TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computing Science

MASTER'S THESIS

Product Software Quality

By

Dipl.ing. Gregor Panovski

Eindhoven, February 2008

Preface

The MSc. study of “Computer Science and Engineering” at the Eindhoven University of Technology (TU/e) is concluded with master thesis project. This research was conducted at the LaQuSo (Laboratory for Software Quality) in Eindhoven.

In the first place, I would like to thank Eindhoven University of Technology and LaQuSo for giving me the opportunity to graduate in such interesting domain. I would like to thank Petra Heck and Alexander Serebrenik for their supervision, advices, comments on the thesis, pleasant cooperation and feedback. I thank to Marko van Eekelen for his practical advices and participation in graduation committee, Natalia Sidorova for participation in graduation committee, Jovan Pehcevski and Jasen Markovski for their readings and comments on my thesis, and all other employees of LaQuSo, for the wonderful time and nice working environment at LaQuSo.

Finally yet importantly, I would like to thank my family, especially to my daughter Mia and my wife Natasha and to my parents Ljiljana and Vladimir for their, support and encouragements.

Abstract

Product Software is a commercial software package, containing a software application and accompanying documentation. Software Quality is defined as conformance of the produced software to stated and implied needs [ISO/IEC 9126-1]. In order to understand and measure quality, scientists often built models of how quality characteristics relate to each other. A *quality model* is a set of quality characteristics and relationships between them, which provides the basis for evaluating product software quality and specifying its requirements [ISO/IEC 9126-1].

Evaluation of product software quality is the topic of this M.Sc. project at the Eindhoven University of Technology conducted at the Laboratory for Quality Software (LaQuSo). The project is focused on evaluation of *external quality*, which means assessing the behavior of product software when it is executed. In this thesis, we present our experiences and guidelines for evaluating quality of product software applications from different application domains.

The major research question we addressed is how one should evaluate external product software quality. As quality is known to be a complex and multidimensional, subject to many constraints, it follows that quality evaluation is a complex process as well. Hence, to fully address this question, the following related sub-questions are considered:

- Is product software quality domain dependent?
- How can we use the ISO/IEC 9126-1 quality model in different domains?
- What are the differences between product software quality and tailor-made software quality?

Current quality models such as ISO/IEC 9126 contain numerous metrics and their full usage requires significant evaluation effort per product. Accordingly, we focus on a subset of metrics that are relevant for chosen application domains and to evaluate quality with the relevant metrics only. We also conduct a survey contacting the software producers in the Netherlands asking them which ISO/IEC sub-characteristics are important for their product software. We analyzed the results, but the response was not sufficiently high to perform a relevant statistical analysis. We believe that the industrial response was limited due to marginal use of the ISO/IEC 9126 standard in the industry.

As a starting point in the quality evaluation, we divided the software products in three categories: *infrastructure software*, *software development tools,* and *application software* [OECD]. Further, we executed product analysis in order to define which quality sub-characteristics are relevant for specific and related products classified in the same category. The reason for this was to reduce the evaluation effort focusing on relevant characteristics per category only. To create category specific quality models, we departed from the ISO/IEC 9126 standard and made use of the methodology proposed in [Botella] for building ISO/IEC 9126-based quality models. The methodology consists of several steps and the basic idea is to derive domain specific metrics starting from ISO/IEC 9126 quality characteristics. In our work, we decomposed each of the relevant ISO 9126 sub-characteristics in more concrete entities, called *attributes* and proposed *metrics* for these attributes. As a guideline for metric definition, we used the external metrics provided by [ISO/IEC 9126-2]. The

first part of the metrics was literary taken from [ISO/IEC 9126-2], the second part of the metrics was inspired by the [ISO/IEC 9126-2] metrics and derived for the specific product, while the third part were metrics not defined by the standard but related to the application domain of product software.

Using the above methodology, we analyzed seven product software examples from the three listed categories and completely evaluated four product software examples: two examples of application software and two software development tools. Different ISO 9126 characteristics were relevant for the three product software categories. For example, functionality is very important for all the three categories but portability is not important for any of them. For the other three ISO 9126 characteristics (usability, reliability and efficiency), we have discovered significant differences between product software belonging to different categories. Usability is very important for application software products, but it is less important for software development tools and infrastructure software. Reliability and efficiency, on the other hand, are very important for infrastructure software but less important for the other two categories. With this analysis, we prepared reduced quality models per category. Using these reduced quality models on our evaluated product software examples we reduced software evaluation time to one week per product software, compared to the time of more than one month per product when using full quality models.

We focused on the metrics provided by [ISO/IEC 9126-2]. We found that several [ISO/IEC 9126-2] metrics require presence of internal information and documents such as requirements specification or the number of faults during development. These documents may not be available for external evaluation, as they contain company confidential information. Hence, we expect this subset of [ISO/IEC 9126-2] metrics to be used for assessment by an internal evaluator only. Other [ISO/IEC 9126-2] metrics, such as the efficiency metric *Throughput* and security metric *Access controllability*, are too general, so the evaluator should refine or translate them to metrics specific for the product. Finally, the third group of [ISO/IEC 9126-2] metrics can be widely applied in different domains.

We conclude that external product software quality is domain or category dependent. We created reduced quality models based on ISO 9126 that can be applied per product software category. Using these reduced quality models, we decreased the evaluation effort per product software. As a proof of concept, we evaluated the external quality of four product software applications from different domains.

Metrics provided by [ISO/IEC 9126-2] can be used as a starting point for metrics definition, but in our opinion, they are not "ready to use". Thus, the evaluator should adapt them to the product category or to the domain and business objectives.

Table of Contents

1. Introduction

This thesis reports is the final step of the "Computer Science and Engineering" graduate program at Eindhoven University of Technology (TU/e). The thesis investigates "Product Software Quality" and was conducted at the Laboratory for Quality Software (LaQuSo), activity of the Faculty of Mathematics and Computer Science at TU/e.

LaQuSo aims to measure quantify and predict the quality of the designed software in order to increase the predictability of the development process. The focus of LaQuSo is on quality of software. The development process is not the subject of this study, but only the output of the development process.

This thesis project was related to LaQuSo certification services, where LaQuSo certifies software artifacts as an independent evaluator. The architecture of LaQuSo certification is presented on the figure below:

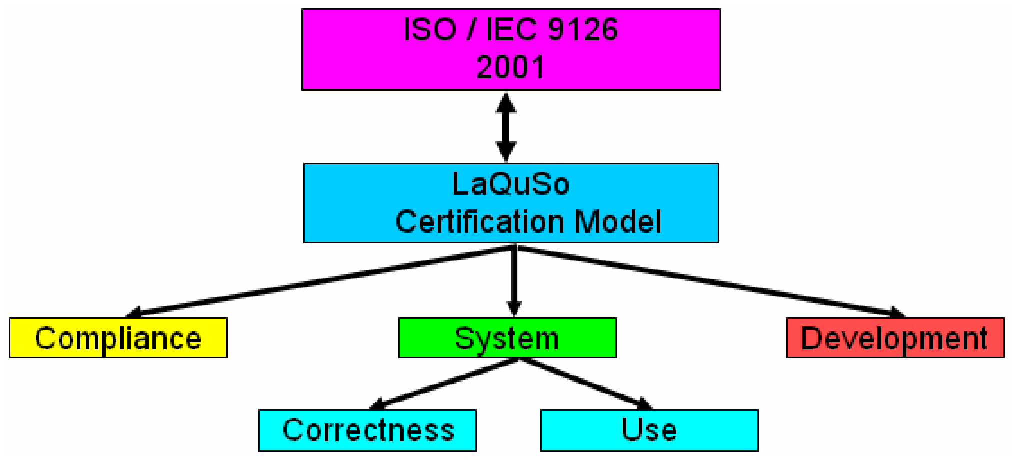


Figure 1 LaQuSo certification architecture

The project falls under "validation (empirical) methods, techniques and tools" in LaQuSo competences, presented on LaQuSo web site on following URL: http://www.laquso.com/research/researchcertification.php.

Research Questions

This document reports on a study of: Product Software Quality, Quality models and Product Software.

In this project, we aim at researching the product software quality assessment, and argue the need of a quality model as a first step in the software quality assessment. A quality model is a set of quality characteristics that should be evaluated. We expect that the relative importance of the quality characteristics could be domain-dependent, where for some domain one characteristic may be very important, while for other domains the same characteristic may be irrelevant.

We address the following research questions:

- How do we measure product software quality?

  Software quality in general is hard to define and therefore hard to measure.
- Is product software quality domain dependent?

  We believe that different quality characteristics are important for different application domains. Our research should be able to endorse or reject this conjecture.
- How can we use the ISO/9126-1 quality model in different domains?

  ISO/9126-1 is too general in order to give coverage for all application domains. We expect that for some domains a reduced set of ISO/9126-1 can be used for assessing the quality. Using a reduced set of characteristics, the assessment and verification effort can also be reduced.
- What are the differences between product software quality and tailor-made software quality?

  We want to compare which quality characteristics are important for the tailor-made software.

Our expectation is that for different domains we can define different *domain-based quality models*. These domain based quality models will be based on the ISO/9126-1 quality model, but we expect that they will not be identical to the ISO/9126-1 model.

Research Methods

During this project, we used the following research methods:

- Literature study

  A literature study involves reviewing readily available scientific papers related to the research area. We conducted literature study in the areas of Quality, Software Quality Models and Product Software.
- Conducting a survey

  This method had several parts:
  - Designing a questionnaire required for executing structured interviews. We prepared a questionnaire and enclosed letters according to the theory of questionnaire design [Litkowski]. The complete version of the questionnaire is available in the Appendix A.
  - Personal interviews with different stakeholders

    We also used the web and library, scientific and industrial resources for interviewing [Litkowski]. These resources helped in conducting structured interviews.
  - Analysis of the survey

    The results of the interviews were further analysed. On the basis of the interviews, we tried to extract information which quality characteristics are important for the specific product software and for the specific application domain. Due to lack of insufficient feedback, our survey did not provide us with statistically significant results.

