An Approach to Software Quality Specification and Evaluation (SPoQE)*

Iwona Dubielewicz[1], Bogumiła Hnatkowska[1], Zbigniew Huzar[1], Lech Tuzinkiewicz[1]

[1]Institute of Applied Informatics, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
{Iwona.Dubielewicz, Bogumila.Hnatkowska, Zbigniew.Huzar, Lech.Tuzinkiewicz}@pwr.wroc.pl

Abstract. The paper discusses how to carry evaluation of software product quality within software development process. The evaluation process bases on a quality model being an instance of a quality model. Quality model, elaborated basing on ISO quality standards, may be used both for specification of quality requirements and quality assessment. The evaluation process is presented in terms of activity diagrams. It is generic and may be concretized for two perspectives of software product quality, i.e. external, and internal quality. Simple example illustrates the proposed approach.

1 Introduction

Quality of a software product can be defined as a totality of characteristics that bear on its ability to satisfy stated and implied customer needs [5].

The meaning of the term "software product" is extended to include any artifact, which is the output of any process used to build the final software product. Examples of a product include, but are not limited to, an entire system requirements specification, a software requirements specification for a software component of a system, a design module, code, test documentation, or reports produced as a result of quality analysis tasks [10].

The aim of the paper is to elaborate a generic process of software product evaluation based on current ISO standards, relating to Software Quality Assurance (SQA). The process is independent from specific software development methodology, and shall ensure software product compliance with quality requirements, moreover with required level of this quality.

SQA processes provide assurance that software products and processes in the project life cycle conform to their specified requirements by planning, enacting, and performing a set of activities to provide adequate confidence that quality is being built into the software [10].

* The work was supported by polish Ministry of Science and Higher Education under the grant number 3 T11C 06430.

Please use the following format when citing this chapter:

Dubielewicz, I., Hnatkowska, B., Huzar, Z., Tuzinkiewicz, L., 2006, in IFIP International Federation for Information Processing, Volume 227, Software Engineering Techniques: Design for Quality, ed. K. Sacha, (Boston: Springer), pp. 155–166.

ISO standard [3] provides a new supporting process called *Product evaluation*. The process is defined in informal way. The paper formalizes *Product evaluation* process and presents it in the form of activity diagrams. The diagrams, expressed in SPEM notation [9], include roles of stakeholders of software process development, and artifacts related to SQA. Considered activities are based on ISO standard [4], while artifacts are instances of our Software Quality Model of Requirement, Evaluation and Assessment (SQMREA) [6]. The fact that ISO quality models are instances of SQMREA explains why we have called SQMREA in [6] to be a quality meta-model.

The paper is organized as follows. Section 2 gives an overview of software quality generic model. In Section 3 our formalization of evaluation process is given. Section 4, on the base of a simple example, explains how the evaluation process may be instantiated. Finally, Section 5 discusses presented proposals and compares them to current literature.

2 Generic model of software quality requirements evaluation and assessment

Our proposal of software quality generic model for requirement, evaluation, and assessment, called SQMREA, is shown in Figure 1. This is an extended version of the model, presented in [6]. The model is presented as UML class diagram with a set of OCL constraints (omitted in this paper).

The reason of SQMREA introduction is to enable evaluation and assessment of quality levels of intermediate artifacts produced during software development as models, specifications etc., and, finally, the resulting software product.

In general, according to ISO standards, quality assessment can be done from three perspectives: external, internal, and in use perspective. The last perspective relates not only to a software product itself but also to its operation in a specific environment and specific context of use. Therefore, our SQMREA model takes into account only the first two perspectives, which concern only a software product. The external perspective represents a viewpoint of a user, while the internal perspective represents a viewpoint of a software developer.

The choice of quality perspective plays the key role in model instantiation. Each perspective determines a quality model, i.e. the set of selected characteristics and relationships between them. Instances of the generic model embrace not only ISO quality models for a given perspective, but also other models that are different from ISO quality models, for example, dependability models based on IEC 300 series of standards [10]. In the sequel, for the sake of brevity, we confine our consideration to ISO quality models.

