication of how well a system works, provided that the functionality requirements are met. The reference to the amount of resources used, which appears in this definition is important, as the traditional measures of efficiency, such as the response time and throughput, are in fact system-level attributes.

Usability is a measure of the effort needed to learn and use a software product for the purpose chosen. The scope of this factor includes also the ease of assessment whether the software is suitable for a given purpose and the range of tolerance to the user errors. The features that are important within the context of usability are adequate documentation and support, and the intuitive understandability of the user interface.

Reliability is defined as the ability of software to maintain a specified level of performance within the specified usage conditions. Such a definition is significantly broader than the usual requirement to retain functionality over a period of time, and emphasizes the fact that functionality is only one of the elements of software quality that should be preserved by a reliable software product.

Maintainability describes the ease with which the software product can be analyzed, changed and tested. The capability to avoid unexpected effects from modifications to the software is also within the scope of this characteristic. All types of modifications, i.e. corrections, improvements and adaptation to changes in requirements and in environment are covered by this characteristic.

Portability is a measure of the effort that is needed to move software to another computing platform. This characteristic becomes particularly important in case of an application that is developed to run in a distributed heterogeneous environment or on a high performance computing platform, which lifespan is usually short. It is less important if the application runs in a stable environment that is not likely to be changed.

It can be noticed that the characteristics correspond to the product only and avoid any statement related to the development process. This way the standard presents a coherent model, which skips such features like e.g. **timeliness**, which can be defined as the ability of a product to meet delivery deadlines. Although timeliness relates closer to the process than to the product itself, it can influence the subjective feeling of the customer. If a product is delivered late, then it may be of good quality, but customers may consider it to be of lesser quality than the products delivered on time [5].

3 ISO 9001 Overview

ISO 9001 [2] describes the requirements for a quality management system, which is a part of the total manufacturing process. The standard is very general and applies to all types of organizations, regardless of their size and of what they do. The recommended practices can help both product and service oriented organizations and, in particular, can be used within the context of software development and manufacturing. ISO 9001 certificates are recognized and respected throughout the world.

Because of this generality it is not easy to map the recommendations of the standard into the practical activities that can be performed within a software process. Moreover, the standard is intended to be used by the manufacturers and not by the auditors that are hired by their customers. Therefore it contains many recommendations that relate to resource management process, which was completely outside the scope of our evaluation. What we were expected to assess was the quality of products of particular steps of the development process: analytical specifications, design documents, testing plans and procedures, user manuals and the resulting code. The actual implementation of the testing process was also subject to our evaluation.

ISO 9001 does not define any particular model of quality. Instead, it adopts a simple approach that the quality of a product is measured by the customer satisfaction. According to this approach, no quality characteristics are defined, and the only basis for quality evaluation are the customer requirements. If those requirements are met, then the product quality can be evaluated high. The lack of a quality model makes this standard orthogonal to ISO 9126. There are no common points between the two, but also no contradiction can be found.

The top level requirement of ISO 9001 is such that a quality management system must be developed, implemented and maintained. All the processes and activities performed within the scope of this system have to be documented and recorded for the purpose of future review. A huge part of the standard relates to the processes of quality planning and management, resource management, and continuous quality monitoring, analysis and improvement. This part is not very helpful in evaluating the quality of a specific software product under design.

The part, which relates directly to the body of a software project, is a section on realization requirements. Basic requirements and recommendations that are stated therein can be summarized as follows:

1. Identify customers' product requirements, i.e. the requirements that the customer wants to meet, that are dictated by the product's use or by legal regulations.
2. Review the product requirements, maintain a record of the reviews, and control changes in the product requirements.
3. Develop the software process, clarify the responsibilities and authorities, define the inputs and outputs of particular stages.
4. Perform the necessary verification and validation activities, maintain a record of these activities, and manage design and development changes.

All of those statements are very concrete and provide valuable guidelines for auditing and evaluating the quality of a software process. Moreover, the stress that is placed on the need to meet customer requirements helps in closing the gap between the quality of the software process and the quality of software itself.

4 Quality Evaluation Method

There is a great difference between the quality evaluation made for and by a software manufacturer and the evaluation that is made for the customer.

A manufacturer can define a set of metrics that describe particular quality characteristics, measure and collect historical data from a set of similar projects, and compare the current data with the ones taken from the historical database. The entire process can be supported by computerized tools like the one described in [6]. Such an approach is focused on a direct evaluation of the quality of a software product, and can be implemented using GQM method (Goal – Question – Metric), described for the first time in [7] and developed since that time by NASA. The set of goals or quality characteristics can be the same or similar to the one defined in ISO/IEC 9126.