- Quality evaluation and construction of a quality model for real product software applications

  This step will help in assessing relevance of the quality model on a real software application. As a starting point, we executed domain analysis in order to define which quality sub-characteristics are relevant for a specific product and to reduce the evaluation effort focusing on relevant characteristics only. To this end we departed from the ISO/IEC 9126 standard and made use of the methodology_proposed in [Botella] for building ISO/IEC 9126-based quality models. The methodology consists of several steps and the basic idea is to derive metrics starting from ISO 9126 quality characteristics. In our work, we derive attributes for all relevant ISO 9126 sub-characteristics and propose metrics for these attributes. As a guideline we used external metrics provided by [ISO/IEC 9126-2]:_some metrics were literary taken from [ISO/IEC 9126-2], other were inspired by the [ISO/IEC 9126-2] metrics, while the third part were metrics not defined by the standard but related to the application domain of the product software.

Report Outline

The rest of this thesis is organized as follows. Chapter 2 explains the notion of product software and discusses the place of product software in the software market. Chapter 3 focuses on quality, software quality, and quality models; the chapter describes related terms and previous research. Chapter 4 discusses the related research. In Chapter 5, we present the survey results and the analysis of these results. Chapter 6 presents the evaluation procedure of four product software applications. Chapter 7 contains reflections and conclusion remarks.

2. Product Software

There is no clear distinction between the terms *product software* and *software product*. We find it important that the term is perceived based on the provided definition. For this research, we will give preference to the term *product software* because this is the terminology used by the researchers in the Netherlands. We will use the following product software definition [Xu]:

***Product software** is defined as a packaged configuration of software components or a software-based service, with auxiliary materials, which is released for and traded in a specific market.*

This definition contains the following terms/concepts. **Packaged software components** mean code, executables and web pages that can be obtained ready from software vendors and do not require much customization. **Software-based service** means commercial software services. **Auxiliary materials** refer to the accompanying software documentation (user manuals and brochures. **Release** and **trading** give the commercial values of the product software.

Another related term is a software product. We cite the definition from the ISO/IEC 9126-1:2001 standard originally published in ISO/IEC 12207:1995:

***Software product** is the set of computer programs, procedures, and possibly associated documentation and data.*

This definition seems too general and means that every running program can be considered as a software product. It defines another term and has other meaning compared to the definition from [Xu]. We will mainly use the definition from [Xu]; we presented the ISO, because it is related to ISO/IEC 9126-1 quality model.

[Lassila] provides another definition of software product:

***Software product** is the application that is productized and can be customized to suit the customer needs by customization.*

This definition is closer to the definition of [Xu], because it assumes that the application is a product (productized) unlike the ISO definition where the software product represents a set of programs and procedures alike. On the other hand, [Lassila] definition differs from [Xu] definition, because [Lassila] assumes customization to suit the customer needs. Therefore, [Lassila] cannot make a clear differentiation between product software and tailor-made software. This fact is also visible on Figure 2. Further, [Lassila] definition appears simpler and less vague, assuming that the reader has an understanding of the term *productized*.

In this report, we use the term *product software*. However, given that the definition from [Xu] is somewhat vague, we simplify the meaning of this term by providing the following definition:

Product Software is a commercial software package, containing a software application and accompanying documentation.

The difference between our definition and the definition from [Xu] is that services are not included in our definition. This is because we do not consider services as a product software.

We should be able to make distingtion between the product software and other software types. Rough division of software types in three groups is presented on the following figure [Lassila]:

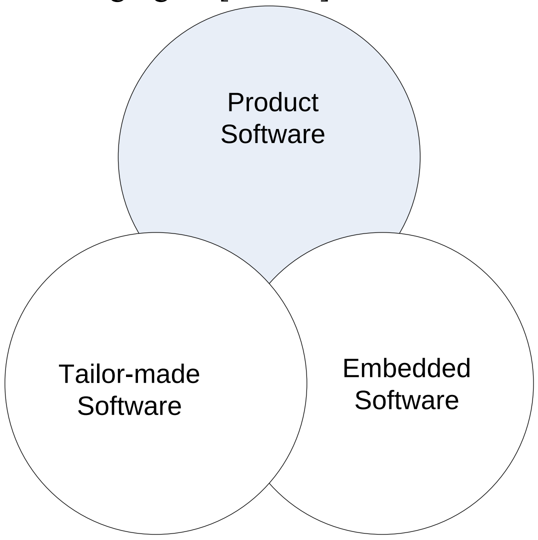


Figure 2 Categories of Software offer [Lassila]

This classification is based on the [Lassila] definition of software product, where authors have not made a clear distinction (border in the figure) between software product and tailor-made software, resulting in an intersection between the two, as shown in figure 3.

In our study, we mainly focus on the category Software Products. [Xu] mentions three main differences between product software and tailor-made software:

1. Product software introduction to the market might need coordination of dependable software engineering activities, like market analysis. Thus, product software is market oriented, while for the tailor-made software we are usually working for only one customer.
2. Product software might require installation and usage of different hardware and software platforms. Tailor-made software is used on only one software and hardware platform.
3. Product software vendor stays owner of the software and the accompanying (auxiliary) materials, and the users of the product software should pay a license for its usage. In case of tailor-made software, the users usually own the software.

Embedded software differs from other software categories because its main role is interaction with the physical world [Lee]. Embedded software usually runs on systems that are not complete computer systems, such as airplanes, mobile telephones, audio equipment, robots, appliances, toys, security systems, pacemakers, heart monitors, weapons, television sets, printers, and manufacturing systems.

Product software terms and categories

The literature distinguishes several product software related terms. [Xu] explains the differences between these terms:

- Shrink-wrapped software is software on boxed, shrink-wrapped mediums. This kind of software is sold in the stores.
- COTS software is developed for a whole market. COTS software can be used as it is, or partly personalized within the boundaries of the application to be modified without changing its original functionality.
- Packaged software describes ready-made software products that can be obtained from software vendors, requiring little modification or customization in order to be used. This term is widely used in the literature with the meaning similar to product software.
- Commercial software is software that should be bought or licensed before it can be used.
- Standard software is routinely installed by Information Technology (IT) staff on most computers within the organization. Standard software usually contains an operating system and accompanying applications.

The following figure presents the relationship between the product software terms:

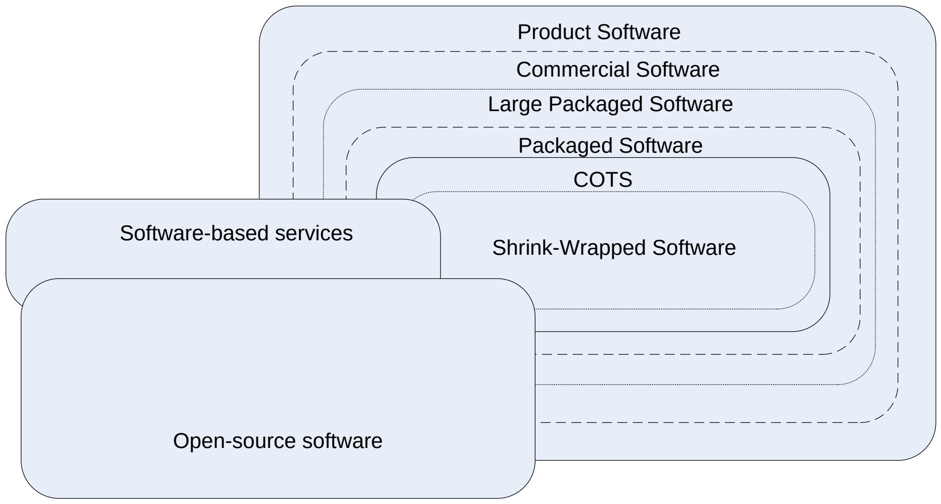


Figure 3 Product software terms [Xu]

[Xu] mentions several product software classifications defined in the literature. These classifications are based on the application domain, the architectural style used, and the programming languages. They give preference to the [OECD] classification, where product software is divided in the following categories:

- System infrastructure software (Operating Systems, middleware and security software)
- Software development tools (database management systems, development environments, development life-cycle management)
- Application software (ERP systems, CAD/CAM/CAE and other applications)

Product software summary

In this chapter, we defined product software related terms used in the science and industry. We also introduced the categorization of product software proposed by [OECD]. Defining product software terms and [OECD] categorization is important for the reader, because in the next sections we will refer to these terms and to the categories of [OECD]. In the next chapter, we will continue explaining and defining terms related to quality, software quality and quality models.

3. Quality and Quality Models

Product quality and software quality have been defined by many authors. In this chapter, we present a number of quality definitions in order to provide a clear picture s about quality related terms.

Quality

ISO 8402 provides the following definition cited in the ISO quality related documents:

The totality of features and characteristics of a product or service that bear on its ability to satisfy specified or implied needs.

Quality has been intensively studied in the past. In the following paragraphs, we will summarize the various contributions on quality, its views and insights.

[Crosby], one of the revolutionary and best selling books about quality, claims that investing in quality means zero cost for the company and this investment can only bring money. He also introduces the “Zero Defects” rule as the only acceptable performance. This is an interesting statement that gives a kind of perfection or excellence dimension to the term quality.

[Garvin] defines the following quality views:

- The **transcendental quality view** means excellence or elegance that can be recognized but not defined.
- The **user-based view** can be summarized as “fitness for purpose i.e. how much the product meets the user needs”.
- The **manufacturing quality view** means conformance to the specification. In software industry, this means conformance to requirement specifications.
- The **product-based view** is an economist view and it considers product quality characteristics and their impact on costs - the higher the quality, the higher the costs (i.e. Rolls Royce in the automotive industry).
- The **value-based view** means that quality depends on the amount that the customer is willing to pay (i.e. Ford Escort in the automotive industry). Similarly as in the software industry, here the quality can be constrained by cost (i.e. people, time and tools).