To do description of the SQMREA generic model more readable, we have grouped its elements into four packages that are presented in Figure 2.

Two central packages relate to elements of quality model, and software requirements respectively. The right side package relates to subjects of quality assessment, while the left side one defines how to do the assessment.

We start with description of the package *Elements of Quality Requirement Specification*, as comprehensive requirement specification is a starting point both to devel-

opment process of a software product and to its quality requirement specification and evaluation. This specification is based on user *needs* that are informal by nature. The needs serve as the basis for the formulation of *requirements.* A requirement is defined as *a condition or feature required by user to solve a problem or to reach specific goal* [10]. The requirements should be expressed quantitatively through referring to values of software product *attributes*, i.e. *measurable physical or abstract properties of the product.*

Software Product, Artifact Specification, Need, and *Requirement* classes are abstractions concerning software quality requirements. Their instances are specific for a given software product. An instance of *Software Product* class can be associated with a set of instances of *Artifact Specification* class. In our considerations, we abstract from semantics of instances of *Artifact Specification,* and we concentrate only on associations between this class and other classes. The association to *Artifact Implementation* class reflects obvious relationship between specification and implementation – specification of an artifact may have many implementations. The association to *Requirement* class (a requirement may be decomposed into other requirements) reflects the fact that artifact implementations will be eventually evaluated in context of some requirements. The association to *Attribute* class points the attributes that are involved in artifact specification and should be also present in artifact implementation.

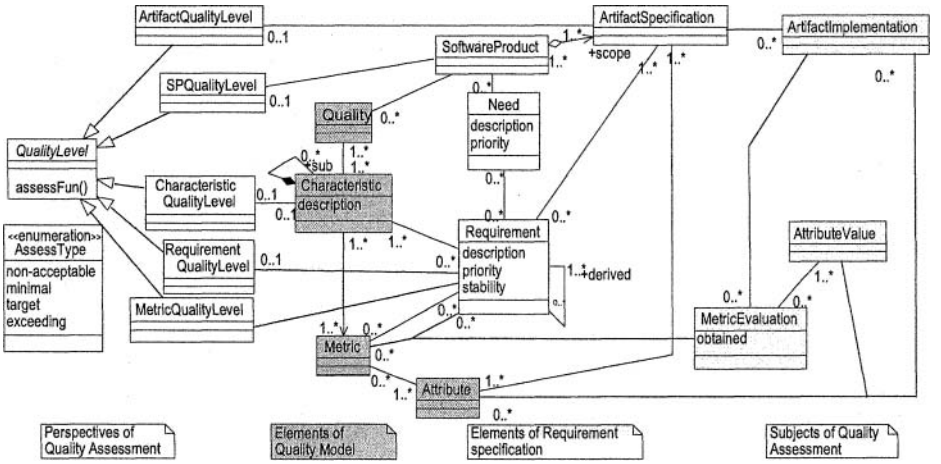

Fig. 1. SQMREA generic model

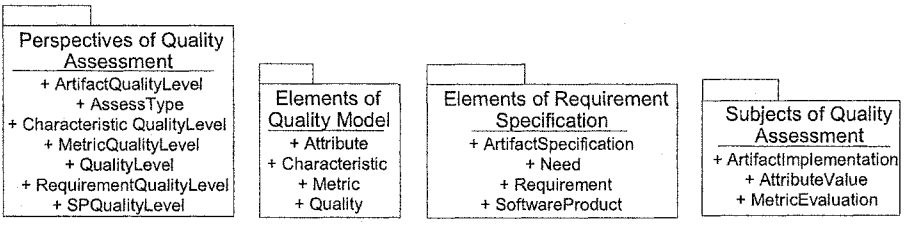

Fig. 2. Elements of main parts of SQMREA model

To do quality requirements measurable the classes of *Elements of Requirement Specification* package are associated with classes of the package *Elements of Quality Model*. The main elements of this package are: *Quality, Characteristic, Metric,* and *Attribute* classes. An instance of *Quality* class is a root of a hierarchy of characteristics and sub-characteristics, and represents a given quality perspective. Standard [1] defines the following characteristics for internal and external quality models: *functionality*, *reliability*, *usability*, *efficiency*, *maintainability* and *portability*. These characteristics may be subdivided into multiple levels of sub-characteristics. For example, according to this standard, there are the following sub-characteristics for functionality: *suitability*, *accuracy*, *interoperability*, *compliance* and *security*. For an agreed sub-characteristic a set of metrics as functions on attributes is given. Acceptable ranges of the metrics specify recommended values of attributes.