It is very difficult to a customer to follow this approach. The lack of historical data makes many of the quantitative characteristics useless. For example, consider the following metric: "The percentage of classes that contain a method for displaying help information for the user" (Tab 1 in [6]) and assume that it has been measured equal to 45%. What does this data mean? Is 45% many or few? How does 45% contribute to the usability of the software product – is it high or low?

4.1 Process-Centric Approach

The problems related to the interpretation of many quantitative metrics of software quality pushed us towards the approach focused on the evaluation of the quality of the development process. The rationale that stands behind this approach is based on a hope that if things are done right, than the results will also be right. Since *hope* is not the same as *certainty*, we tried to relate somehow the attributes of process quality to the attributes of product quality defined in ISO 9126. An important advantage of such a process-centric approach is that it gives the auditor an opportunity to formulate recommendations on what to improve in the development process.

The method we developed was based on a set of criteria that characterize the quality of the software process or the quality of a product of a particular step of the software process. The method was similar to GQM in that the evaluation of particular criteria was based on a process of asking questions and giving answers to these questions. To avoid problems with the interpretation of data, the sets of questions were limited to the ones that were meaningful for the user. To make the evaluation process structured, we divided the problem space into six subject areas. The first subject area corresponded to the development process itself, the next four areas corresponded to particular activities within the process, and the last one dealt with the software documentation. This way the following areas were defined:

1. Software process and development methods.
2. The analysis and analysis products.
3. The design and design products.
4. The implementation and the code.
5. Testing process and test documentation.
6. User manuals.

It can be noted from the above list that the areas 2 – 5 cover all the activities that exist in both: waterfall and incremental models of software development. The decomposition of the software process into the subject areas is then exhaustive.

Each subject area was covered by a set of criteria, organized along the paths of traceability: from requirements to verification, from requirements to the design, and from the design to the implementation. The majority of answers was qualitative. Quantitative measures were also used, however, only in such cases in which the numbers could be meaningful for the customer. Sample criteria and questions that were stated in particular subject areas, are discussed briefly in the next two subsections. The mechanics of the quality evaluation is described in Section 4.4.

4.2 Software process and development methods

The main goals of the activities performed within this subject area are the identification of methods and standards that were used throughout the development process and the verification of the use of those methods with respect to completeness, readability and traceability of the resulting products. In order to achieve these goals we defined the following set of criteria, accompanied by the appropriate questions:

1. Methods and standards: Which methods and standards were used throughout the development process? How was the scope of the methods?
2. Completeness of results: Which artifacts recommended by the methods have been created? Is the set of artifacts sufficient? Are the results documented properly?
3. Readability and modifiability of the documentation: Are the created documents readable and modifiable?
4. Traceability of the documentation: Are the documents created in subsequent steps of the development process traceable?

As can be seen from the above list of criteria, the evaluation within this subject area was focused on formal aspects of the development in that it did not include an in depth analysis of the contents of the documents created.

The criteria correspond to the recommendations of ISO 9001, which state that a software process shall be developed, the outputs of particular activities shall be defined, and the results shall be recorded. The criteria have also a clear relation to the quality characteristics defined in ISO 9126, because the completeness of results together with readability, modifiability and traceability of the documentation promotes the maintainability and portability of the software product.

4.3 Analysis

The main goals of the activities performed within this subject area are the identification of methods and models that were used throughout the analysis and the verification of the use of those methods with respect to correctness, completeness and verifiability of the resulting documents and other products. In order to achieve these goals we defined the following set of criteria, accompanied by the appropriate questions:

1. Completeness of sources: Which sources of information were used throughout the analysis? Was the selection of legal regulations complete?
2. Consistency of the analysis model: Is the business model created during the analysis consistent with legal regulations that have been identified in criterion 1?
3. Completeness of the analysis model: Is the created analysis model complete in that it comprises all the business procedures defined by the legal regulations? Is the list of possible scenarios complete?
4. Completeness of the context definition: Is the set of input data sufficient to achieve the business goals of the system? Is the set of output data complete with respect to business and legal requirements?
5. Completeness of the functionality: Are the functional requirements complete in that all the business procedures identified in criterion 3 are supported by the appropriate functions of the software?
6. Usability of the user interface prototype: Is the prototype complete and ergonomic?
7. Correctness of the data model: Is the data model complete and consistent? Is the data model consistent with the business procedures? Does the data model comply with the best engineering practices?
8. Completeness of the non-functional requirements: Are the non-functional requirements complete in that they define the expectations related to the security of data, response time, throughput and reliability?
9. Verifiability of the non-functional requirements: Are the non-functional requirements defined verifiable (testable)?
10. Credibility of the verification: How is the coverage of business procedures by the test scenarios? Is the performance testing sufficient? How is the credibility of the testing procedures?