Garvin also stresses that different people in different areas (like philosophy, marketing, and operations management) perceive quality differently.

[Gillies] presents the following insights about quality:

- Quality is **not absolute**, unlike its physical characteristics i.e. temperature quality cannot be measured on a quantitative scale.
- Quality is **multidimensional**, meaning that many factors define and influence the quality.
- Quality is subject to **constraints,** i.e. by means of costs or resources.
- Quality is about **acceptable compromises**, meaning that sometimes some quality attributes may be rejected in favor of other ones.

- Quality **criteria are not independent,** i.e. quality criteria interact with each other, possibly causing conflicts.

These above statements provide a nice description of quality and explain that the assessment of quality is a complex process.

Software Quality

[Ince] provides the following statement about software quality:

"A high quality product is one which has associated with it a number of quality factors. These could be described in the requirements specification; they could be cultured, in that they are normally associated with the artefact through familiarity of use and through the shared experience of users; or they could be quality factors which the developer regards as important but are not considered by the customer and hence not included in the requirements specification"

This is an interesting statement explaining that only few quality factors can be mentioned in the requirements specification. Here we also see a differentiation between quality views and stockholder's views as mentioned by [Garvin], where the requirements specifications are covering the manufacturer quality view only. However, as [Ince] observes, the stakeholder's views do not completely cover the quality requirements.

We use the following software quality definition [Fitzpatrick]:

Software quality is the extent to which an industry-defined set of desirable features are incorporated into a product so as to enhance its lifetime performance.

We have chosen this definition because it mentions existence of a product that relates it to our research. Another reason to use this definition is that it focuses on the timely dimension of quality.

Quality Models

In order to understand and measure quality, scientists have often built models of how quality characteristics relate to each other [Kitchenham]. So far, the scientists have prepared many models intending to cover the entire software development. In this report, we mention a number of important quality models.

As a starting point, we quote the definition of quality models from [ISO9126]:

A ***quality model*** *is the set of characteristics and the relationships between them, which provide the basis for specifying quality requirements, and evaluating quality.*

Initial quality models were developed in mission critical application domains (Air Force Systems, Rome Air Development Center and NASA).

The first published method for Software Quality Evaluation is [Rubey], which proposes quality attributes and metrics. The paper further considers external factors that have impact on the program's performance. Using the metrics and the external factors, the authors also propose a quality model that can be used both for program quality and programming environment quality evaluation.

McCall Software Product Quality Model

[McCall] proposes one of the first structured quality models. Some authors consider McCall's model as the first and the most used quality model [Fitzpatrick]. [McCall] proposes the following framework:

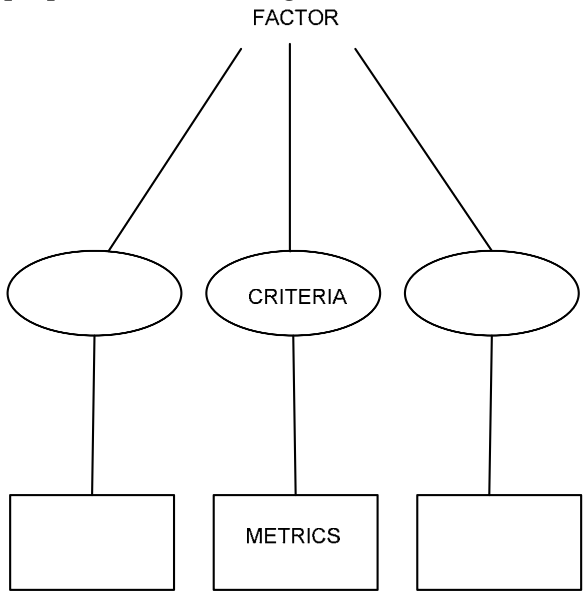


Figure 4 Software Quality Framework [McCall]

On the highest level, the major aspects or factors are specified. The framework assumes that these factors represent the management or customer view of product quality.

This model mentions the following software quality factors:

- Correctness
- Reliability
- Efficiency
- Integrity
- Usability
- Maintainability
- Testability
- Flexibility
- Portability
- Reusability
- Interoperability

On the middle level [McCall] places the attributes that provide the characteristics for the factors. Few criteria are defined for each factor. For example, *Access control* and *Access audit* criteria are defined for the Integrity factor.

The first two levels of the [McCall] quality model are presented on the following figure:

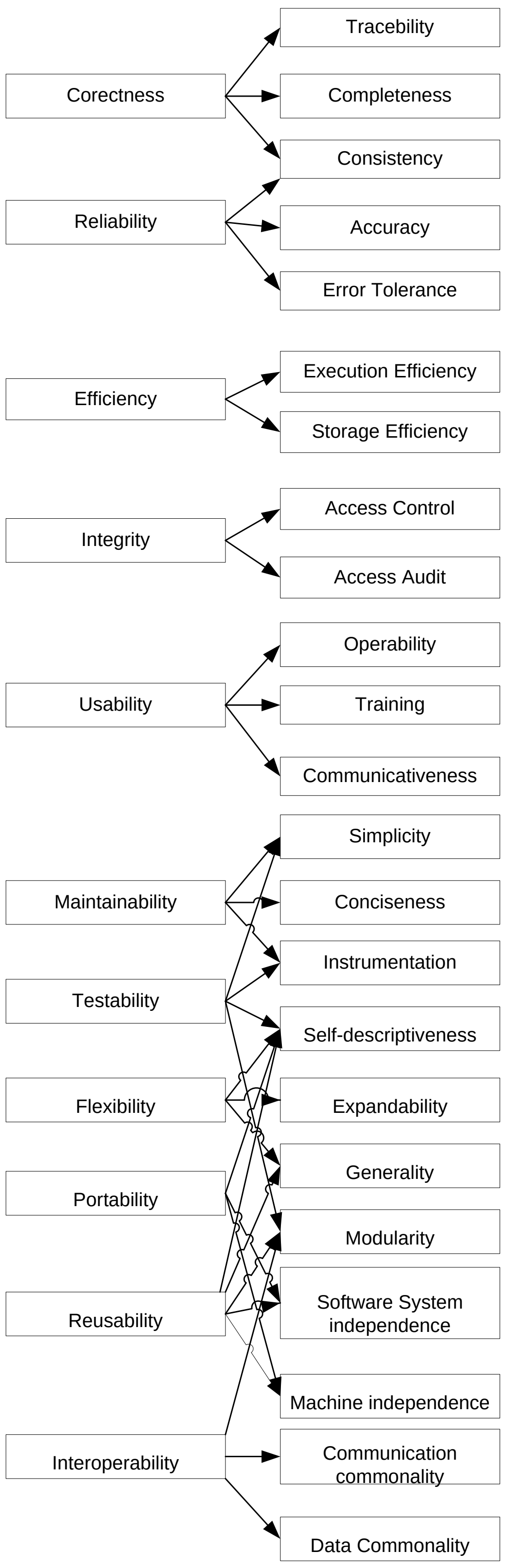


Figure 5 McCall quality model hierarchical representation

On the lowest level [McCall] places the software quality metrics that should measure the software attributes.

According to some authors, the main idea behind McCall's model is assessment of relationships among external quality factors and product quality criteria [Ortega]. The external quality is related to the product and is measured by the customers, while the internal quality is quality experienced during the development and it is measured by the programmers [Kent].

Further, McCall organizes these factors in three categories based on the uses of a software product. His classification is presented on the following figure:

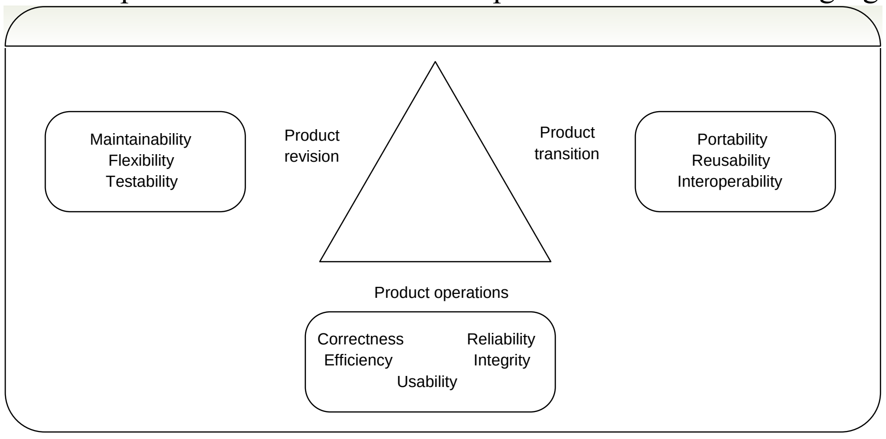


Figure 6 McCall Quality Model [McCall]

Product operation refers to the product's ability to be understood, to be stable and functional. Product revision is related to error correction and system adaptation. Product transition is related to portability characteristics assuming rapidly changing hardware.

Boehm Software Product Quality Model

[Boehm] considers the code as realization of requirements and design. The authors assume that software quality can be defined as a function of the metrics' values. After proposing a number of metrics for some of the program characteristics, they conclude that considering a value of a single metric rather than of the entire collection of metrics may be advantageous. This is because major quality characteristics often conflict, e.g. efficiency conflicts with portability.

The proposed model has the model of McCall as a starting point. The authors added a number of additional characteristics stressing the importance of maintainability of a software product. Boehm's model is based on a wider range of characteristics than the McCall's model and incorporates 19 criteria, including characteristics of hardware performance, which are missing in the McCall model [Ortega].