A requirement may be decomposed into other requirements. The leaves of the requirement's tree are associated with metrics by *Metric Quality Level* association class.

The elements of the left side package *Perspective of Quality Assessment* supplement the package *Elements of Requirement Specification* by delivering functions that assess quality levels for: (1) a requirement (*Requirement Quality Level*), (2) a characteristic (*Characteristic Quality Level*), (3) an artifact (*Artifact Quality Level*), and (4) a whole software product (*SP Quality Level*). The mentioned classes (in braces) are specializations of an abstract *Quality Level* class. Each specialization of *Quality Level* class should provide its own assessment function (*assessFun*). The functions yield values of *Assess Type*, i.e. non-acceptable, minimal, target or exceeding. These values define quality of a given element (requirement, characteristic, single artifact, and finally a software product, understood as a set of artifacts).

Assessment functions form a hierarchy of functions relating to requirements (the lowest level), sub-characteristics, characteristics, artifacts and software product (the highest level of the hierarchy). The elementary assessment relates to a given metric for a given requirement (*Metric Quality Level*), and is represented by a respective *assessFun* function. The function classifies the set of possible values of a given metric to one category of *Assess Type*. Other assessment functions are not elementary – they are composed of assessment functions that are at lower hierarchy level.

The values of the assessment function for *Metric Quality Level* are arguments for assessment functions of *Requirement Quality Level*. The values of the assessment functions for *Requirement Quality Level* are arguments for others assessment functions, i.e. *Characteristic Quality Level*, and *Artifact Quality Level*. Assessment of the whole software product (*SP Quality Level*) is done with regard to the results from artifact quality level assessments.

For example, assessment functions defined for i-level in hierarchy may take a form as below:

$$ass_{level(i)}(x_1, x_2, \ldots, x_n) = \min(ass_{level(i-1)}(x_1), ass_{level(i-1)}(x_2), \ldots, ass_{level(i-1)}(x_n)) \quad (3.1)$$

where $ass_{level(i-1)}(x_k) \in \{\text{Non-acceptable, Minimal, Target, Exceeding}\}$ for $k = 1,\ldots, n$, and the values are ordered linearly in the following way:

$$\text{Non-acceptable} < \text{Minimal} < \text{Target} < \text{Exceeding}$$

The composition may take different forms, for example, a given higher level assessment function may take a form of weighted sum of values that are results of lower level assessment functions.

The package *Subjects of Quality Assessment* contains the classes that represent instances of artifacts (*Artifact Implementation* class), values of their attributes (*Attribute Value* class), and metric evaluations (*Metric Evaluation* class), calculated based on attribute values.

3 Model of evaluation process

This section presents our formalization of product evaluation process, informally defined in ISO standard [4]. The formalization uses activity diagram for expressing artifacts flow among different roles engaged within the process. Artifacts used within the process are instances of our SQMREA model. Roles are selected from those, proposed in [11].

Model of software product evaluation process is presented in Figure 3.

As was mention in Section 2, the main elements of SQMREA model were divided into four packages. The names of packages are used on the activity diagram as the names of artifacts produced by different activities. This means that a given activity yields all elements from the package. To simplify the picture only the outputs for activities are presented. The output of a given activity is also an input for the following one.