As can be seen from the above list of criteria and questions, the evaluation within this subject area is focused on the contents of the analytical products. The sequence of questions moves along a path: From sources of information to business model, from business model to functions and efficiency, from functions and efficiency to verification. The results of the evaluation are in good relation to the following characteristics of ISO 9126: Functionality is supported by criteria 2, 3, 4, 5, 7, efficiency by criteria 8, 9, usability by the criterion 6, reliability by criteria 8, 9 10, and maintainability by the criterion 7. Portability was not considered a significant premise at the analysis level of IACS.

4.4 Evaluation process

Input data to the quality evaluation process consisted of all the documents that had been created in the entire software development cycle. These included:

- a business model developed in the inception phase,
- early analysis model of the elaboration phase,
- the set of analysis and design models created in the construction phase,
- the code and the complete set manuals,
- test plans and test reports (plus the observation of the testing process).

Because of the incremental nature of the development, part of the documents circulated in several versions, issued in a sequence of subsequent increments.

The evaluation process was decomposed into the areas listed in section 4.1 and structured according to the set of criteria exemplified in sections 4.2 and 4.3. The analysis that was done within the context of a particular criterion was guided by the set of questions. Sometimes, a compound area was decomposed into sub-areas and the questions were stated and answered separately for particular groups of artifacts. For example, the evaluation of the criterion 1 of section 4.2 (Methods and standards) was decomposed into a review of the software process, analysis methods, design methods, implementation methods and tools, testing methods, and documentation standards.

The answers to the questions were, in general qualitative, formulated usually in terms of a ranking: good, satisfactory, bad. Quantitative metrics were also used, however, only in such cases in which the results could be meaningful to the user. An example of such a metric is the coverage of use case scenarios by test scenarios.

The evaluation report was structured in accordance with the areas and the criteria. The results of the evaluation within a particular area were concluded in the form of two sections: Risks for the project and recommendations to the project. The risk section summed up the ‘bad’ answers and related the deficiencies to the quality characteristics of ISO 9126. For example:

- The lack of functions that support certain business procedures creates the risk that the functionality of software will not be met.
- The lack of readable design models creates the risk that the maintainability of software will be low.

The recommendation section advised what to do in order to avoid the risks identified in the evaluation process and described in the previous section. An advice related to the first point above could, for example, be: Define the functionality that is missing.

5 Conclusions

This paper describes a practical method that can be used to evaluate the expected quality of software under design. The evaluation process does not refer directly to the existing standards, however, it is consistent with the basic elements of the quality models of both ISO 9001 and ISO 9126. The mechanics of the evaluation is based on a set of criteria that are decided by stating questions and finding answers to those questions. The collection of criteria is structured into a set of subject areas that exhaustively cover the set of activities that exist in the most popular software processes: Phases of the waterfall model or activities of the RUP incremental process.

The method was used successfully in evaluating the quality of software products during the development of IACS system. The evaluation was performed on behalf of the customer and not the manufacturer of the system. The criteria and questions that guided the evaluation process allowed for a systematic, in depth analysis of the deliverables of the particular development activities. As result, several risks were revealed and identified. The recommendations helped in avoiding the risks in the final product. IACS system was build and certified for use within the deadline.

The advantages of the method can be summarized as follows:

- The method does not depend on any particular software process or method that can be used in the development of software.
- The results of the method, i.e. the answers to the questions that are stated in the evaluation process, are readable and meaningful to the customer.
- Negative answers to particular questions can easily be translated into recommendations on what to change in order to enhance the quality of products.

The method is simple in use, does not relay on any historical data, and need not be supported by a computerized tool.

References

1. ISO/IEC 9126-1: Software engineering – Product quality. ISO/IEC (2001)
2. ISO 9001: Quality management systems – Requirements. ISO (2001)
3. Zalewski A., Cegieła R., Sacha K.: Modele i praktyka audytu informatycznego, in: Huzar Z., Mazur Z. (eds.): Problemy i metody inżynierii oprogramowania, WNT, Warszawa (2003)
4. Fenton, N: Software Metrics: A Rigorous Approach, Chapman and Hall (1993)
5. Horgan G., Khaddaj, S., Forte P.: An Essential Views Model for Software Quality Assurance, ESCOM-SCOPE 99 (1999)
6. Szejko S.: RDQC – sterowana wymaganiami kontrola jakości oprogramowania, in: Górski J, Wardziński A. (eds): Inżynieria oprogramowania: Nowe wyzwania, WNT, Warszawa (2004)
7. Basili V.R., Weiss D.M.: A Methodology for Collecting Valid Software Engineering Data, IEEE Transactions on Software Engineering, Nov. (1984)