The model of [Boehm] further defines a hierarchical set of quality characteristics; the quality model is presented on the following figure:

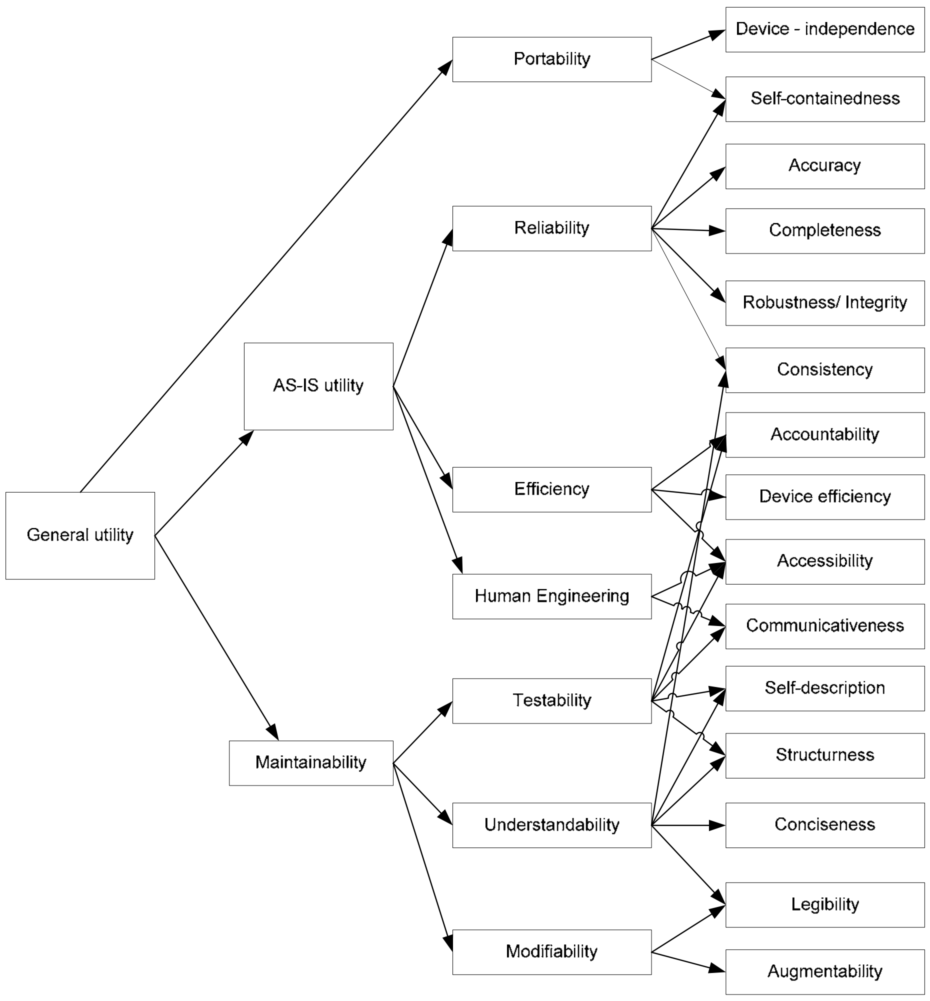


Figure 7 Software Quality Characteristics Tree [Boehm]

The second level of hierarchy is divided by considering the following three questions:

- How well can the software product be used? Authors name these characteristics *as-is utility.*
- How easy can the software product be maintained? Authors name these characteristics *maintainability.*
- Can the software product still be used in case of change of the environment? Authors name this characteristic *portability*.

These three high-level characteristics are associated with a set of lower level characteristics. Some of the low-level characteristics are related to more than one high-level characteristic. Therefore, the model is not purely hierarchical, or according to [Gilies] the model is represented as an extended hierarchy where quality attributes are sub-divided.

FURPS model

The FURPS model has been proposed by Robert Grady and Hewlett-Packard Co [Grady]. The model uses five characteristics, its name being derived from these characteristics: Functionality, Usability, Reliability, Performance, and Supportability.
The model decomposes characteristics into two categories of requirements:
-Functional requirements: Defined by input and expected output.

-Non-functional requirements: Usability, Reliability, Performance, and Supportability.

According to [Grady], the FURPS model should be applied in two steps: first, priorities should be set, where measurable quality attributes should be defined. [Grady] notes that setting priorities is important given the implicit trade-off between the characteristics (improving one quality characteristic can deteriorate another quality characteristic). One disadvantage of this model is that it fails to consider the software product's portability.

ISO/IEC 9126:1991

ISO 9126 defines product quality as a set of product characteristics [Ortega]. The first quality model version was proposed in 1991 and it is known in the literature as ISO 9126:1991. The first version had six main characteristics (functionality, reliability, usability, efficiency, maintainability and portability) and 20 sub-characteristics.

The model seems more structured than the previous models. One advantage of this model is that it is completely hierarchical: every characteristic has a set of sub-characteristics as presented on the following figure:

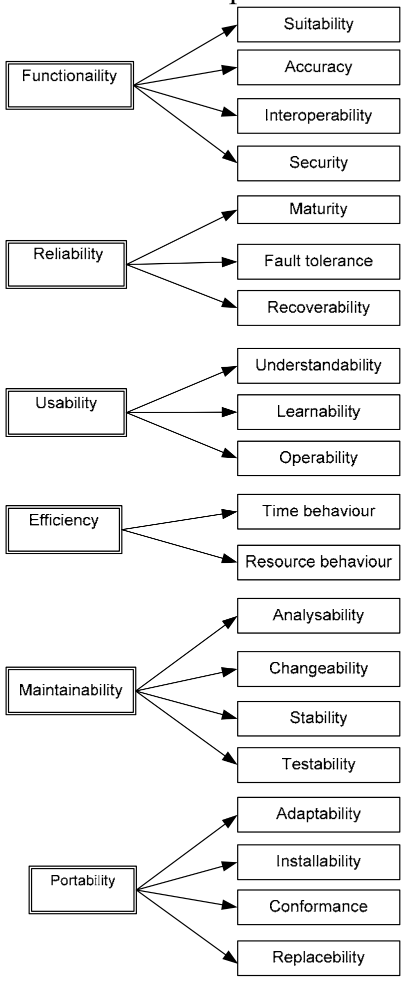


Figure 8 ISO/IEC 9126:1991 [Kitchenham]

Some authors such as [Dromey] consider ISO 9126:1991 as being derived from the Boehm model. That statement is partly true, but unlike the Boehm's model, sub-characteristics in ISO 9126:1991 are hierarchical, thus related to only one high-level characteristic.

Dromey Quality Model

[Dromey] proposed a model that extends ISO 9126:1991. Dromey's model consists of eight high-level quality characteristics, the same six from ISO 9126:1991 plus Reusability and Process Maturity.

[Dromey] presents a process of building a product quality model. The author divides the model building process in the following tasks:

- The first task addresses the users' requirements of the model.
- The second task defines the architecture of the model.

He proposes the following architecture of the model:

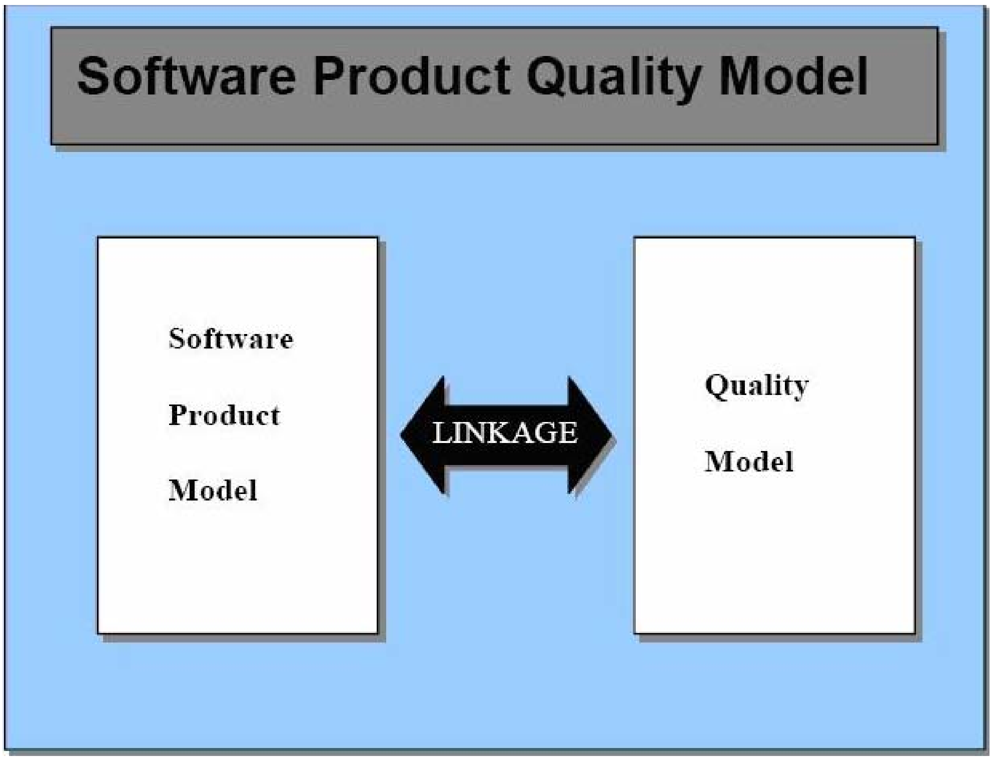


Figure 9 Software Product Quality Model Architecture [Dromey]

Based on this architecture, he divides the further work in three parts:

- Constructing a software product model
- Constructing a quality model
- Linking the software product and quality models to a software product quality model

[Dromey] searches for relationships between the characteristics and the sub-characteristics of quality. He also attempts to pinpoint the properties of the software product that affect the characteristics of quality [Kececi]. We have a similar approach as Dromey, because we also believe that quality is category or domain specific.