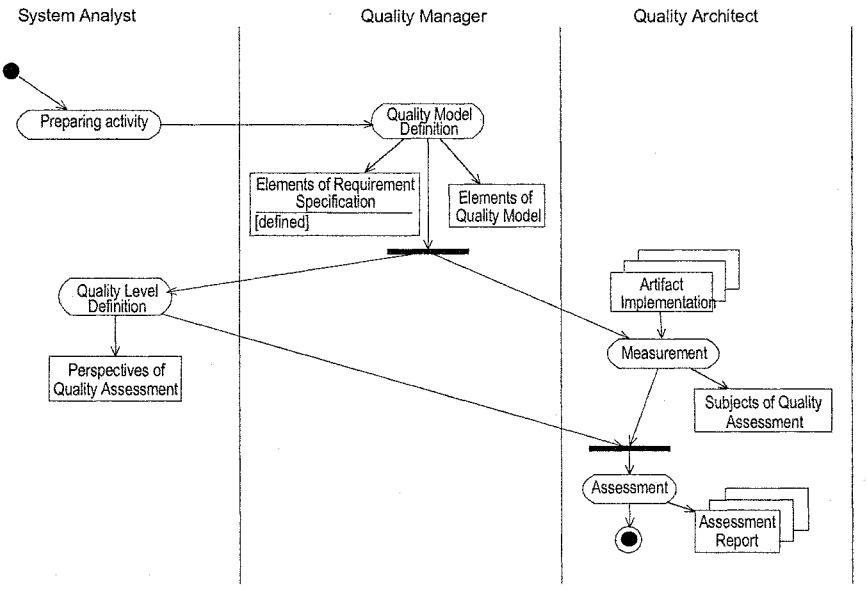

Fig. 3. Model of software product evaluation process

The process starts with *Preparing activity*. This activity can take one of two forms according to the quality perspective. In the case of external quality the activity is substituted by *Software Requirement Specification* activity, and in the case of internal quality – by *Internal Requirement Specification* activity, see Figure 4.

During *Software Requirement Specification* activity system analyst should, first of all, to determine the real purposes of the software. The purposes are expressed as needs, and they are represented by instances of *Need* class in the activity diagram. Needs are written informally, in natural language. Based on them system analyst identifies kinds of output artifacts, elicits quality requirements, and associates artifacts specifications with quality requirements.

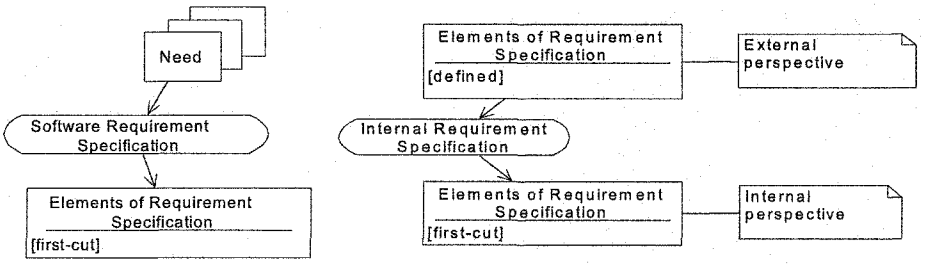

Fig. 4. Details for Preparing activity for external, and internal quality perspective

The requirements at that moment are described informally, what is marked by object-flow state called "first-cut". They take a form of system features. System feature is defined as a general system service that is associated with fulfillment of one or more needs [8]. ISO standard [4] introduces a term *general requirement* for system features description. An example feature for a weapon control system can be defined as: *in the case of attack two independent authorizations are needed* [8].

Based on general requirements system analyst decides what artifacts will be expected to represent a software product from interesting quality perspective (it proposes instances of *Artifact Specification* class). When external perspective is considered the example artifacts may be: executable components, user manual, installation manual, and so on.

Internal Requirement Specification aims with transforming external requirements into internal ones that are associated with internal artifacts. The example internal artifacts may be: software requirement specification, software architecture document, source code, and so on. Needs are omitted at this stage.

The most difficult is *Quality Model Definition* activity as it is responsible for transformation of informally defined, general requirements into formally defined requirements. The main aim of this activity is to elaborate a quality model, expressed in terms of characteristics, metrics, and attributes. These elements must be suitable for interested quality perspective. Their association with not empty set of metrics formally defines the leaves of requirement tree. Some identified attributes may contribute to more than one requirement for a software product. Some of them may be mutually in conflict, which must be resolved.

For each pair: requirement-metric an instance of *Metric Quality Level* class and an instance of *Metric Evaluation* class are created. The assessment function for instances of *Metric Quality Level* will be defined in the next activity, called *Quality Level Definition*. The attribute obtained for instances of *Metric Evaluation* class will be filled in *Measurement* activity.

Quality Level Definition activity yields assessment functions for elements which quality we want to evaluate and assess. This activity is a complex one and can be decomposed, as it is shown in Figure 5. The activity is repeated for each considered quality element we want to evaluate. First of all the assessment function for all pairs: leaf-requirement—metric must be defined. Next, following assessment functions are defined: for each leaf-requirement, for each artifact, and for the whole software product. *Quality Manager* may introduce (or may omit) definitions of assessment functions for selected characteristics from quality model.

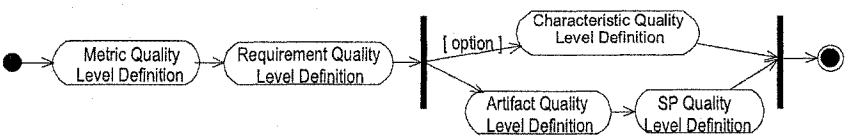

Fig. 5. Details of *Quality Level Definition* activity

The last two activities in Figure 3 are deferred up to artifact implementations are created. *Measurement* activity returns values of attributes used in metrics for any artifact implementation, and deriving from attribute values metric evaluations for each pair: leaf-requirement—metric. These evaluations are represented by *obtained* attribute in *Metric Evaluation* class.

During *Assessment* activity assessment functions, defined in *Quality Level Definition* activity are performed. Results of the functions are gathered in assessment report.

4 Example

Presented example deals with situation when at the beginning of the semester the course timetable for students' courses contains some inconsistencies. The timetable is prepared manually using data from the given external database. So, the *problem* is how to get feasible timetable within a given deadline. A software product, which supports the problem solution, is expected.

The investigation performed by system analyst with administrative staff shows that possible *reasons* of timetable incorrectness come from:

- incomplete or incorrect input data,
- no direct access to database,
- constraints received from teachers,
- temporary overloading of faculty staff at the beginning of the semester.

To resolve problem the following information needs are formulated:

- recording of teacher's constraints up to a given deadline,
- data relating to timetable should be accessible all time they are needed (*),
- preview of current assignments of lecturers and classrooms to courses during the process of timetable preparation (*).

To fulfill these needs requested software product should have the following *features*:

- system is accessible on demand,
- system enables reporting of current resource usage.

In further, we investigate an exemplary quality evaluation process conducted for the needs marked by (*). Additionally, we restrict the example only to external perspective of software product quality.

Activity 1: **Software requirement specification**
Input: User needs
Output: Requirements specification; it contains following requirements:

R1) Some kinds of analytical reports are expected; the following two of them are further considered:
Report1 – shows the vacancy of classrooms along weekdays
Report2 – shows preliminary timetable based on current assignments of lecturers and classrooms to courses

R2) Report presentation should take a format of pivot tables and pivot charts

R3) Data for reports are retrieved from the external database

R4) System is accessible for use in any time when it is needed

R5) The expected software should be implemented on Microsoft platform.
Comment: The software functionality is limited to preparation of a set of analytic reports to support current, manually conducted process of timetable preparation.

Software Product (SP) will consists of two kinds of artifacts:
- Code: Resource Planning Reports system (RPR system)
- Documentation: User manual

The RPR system will operate in conjunction with:
- DBMS (MS SQL or Access)
- Excel

Activity 2: **Quality model definition**
Input: Requirement specification, list of artifacts, and ISO quality standards [1], [2].
Output: Elements of quality model – presented in Table 1.