The disadvantage of the Dromey model is associated with Reliability and Maintainability, since it is not feasible to judge both these attributes for a system before it is actually operational in the production area.

ISO/IEC 9126:2001

ISO/IEC 9126:2001 is industrial standard proposed for quality evaluation.

In 2001 ISO prepared an updated version of the ISO/ IEC 9126:1991 standard also known as ISO/IEC 9126:2001.

The ISO/IEC 9126-1 quality model is presented on the following figure (from [ISO 9126]):

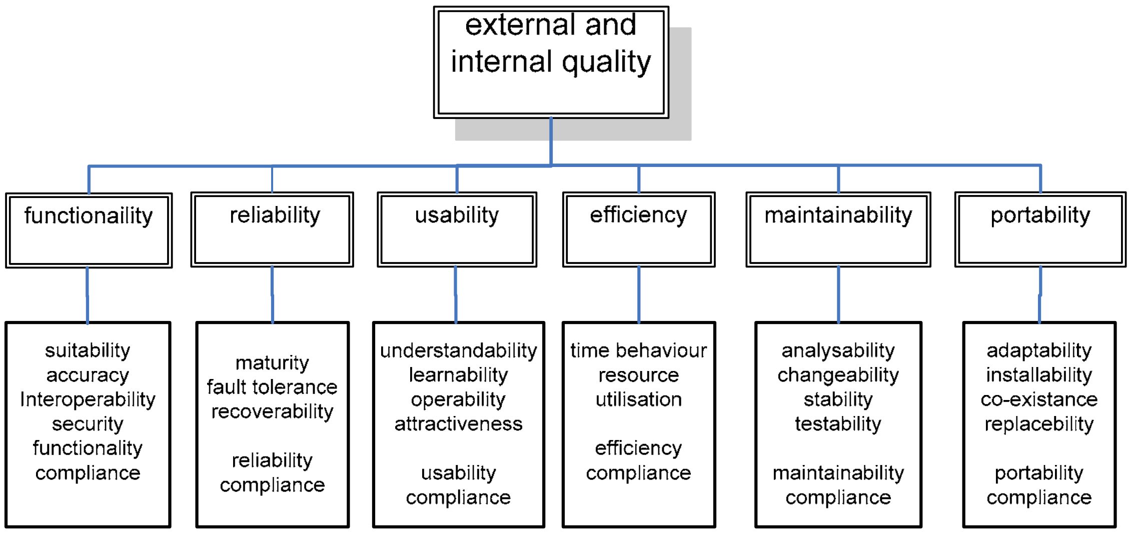


Figure 10 ISO/IEC 9126:2001 quality model [ISO9126]

As shown on Figure 8, ISO/IEC 9126-1 contains six main quality characteristics: functionality, reliability, usability, efficiency, maintainability and portability. Every characteristic contains sub-characteristics; there are in total 27 sub-characteristics (suitability, accuracy,…,replaceability, portability compliance). More details and definitions of ISO/IEC 9126-1: 2001 are available in the Appendix section.

Similarly to the ISO/IEC 9126:1991 standard, ISO/IEC 9126:2001 is also hierarchical, so every high-level characteristic has a number of related sub-characteristics. ISO/IEC 9126:2001 contains seven more sub-characteristics than ISO/IEC 9126:1991, six of them are compliance characteristics and the seventh is attractiveness, which was not part of ISO/IEC 9126:1991.

New Generation of ISO/IEC Software Product Quality Standards

ISO/IEC decided to prepare a new generation of software product quality standards in order to repair the imperfections of ISO/IEC 9126 standard. [Suryn] and Sawyer mention several imperfection of ISO/IEC 9126:

- The standard does not tackle quality requirements specification;
- Consistency with other ISO standards published in parallel;
- Scope of applicability, addressing quality needs in system life and user guidance for various users and methodology for applying quality engineering instruments within the standard.

The new standard had a working name SQuaRE or Software Product Quality Requirements and Evaluation. The new generation working group has the following guidelines:

- merging separate series ISO/IEC 9126 and *ISO/IEC 14598 Software engineering - Product evaluation* in a harmonised one,
- introducing new organization of the standard,
- introducing a new reference model,
- introducing detailed guides,
- introducing of a standard on Quality Requirements,
- introducing of guidance on the practical use of the series with examples,
- coordination and harmonization of the measure model with *ISO/IEC 15939 Software Engineering - Software Measurement Process*.

The new standard SQuaRE consists of 14 documents grouped under five thematic headings:

- Quality Management, defining all common models, terms and definitions referred to by all other standards in the SQuaRE series,
- Quality Model, probably updated version of ISO/IEC 9126-1
- Quality Measures, derived from ISO/IEC 9126 and ISO/IEC 14598,
- Quality Requirements, standard for supporting the specification of quality requirements, and
- Quality Evaluation, providing requirements, recommendations and guidelines for software product evaluation.

Comparison of the Quality Models

The following table from [Ortega] compares characteristics of different quality models. The table illustrates the characteristics and their updates during the last 30 years. ISO 9126 in the table is based on revision from 1998, which is version between ISO/IEC 9126:1991 and ISO/IEC 9126:2001.

Quality Characteristic	**Boehm**	**McCall**	**FURPS**	**ISO 9126**	**Dromey**
Testability	X	X		X	
Correctness		X			
Efficiency	X	X	X	X	X
Understandability	X		X	X	X
Reliability	X	X	X	X	X
Flexibility		X	X		
Functionality			X	X	X
Human Engineering	X				
Integrity		X		X	
Interoperability		X		X	
Process Maturity					X
Maintainability	X	X	X	X	X
Changeability	X				
Portability	X	X		X	X
Reusability		X			X

Table 1 Comparison between the quality models [Ortega]

In the above table, it is visible that all quality models score more or less equally well. Examples about model specific characteristics are "Human Engineering" and "Changeability" for Boehm model and "Process Maturity" for Dromey model, however we do not consider these model specific characteristic too relevant, therefore we will not pay attention in our next section of the report.

Quality and Quality Models Summary

In this chapter, we described main quality and software quality terms. Further, we provided historical overview of quality models.

ISO/IEC 9126:2001 is an international and widely recognized standard in the IT society [Cote]. However, some authors [Pfleeger] note that ISO/IEC 9126 is mainly used by academic and research institutions and only marginally used in the industry. We believe that ISO/IEC 9126:2001 can be used as a basis for domain-based software quality model, because ISO/IEC 9126:2001 is internationally approved standard and contains hierarchical organization of characteristics and sub-characteristics. Another argument for ISO/IEC 9126:2001 is that it is the latest introduced model, containing the experiences from previous models and providing basis in software quality.

Domain-based software quality models can be seen as sub-models of ISO/IEC 9126:2001, including quality characteristics relevant for the application domain, but also including domain-specific sub-characteristics and metrics. In chapter 6 we will try to design category-based quality models, applicable for categories of products defined by [OECD].

4. Related Research

Introduction

During the literature study, we found many scientific papers related to the Quality and Software Quality Models. In this chapter, we present several of them that seem relevant for our project.

Korea University

Scientists from Korea University executed a similar ISO/IEC 9126:2001 related survey [Jung]. The study was conducted by means of a questionnaire, which referred to 18 of the 27 quality sub-characteristics. In the questionnaire, the Reliability and compliance sub-characteristics were omitted, because the pretest users had difficulties with these sub-characteristics.

The survey contains input from 75 users of product software from one company producing a query and reporting tool for business databases. From these 75 users, 48 were end users, 25 were developers, and two were "other" users.

After processing the results, 14 sub-characteristics were classified in 5 dimensions based on correlations between the sub-characteristics. The values next to the sub-characteristics are the correlation coefficients ranging from -1 (total negative correlation) to +1 (perfect positive correlation)

Dimension 1	Dimension 2	Dimension 3	Dimension 4	Dimension 5
Analyzability (0.616)	Understandability (0,769)	Time behaviour (0.805)	Suitability (0.818)	Security (0.856)
Changeability (0.653)	Learnability (0.827)	Resource utilization (0.766)	Accuracy (0.648)	
Stability (0.814)	Operability (0.848)		Interoperability (0.796)	
Adaptability (0.699)	Attractiveness (0.616)			

Table 2 Correlations Table [Jung]

The sub-characteristics with correlation coefficients between them and derived dimension of 0.6 or higher are grouped in same dimensions. Four sub-characteristics: testability, installability, replaceability and co-existence were not related to any dimension and so they are not presented in the table.

The categorization above is similar to the ISO/IEC 9126-1 quality model categorization. The first dimension groups maintainability sub-characteristics with adaptability (a sub-characteristic of portability). The second dimension contains sub-characteristics of usability. The third dimension groups sub-characteristics of efficiency, while the forth dimension contains functionality sub-characteristics. It should be noted that security is in a separate fifth dimension, not related to the other functionality sub-characteristics.

The authors executed a survey similar to our surveys described in chapter 5, but they received a higher response rate. Therefore, they were able extract a statistical data from the survey. Our survey did not have a high response rate, but we managed to go step further and create category-based quality models.

National ICT Australia

[Al-Kilidar] conducted experimental evaluation of ISO/IEC 9126 quality standard. The experiment aimed at assessing the applicability of ISO/IEC 9126 to product design and possibilities of the quality assessment of the intermediate design product.

[Al-Kilidar] made the following remarks for the ISO/IEC 9126 standard:

- Parts of the ISO/IEC 9126 standard are ambiguous, e.g. the definition for functional compliance.
- Some definitions are overlapping, which can lead to multiple counts when the metrics are constructed.
- Some measures contain imperfections because they require information that is not available for the designers.

[Al-Kilidar] therefore concludes that, due to ambiguities and omissions, the current version of ISO/IEC 9126 standard fails to achieve the desired objectives.