Table 1. Quality model for RPR system

Quality model characteristic/ subcharacteristic	**Metric/ Metric_ID**	**Measurement formula & attributes**	**Tracing for**	**Assign to**
Functionality/ Suitability	Coverage/ Fsm1	X=1-A/B A–number of function incorrect or missing B–number of function described in requirement specification	R1	Code Doc
Functionality/ Interoperability	Data exchangeability/ Fim1	X=A/B A–number of data formats exchanged successfully with other software B–total number of data formats to be exchanged	R3	Code

Usability/ Usability compliance	Usability compliance/ Ucm1	X=1-A/B A–number of usability compliance. items missing B–total number usability compliance items specified	R2	Code Doc
Reliability/ Recoverability	Availability/ Rrm1	X=A/B A–total available cases of user successfully software use when attempt B–total number of cases of user's attempt to use software during observation time	R4	Code Doc
Functionality/ Interoperability	MS software compliance/ Fim2	X=A/B A–number of Microsoft software products being used B–total number of used software products	R5	Code

Activity 3: **Quality level definition**
Input: Quality model definition for RPR system
Output: During this activity the assessment functions for all elements from assessment perspective package are established. First, the metric quality level assessment functions for any pair: requirement-metric are defined. They are shown in Table 2.

Table 2. Definition of assessment function for metric quality level

Req.	Metric_ID	Assessment function definition
R1	Fsm1	Minimal>0.80; Target=1.0
R2	Ucm1	Minimal>0.5; Target≥0.9
R3	Fim1	Minimal=0.9; Target=1.0
R4	Rrm1	Non-acceptable≤0.5; Minimal<0.9; Target=0.9; Exceeding>0.9
R5	Fim2	Non-acceptable≤0.8; Minimal<1.0; Target=1.0

We have assumed the assessment functions for all quality levels are defined according to the formula (3.1). The only difference is the formula of an assessment function for *SPQualityLevel*, and it is defined as follows:

$$ass_{SP\text{-}level}(x_{Code}, x_{Doc}) = \begin{cases} \text{Non-acceptable} & \text{if } x_{Code} = \text{Non-acceptable} \\ \text{Minimal} & \text{if } x_{Code} = \text{Minimal} \\ \text{Target} & \text{if } x_{Code} = \text{Target and } x_{Doc} \geq \text{Minimal} \\ \text{Exceeding} & \text{if } x_{Code} = \text{Exceeding and } x_{Doc} \geq \text{Minimal} \end{cases} \quad (4.1)$$

The assessment functions are accepted or given by user.

Activity 4: **Measurement**
Input: Code and documentation of RPR system.
Output: During this activity the measurement of software artifacts is performed. The resulting metric values are presented in Table 3.

Table 3. Examples of performed measurements for metric level

Metric_ID	Metric	Obtained metric values	
		Code	Documentation
Fsm1	coverage	X = 0.85	X = 0.4

Ucm1	usability compliance	X = 0.9	X = 0.8
Fim1	data exchangeability	X = 1	n/a
Rrm1	availability	X = 0.6	X = 1
Fim2	MS software compliance	X = 1	n/a

*n/a – not applicable

Comment. The measurement of SP is performed in user's environment

Activity 5: **Assessment**
Input: Results of activities 3 and 4
Output: Quality values for all levels

Using the assessment function (3.1) defined in clause 3, for each level of assessment there were obtained values given in tables 4–7. As each requirement has one associated metric only, the results of quality values for requirement quality level are the same as for metric quality level. The assessments are done independently on each level for every artifact.

Table 4. Obtained quality values for metric and requirement quality level

Req	Metric_ID	Obtained quality values	
		Code	Documentation
R1	Fsm1	Minimal	Non-acceptable
R2	Ucm1	Target	Minimal
R3	Fim1	Target	n/a
R4	Rrm1	Minimal	Target
R5	Fim2	Target	n/a

Table 5. Obtained quality values for sub-characteristic level

Sub-characteristic	Metric_ID	Obtained quality values	
		Code	Documentation
Suitability	Fsm1	Minimal	Non-acceptable
Interoperability	Fim1 Fim2	Target	n/a
Usability compliance	Ucm1	Target	Minimal
Recoverability	Rrm1	Minimal	Exceeding

Table 6. Obtained quality values for characteristic level

Characteristic	Obtained quality value	
	Code	Documentation
Functionality	Minimal	Non-acceptable
Usability	Target	Minimal
Reliability	Minimal	Exceeding

Table 7. Obtained quality values for artifact level

Artifact	Obtained quality value
Code	Minimal
Documentation	Non-acceptable

Final assessment of the RPR system quality is minimal according to (4.1) definition of assessment function for the software product.