However, ISO 9126 was not proposed for design assessment. Therefore, it is a bit preposterous to accuse ISO measures of imperfections. On the one hand, our opinion is also that ISO/IEC 9126 standard is too general. On the other hand, unlike [Al-Kilidar], we think that ISO/IEC 9126 can be used as a basis for a domain-specific quality model. In chapter 6, we will prove how ISO/IEC 9126 standard can be applied as a base for creation of categoru-based quality models based on ISO/IEC 9126.

Universitat Politèchnica de Catalunya (UPC)

This institution published numerous papers on the ISO/IEC 9126 quality model. The most relevant work is [Botella], which presents a methodology for building a domain-based quality model. The same group of authors also published quality models for some COTS products such as ERP and Mail Servers [Burgues].

These papers provide examples of domain-based quality models. We also believe that quality models are domain related, and therefore we use their methodology to construct domain-based quality models.

Universidad Simón Bolívar and Universidad Ezequiel Zamora

[Ortega] presents the design of a quality model with a systematic approach to product software. Their model is mainly focused on product’s efficiency and effectiveness.

Product efficiency is determined by internal design and programming activities, while product effectiveness is determined by activities involving requirement identification, interface design and general network design.

The authors designed a quality model that aims at reflecting the most important attributes of quality. They evaluated the model using the following evaluation steps:

- Designing a survey, where they define that they will evaluate similar products by evaluators with similar background
- Formulating, validating and applying metrics
- Defining an algorithm to obtain the quality estimates
- Analyzing the results

This research is interesting since we can use model design and model evaluation techniques similar to [Ortega]. The difference is that we execute the survey as part of the model design and not in the model evaluation phase.

Related Research Summary

In this chapter, we provided an overview of the research related to ISO/IEC 9126 and software quality. Cited papers and institutions prove that software quality and quality models were interesting and challenging research topics for scientists worldwide.

Research related so design of domain based quality models of [Botella] and [Burgues], is the most related and the most relevant to our work. Therefore, we will use their methodology as a guideline in our design of category-based quality models in the next chapters.

5. Survey results

Introduction

The surveys were executed in order to gain an industry input about the importance of ISO/IEC 9126:2001 sub-characteristics and characteristics.

We created two questionnaires in order to execute two surveys. The first version of the questionnaire was longer and it contained questions about all ISO/IEC 9126:2001 sub-characteristics; details about this questionnaire are available in the appendix. The second shorter version of the questionnaire contained questions about high-level characteristics of ISO/IEC 9126:2001.

We have executed two actions to invite the companies to participate in our surveys:
-The first action was in November 2006, when we sent an invitation by mail to nine companies to participate in our survey. We selected companies producing product software for different application domains, in order to gain impression about the importance of ISO/IEC 9126-1:2001 characteristics in different domains. From nine invited we organized one interview and we received one filled questionnaire.
- The second action was in March 2007, during the VVSS 2007 (Symposium on Verification and Validation of Software Systems). On this event, we distributed about 300 shorter questionnaires and we asked the event participants if they are interested to be contacted for an interview or for a longer version of the questionnaire. On that occasion, we received twelve filled short questionnaires and only six of them responded that they would like to participate in the survey based on the longer questionnaire.

Long Questionnaire Results

We executed two interviews and we have received three filled questionnaires. Participating companies are developing or testing product software in completely different application domains. However, we found the following to be common for most of the questionnaires:

- Functionality was selected as the most important high-level characteristic by all of the companies. This is not surprising, since the functionality consists in determining whether the software meets the functional requirements.
- Portability was selected as the least important high-level characteristics from four of total five companies. This is related to the fact that companies develop product software that should run on one software platform (operating system).
- Suitability, accuracy and interoperability were selected as the most important low-level sub-characteristics. This corresponds to an earlier remark that functionality is the most important high-level characteristic, so functionality sub-characteristic should be the most important sub-characteristic too.
- Replaceability and compliance sub-characteristics (for portability and maintainability) and other portability sub-characteristics were selected as some of the least important. That is probably related to the fact that portability was the least important high-level characteristic and that companies do not have to meet the compliance criteria.

Short Questionnaire Results

We received twelve filled short questionnaires from total of 200-250 questionnaires that were distributed during VVSS 2007, symposium organized by LaQuSo. From the twelve questionnaires, two contained invalid data so they were not included in the results.

In the short questionnaire, the participants were asked to rank the high-level characteristics by giving a percentage. They were also requested to provide information about their area of expertise and the type of software that they are verifying or developing.
The table bellow presents the results of this survey.

The first six columns show the percentage that the survey participants assigned to each high-level characteristic:
-F% presents the percentage assigned to functionality.
-R% presents the percentage assigned to reliability.
-U% presents the percentage assigned to usability.
-E% presents the percentage assigned to efficiency.
-M% presents the percentage assigned to maintainability.
-P% presents the percentage assigned to portability.

The next five columns show the types of product software that the survey participants were assessing, where:
-SIS means System Infrastructure Software.
-STD means Software Development Tools.
-AS means Application Software.
-SBS means Software Based Services.
-Other means other types of software not covered by the above four categories. Examples of these products were Embedded Software and Doc. Conv., meaning Document Conversion Application.

The last column shows the area of expertise of the participants, where:

- QA means testing and Quality Assurance.
- SE means Software Engineering.

F %	R %	U %	E %	M %	P %	Software Category	Expertise
40	9	10	10	20	1	AS, SBS	QA
40	15	15	20	8	2	AS, SBS	QA
70	10	5	10	5	0	AS, SBS	QA
30	20	15	10	25	0	AS	SE
30	15	20	10	10	15	SIS,AS, SBS	QA
30	25	20	10	15	0	SBS	QA
30	20	10	20	10	10	AS, SBS	QA
60	15	13	10	0	2	SBS	QA
50	10	10	20	5	5	SBS	QA
40	12	12	12	12	10	SIS,SDT, AS, SBS	SE, QA

Table 3 Results of the short questionnaire survey

The second table list the summations (with suffix Sm) and average (with suffix Av) percentage for the categories of Application Software (AS) and Software Based (SBS) Services, where:

- AS Av is the average percentage for Application software
- SBS Av is the average percentage for Software Based Services

	F%	R%	U%	E%	M%	P%
AS Sum:	280	101	87	92	90	38
AS Avg.	**40**	**14,4286**	**12,43**	**13,14**	**12,86**	**5,43**
SBS Sum	390	131	115	122	85	45
SBS Sum	**43,33**	**14,56**	**12,78**	**13,56**	**9,44**	**5**

Table 4 Summation and average values of the short questionnaire survey

On the basis of the above table, we can notice that:
-Functionality was selected as the most important high-level characteristics in this survey.
-Portability was selected as the least important high-level characteristics in this survey.
- Other four high-level characteristics (Reliability, Usability, Efficiency and Maintainability) have approximately same importance for the participants in our survey. The exception is that maintainability seems to be less important for software-based services.

Survey Results Conclusions and Recommendations

The response for our survey was not sufficient to provide statistically significant information. It seems that the ISO/IEC 9126 is not very interesting for the industry. However, based on our two surveys we can draw the following conclusions:

- All of the participants selected functionality as the most important high-level quality characteristic.
- Nine of ten participants selected portability as the least important high-level quality characteristic.
- The other high-level quality characteristics: reliability, usability, efficiency and maintainability were selected as equally important, which mainly depends of the business objectives of the software producer or verifier.

Based on these conclusions, we can make the following recommendations for future work:

- We will not assess the compliance sub-characteristics, because they do not seem relevant to us. This fact was also confirmed by most of the product software companies' questionnaires.
- We will not assess the portability in details, because it seems less relevant for the product software companies. However, some sub-characteristics as installability are important for some of the product software categories, therefore installability will be evaluated in the categories where it is relevant.
- We will further analyze which sub-characteristics are important for specific product software based on our domain analysis and of our longer questionnaire survey.

6. Analysis of product software

Introduction

In the previous chapter, we analyzed the results of our survey. Since we did not receive enough survey responses, we decided to determine ourselves which quality characteristics are important for product quality on specific examples from different categories defined by [OECD]. Accordingly, in this chapter, we continue with our quality evaluation by analyzing several examples of product software from different categories and create category-based quality models. Created category-based quality models contain the relevant ISO/IEC 9126-1 characteristics and sub-characteristics, attributes related to the sub-characteristics and metrics for the attributes.

The idea is that we assess product software from different categories, with our survey results and the product software documentation being used as an input. We will use the customer or market view to evaluate the selected product software applications.

The results from our survey revealed that functionality is the most important characteristic and that suitability is the most important sub-characteristic. These results should not be a surprise since functionality represents conformance to functional requirements that should be important for any product.

One important question, however, is which of the other characteristics representing non-functional requirements are important for various product software categories. In this chapter, we specify which characteristics are important for the three product software categories identified by [OECD]: infrastructure software, software development tools, and application software. We analyze them from a user perspective, trying to specify which characteristics are important for the end user. We also measure external quality (i.e., software behavior when executing) using external quality metrics.

In creation of category-based quality models, we used a methodology similar to the methodology described in [Botella] and [Burgues]. The methodology is divided in the following steps:

- Determining the importance of high-level quality characteristics. This step defines which ISO/IEC 9126:1 high-level quality characteristics are important
- Determining the importance of quality sub-characteristics. This step defines which ISO/IEC 9126:1 quality sub-characteristics are important for the product software. We estimate which sub-characteristics are important, but we also check the product documentation and survey results in order to verify our statements.
- Decomposing sub-characteristics into attributes. In this decomposing step, the abstract sub-characteristics are decomposed in more concrete entities - attributes.
- Determining metrics for the attributes. In this step, metrics for selected attributes are selected.

In our analysis, we will not consider the compliance sub-characteristics because our survey gave an indication that they are irrelevant. In addition, maintainability and its sub-characteristics will also not be considered, since we are executing external (black box) analysis, while the maintainability sub-characteristics such as changeability and testability are code (white box) related.

Some of the sub-characteristics can be directly decomposed to metrics. This is due to the fact that these sub-characteristics are not abstract in nature, so ISO/IEC 9126-2:2003 or other metrics can be used directly to measure the quality of the sub-characteristic, i.e., reliability sub-characteristic, maturity can be directly assessed and it is usually assessed in the industry with the metric *Mean time between failures (MTBF).*

In this chapter, we present a brief summary of the Infrastructure Software, Application software and Software Development Tool category. In Appendix C we provide further details about created category-based quality models presenting the attributes, metrics and results of the evaluation for the categories of software development tools and application software.

System Infrastructure Software

We evaluated two product software applications from this category:

1) Sun Solaris Operating System version 10
2) HP OpenView Operations for UNIX network management software version 8

We have chosen these two products, because they are typical representatives of System Infrastructure Software category. Sun Solaris is commonly used operating system and OpenView is one of the most popular network/infrastructure applications.

For this software category, we considered the following characteristics and sub-characteristics as relevant:

Functionality – has high importance for Operating Systems and Infrastructure Software category, because the product software should provide functions that are able to meet stated and implied needs. We found the following functionality sub-characteristics relevant for this product category:

- Suitability
- Interoperability
- Security

Reliability – has high importance for Operating Systems and Infrastructure Software category, because we expect these products should correctly operate under specified and extreme conditions such as software and hardware failures. We found the following reliability sub-characteristics relevant for this category:

- Maturity
- Fault tolerance
- Recoverability

Efficiency - has high importance for Operating Systems and Infrastructure Software category, especially for Operating Systems, because we expect that Operating

Systems should keep resources available for higher-level applications. We found the following efficiency sub-characteristics relevant for this category:

-Time behaviour

- Resource utilization

We found the following characteristic and accompanying sub-characteristics to be less relevant:

Usability – has medium importance for related products addressing home users market (i.e. Microsoft Windows), thus involving users with moderate computer skills.

Portability - is usually irrelevant for the operating systems, but partly relevant for infrastructure software. It is common for the System Infrastructure category products that producers prepared separate versions for every software and hardware platform.

Software Development Tools

We will monitor the following product software applications from this category:

1) TOAD tool for management and development of databases representing Database development tools
2) SA4J code analyzer for Java programming language product of IBM and Alpha Works representing code analyzer tools

We have chosen above products because they are typical representatives of this group. TOAD is popular application for Database development used in the industry; SA4J is example of code analyzer tool, product that is closer to academic environment. Both of these product have freeware versions that makes them easy accessible.

For this software category, we assume the following (sub-) characteristics as relevant:

Functionality:

- Suitability: these products should provide their specific functions like compiling or database development. Suitability of SA4J is defined that SA4J application should analyze structure of Java classes. Suitability of TOAD is defined with compilation, debugging and execution of stored procedures, triggers, functions and types.
- Accuracy is important for these products. The meaning of accuracy for these applications is to provide code fault detection. Accuracy of SA4J means that it should discover architectural problems in the analyzed application and detect *antipatterns* (bad design elements based on the set of known bad design examples). Accuracy for TOAD can be presented with SQL Optimization feature, which optimizes the database performance.

Reliability:

- Recoverability has high importance because we expect these product software applications to be able to recover the data or the actual work in the case of failure.

Usability:

- Understandability has medium importance because the users should be able to understand how to use these product software applications for their development tasks.
- Learnability has medium importance because users of these applications are expert, so we can understand learnability as enabling expert users to learn these applications quickly.

Efficiency:

- Time behaviour has medium importance because we expect that these product software applications should have low processing times. Otherwise, their usage will cost more, when we calculate the time when the developer is waiting for the results.

We found the following characteristic less relevant:

Portability is usually not very important characteristic for this product category. Development tools usually have different versions for different platforms. Installability is a bit relevant, but these applications should not necessarily install easily since their users have computer literacy.

Application Software

This product category is broader and contains many different products and product domains; therefore, we divide this category in the following subcategories:

- Administrative or office applications like text editors and email client application.
- Entertainment application like games, focusing on action games

We monitor the following product software applications from this category:

1) Microsoft Office Word part of the Microsoft Office suite
2) Minesweeper, game delivered as part of Microsoft Windows Operating System. Entertainment (games) subgroup will be analyzed in the next subsection, because we assume that quality characteristics of entertainment application are different with the previous categories.

We have chosen these two products, because they are good representatives of the category. Microsoft Word is the most popular text editor and probably one of the most used example of Application Software category worldwide. Minesweeper is a game that is installed with every Windows operating system, thus also with significant number of installations.

For office applications, we assume the following characteristics and sub-characteristics as relevant:

Application Software Office Applications

Functionality – has high importance for Office applications, because the applications should execute their functions like text editing. We found the following functionality sub-characteristics relevant for this product category:

- Suitability
- Accuracy
- Security

Reliability – has medium importance for Office applications, because we expect that data can be retrieved in case of failure. We found the following reliability sub-characteristic relevant for this product category:

- Recoverability

Usability - has high importance for Office applications, because we expect that these applications should be easy to understand, to use and to learn. We found the following usability sub-characteristics relevant for this product category:

- Understandability
- Learnability
- Operability

Efficiency - has medium importance for Office applications, because we expect that these applications are not very slow and they do not fully utilize the system resources. We found the following efficiency sub-characteristics partly relevant for this product category

- Time behavior
- Resource utilization

Portability: - has medium importance for Office applications, because we expect that these applications can be easily installed. We found the following portability sub-characteristic relevant for this product category:

- Installability

Corrections:
Security
Initially we did not select security as important sub-characteristic of application software. Analyzing the new features of Word 2007, we discovered that the producer introduced several security features such as digital signature, detection of documents containing macros and prevention of changes to the final version of the documents.
These facts prove the relevance of security for this product category.

Application Software Entertainment Applications

Functionality - has high importance for Entertainment application, because these applications should execute their basic functions that are running the animated gaming application, and these days they commonly run in a networked environment. We found the following functionality sub-characteristics relevant for this product category

- Suitability
- Interoperability

Usability - has medium importance for Entertainment applications, because we expect that these applications should be easy to understand, easy to use, easy to learn and attractive for the user. We found the following usability sub-characteristics relevant for this product category

- Understandability is important because the users should be able to understand and use these applications without much effort
- Learnability is also important because the users should be able to learn how to use these applications
- Attractiveness is probably the most important usability sub-characteristic, because if the product is not attractive the users might use similar but more attractive product.

Efficiency – has medium importance for Entertainment applications, because they should not be too slow. We found the following efficiency sub-characteristic partly relevant for this product category

- Time behaviour

Portability - has medium importance for Entertainment applications, because we expect that these applications can be easily installed and run with other applications. We found the following portability sub-characteristics relevant for this product category

- Installability
- Co-existence

Summary of analysis of product software

In the following table, we present a summary of sub-characteristics' importance for different product software categories. We use three levels of importance:

- "--" means low importance
- "+-" means medium importance
- "++" means high importance

ISO/IEC 9126:2001 sub-characteristics	System Infrastructure Software	Software Development Tools	Application Software Office and Antivirus applications	Application Software Entertainment applications
Functionality	++	++	++	++
Suitability	++	++	++	++
Accuracy	--	+-	+-	+-
Interoperability	++	+-	+-	+-
Security	++	+-	+-	--
Reliability	++	+-	+-	--
Maturity	++	+-	--	--
Fault tolerance	++	+-	--	--
Recoverability	+-	++	++	+-
Usability	--	+-	++	++
Understandability	--	+-	++	++
Learnability	+-	+-	++	++

Operability	--	+-	++	+-
Attractiveness	--	+-	+-	++
Efficiency	++	+-	+-	+-
Time behaviour	++	+-	+-	+-
Resource utilization	++	--	+-	--
Portability	--	+-	+-	+-
Adaptability	--	--	--	--
Installability	--	+-	++	++
Co-existence	--	+-	+-	+-
Replaceability	+-	+-	--	--

Table 5 Overview of sub-characteristics importance per product category

Based on the above table, we can define the profiles for every application category graphically as described in [Maiocchi], we will intentionally omit functionality because it has high importance for all product categories:

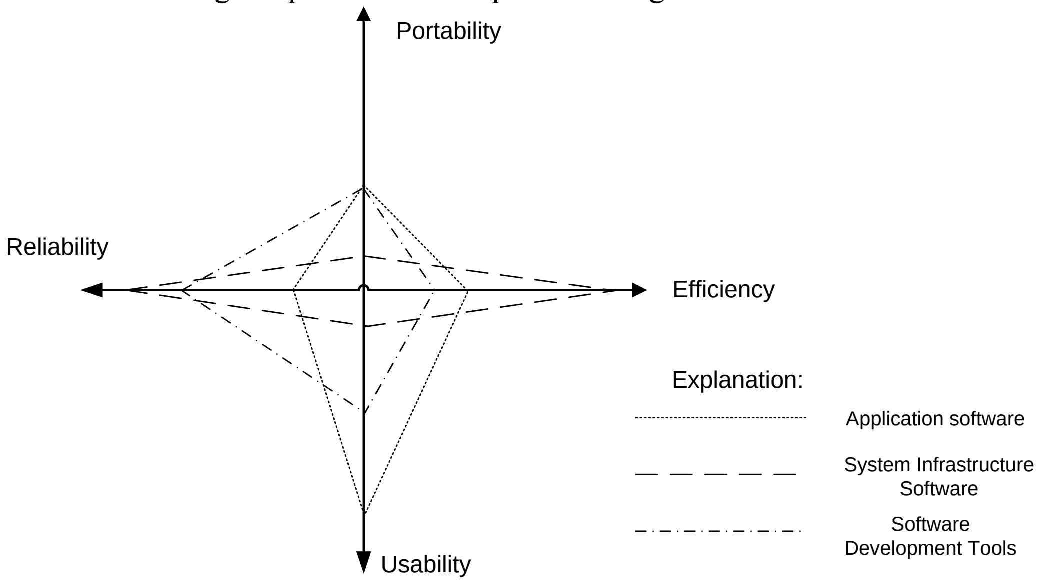


Figure 11 Quality chart for software categories

From the table and from the chart it is visible that different ISO/IEC 9126 characteristics are relevant per category. For System Infrastructure Software category next to the functionality, reliability and efficiency have high importance. For Software Development tools next to the functionality reliability, efficiency and usability have medium importance. For Application Software next to functionality, usability has high importance.