5 Conclusions and related works

The paper formalizes and refines ISO standards relating to processes of quality evaluation and assessment [5]. It systematizes notions used for quality specification. The notions are elements of SQMREA model [6]. The developed SQMREA model is general. It enables for instantiations of different kind of quality models, not only those proposed by ISO. For example, it is possible instantiate the quality model for high dependable systems, which concentrates on such characteristics as: safety, security, usability, availability, and reliability [10]. In general, other existing quality models use different notions, but they share the same structural elements (characteristics, subcharacteristics and metrics).

The SQMREA model is our original contribution, while the model of evaluation process is a refined and formalized version of the evaluation process, presented in the series 14598 of ISO/IEC standardization documents. The new elements include definition of roles, specification of artifacts, an assignment of artifacts to roles and activities, performed by roles.

The paper also presents SPoQE methodology for software quality product evaluation and assessment. The methodology is defined in terms of SPEM notation [9], i.e. roles performing some activities on a given set of artifacts. The SPoQE methodology is independent from a software development methodology provided that the methodology distinguishes at least the following two processes, defined in [3]:

- system requirement analysis,
- software construction.

The activities of the proposed software product evaluation process conform to those proposed in [4] with only one difference:

- *measurement* activity in [4] is proceeded with planning activity – we omit this activity as it is part of a management process.

The presented quality evaluation process can be considered as an important part of *Quality Control* process within SQA. It can also be considered within CMM (fourth level) [6] as part of quality management. The SPoQE methodology concentrates on product evaluation only, and does not take into account evaluation of development process. The knowledge about the results of evaluation enables to carry out some corrective actions, and in this way can positively influence the final quality of the software product.

We have not found another approach to software product evaluation based on ISO standard [4].

It is evident that application of proposed method of product evaluation is labour-consuming. The cost-benefits analysis was not the subject of our interests. It can, and should be concern of further investigations as it is obvious that some trade-off between evaluation effort and resulting quality of software product is expected.

In further research we are going to:

- apply and validate SpoQE methodology for industrial projects,
- develop a tool supporting evaluation and assessment of a software product according to SPoQE.

References

1. ISO/IEC 9621-1:2000, Software engineering – Product quality - Part 1: Quality model
2. ISO/IEC TR 9621-2:2002, *Software engineering – Product quality - Part 2: External metrics*
3. ISO/IEC 12247:1995/Amd.1:2002, *Information technology — Software life cycle processes*
4. ISO/IEC 14598-3:2000, *Software engineering – Product evaluation – Part 3: Process for developers*
5. ISO/IEC 25000:2005, Software engineering – Software Product Quality Requirements and Evaluation (SQuaRE) – Guide for SQuaRE
6. Dubielewicz I., Hnatkowska B., Huzar Z., Tuzinkiewicz L., Software Quality Metamodel for Requirement, Evaluation and Assessment, ISIM'06 Conference, Prerov, 2006, Czech Republic, Acta Mosis No. 105, pp. 115–122.
7. Jalote P., CMM in Practice: Process for Executing Software Projects at Infosys, Boston, Addison-Wesley, 2000
8. Leffingwell D., Widrig D., Zarządzanie wymaganiami, (in Polish) WNT, 2003
9. Software Process Engineering Metamodel Specification (SPEM), version 1.0, OMG 2002
10. SWEBOK, Guide to the Software Engineering Body of Knowledge, 2002
11. Unhelkar B., Process Quality Assurance for UML-Based Projects, Addison-Wesley, 2002