This analysis shows that for different software categories, we should focus our evaluation on different ISO/IEC 9126 characteristics. We followed this guideline in our evaluation and creation of category-based quality models. In appendix C we presented the details of our evaluation of sample products from different categories and the details of the category-based quality models. Presented category-based quality models can be reused for other products that can be categorized in one of the three categories.

In this chapter, we provided a summary of our product evaluation procedure and our results about the importance of different characteristics for different product categories. In the next chapter, we will present the reflection and concluding remark about this project.

7. Reflection

Reflection about the thesis project

The project presented in this thesis has been carried out as a part of LaQuSo certification activities. The objective of this project was to determine the quality attributes of product software artifacts.

The initial project description was clear and concise; our impression was that a project based on such description is interesting and challenging. The project description was also assuming participation of the software industry in the Netherlands and neighboring countries. Industry representatives were expected to participate in the interviews conducted by the author, discuss the quality attributes of their products and provide them for evaluation. Unfortunately, we did not gain enough support from the industry and that fact changed partly the project direction.

We believe that several factors might have been responsible for insufficient industrial participation.

First, the software industry hardly uses the ISO/IEC 9126 standard as noticed by [Pfleeger]. Our opinion is that the main reason for low industrial popularity of ISO/IEC 9126 is due to the growing market demand of certification, on the one hand, and inability of ISO/IEC 9126 to provide for any form of certificate, on the other. Producing software in different domains demands for compliance to domain-dependent standards (e.g. HIPAA or FDA compliance for medical product software, SOX compliance for EAI software). Therefore, the producers are focused on compliance with respect to the standards demanded by the market and do not pay much attention to standards that are not required. Consequently, evaluating the product software towards market demanding standards seems like a better business opportunity.

For software producers ISO/IEC 9126 is probably just another standard; they are not supposed to follow it, and they are not very enthusiastic about the standard. The producers have some internal or market objectives that are also part of the ISO/IEC 9126 standard; examples of these objectives are reliability metrics of [ISO/IEC 9126-2] *Mean Time Between Failures (MTBF)* i.e. of 200 hours or *availability* of i.e. 99,5%. Thus, in the case when producers want to have their product evaluated on some of the quality characteristics, ISO/IEC 9126 standard can be a good starting point. Evaluating quality characteristics important for the software producers, such as efficiency, reliability and usability, seems like a good business opportunity. Our expectation is that not every producer considers all three characteristics important. Therefore, potential projects could be evaluation of usability for producers of application software applications, and/or evaluation of reliability for infrastructure software producers.

Second factor contributing to low industrial participation is the time issue; software producers could not find time for participating in our survey. Our assumption is that it was difficult to claim some time for activities that are not essential for their business. With approach that is more persistent, we could organize more interviews with software producers, but even then, we would not get their software for evaluation. Another approach could be using the personal network of contacts within the Dutch Software Industry. We organized few interviews by contacting colleagues or friends employed in the software industry, but with this approach, we cannot get an input from various domains.

Third factor is that ISO/IEC 9126 does not provide procedure and methodology for evaluation. The ISO/IEC 9126 standard contains a quality model, collection of different metrics, but it does not contain a methodology and process for evaluation, process and methodology are described in the ISO/IEC 14598 standard. This issue makes the implementation of the ISO/IEC 9126 standard complex and vague for the industry.

Fourth factor contributing to low industrial participation is the price of ISO/IEC 9126 and related standards. Complete set of ISO/IEC 9126 and ISO/IEC 14598 standards costs about thousand US dollars, which is significant investment for private users and small companies. We expect that if the price is lower or the standards are free the popularity of the standards will be higher. In that case, private users could order or download the standards and use the parts that are interesting for them.

Despite of the above remarks, we nevertheless managed to prove that product software quality is domain dependent. We developed domain/category based quality models, and we demonstrated that such created models can be reused for evaluating various examples of product software applications with reasonable evaluation effort.

Evaluation process

ISO/EIC 9126 standard does not contain a methodology or a process for evaluation as part of the standard. Therefore, we followed the methodology published by [Botella] and [Burgues] as a guideline. The methodology is briefly described in Chapter 6. The difference with our approach is that we assumed that we could reduce the number of relevant quality characteristics and sub-characteristics per product. The first two steps, defining relevant quality characteristics and sub-characteristics were executed based on our category or domain investigation and our domain perception. The following two steps, deriving attributes for the sub-characteristics and defining metrics for the attributes, were more demanding.

We experienced difficulties in decomposing several sub-characteristics into attributes. These difficulties were most likely because the ISO/IEC 9126 standard does not contain the attributes layer. As a result, some sub-characteristics, such as “resource utilization” and “installability”, were difficult to be decomposed to attributes. We resolved these issues by checking the proposed metrics in [ISO/IEC 9126-2] and then proposing attributes based on the metrics.

Another issue was defining objective metrics. Examples were related to performance and usability metrics. In the case of performance metrics, like “response time”, the results are dependable on the hardware resources. Thus, we had to use our working hardware configuration as a reference configuration. Similar was the issue with usability metrics, where metrics defined in [ISO/IEC 9126-2] required a test user, where in absence of a test user an evaluator with higher IT literacy should act as a test user.

After executing evaluation of the categories of infrastructure software, software development tools and application software, we actually created three category-based quality models that can be reused for any product belonging to these three categories. Using the category based quality models we also designed an evaluation process that can be summarized with the following steps:

- Categorizing the product software, where the product should be assigned to one of the three categories, based on the product software characteristics and usage.
- Specifying the relevant metrics that will be measured, based on the category quality model. Category quality models are described in Chapter 6 and Appendix C. Several metrics related to functionality should be modified in order to allow functionality of evaluated product software and related product software from same domain.
- Executing measurement of the product software. Using the metrics defined in the previous phase, the evaluator should execute the test cases defined by the metrics.

One open issue is how we can derive an overall grade of product software quality. At some moment of our evaluation, we were calculating the average value of metric results per sub-characteristics, but this does not seem to be an appropriate method of calculation, because we came to the point where we were calculating average values of unrelated metrics. Consequently, the average value was not presenting an adequate picture of the quality per sub-characteristics. One solution for this issue could be defining norms for each tested metrics in different categories; with this approach, we can have pass/fail criteria per metric. Another solution for grading the product software could be giving a weight per metric and then deriving the final grade.

Another open issue is that ISO/IEC 9126 does not contain evaluation guidelines explaining the evaluation process, evaluation process and methodologies are described in another standard ISO/IEC 14598. Therefore, the evaluator using ISO/IEC 9126 only, should assess the software based on scientific papers, his experience and knowledge, without having clear guidelines examples and recommendations. The issue was tackled by the new series of ISO/IEC software product quality standards – SQuaRE. The SQuaRE series provides Quality evaluation guide and processes not only for evaluators but also for acquirers and developers in one standard group.

Reflection about [ISO/IEC 9126-2] metrics

In this chapter, we describe our experiences and provide our opinion about [ISO/IEC 9126-2] standard metrics. We focused on the metrics for external quality described in the second part of the ISO/IEC 9126 standard referred as [ISO/IEC 9126-2]. We evaluated four product software applications using these metrics. In the following paragraphs, we present our findings about possibilities to use [ISO/IEC 9126-2] as an external evaluator.

Numerous metrics

[ISO/IEC 9126-2] standard contains numerous metrics. For several sub-characteristics, [ISO/IEC 9126-2] also proposes numerous characteristics. For example, for Operability metrics the standard proposes eight metrics. In total, the standard contains more than a hundred external quality metrics. Assessing a product with all the proposed metrics can take months of effort per product. Authors of the standard were aware of this fact; therefore, they proposed evaluation based on the business objectives and the nature of the product [ISO/IEC 9126-1]. Our approach was similar to their proposal, so we tried to conduct a survey with the software producers from different domains in order to define which quality sub-characteristics are relevant for various products. The survey did not provide required responses from the industry, so we analyzed products from various domains and defined which characteristics are relevant. Our idea was to evaluate only a set of sub-characteristics that are relevant for the specific product software. With this method, we reduced the number of relevant metrics and consequently the evaluation effort per product down to one week.

General metrics

Some of the metrics proposed in [ISO/IEC 9126-2] are too general. This is logical, because the standard was designed and written to be applicable for any software product in any domain. Evaluators should refine the metrics according to the product they are evaluating. For example, [ISO/IEC 9126-2] proposes two security metrics: a metric *Access Auditability*, which for different products has a different meaning (although what the metric means is clear); and A*ccess Controllability,* which can be represented with a restricted permissions feature when evaluating an office product software such as MS Word 2003.

Similar general metrics are efficiency-related metrics such as *throughput and response time* metrics. Throughput is defined by [ISO/IEC 9126-2] as number of tasks completed over a time period. Evaluators should define which task is product specific and measure how many of these tasks are completed within a time period. Example of this can be found in the literature where UPC presents a metric *message throughput* for mail servers; our opinion is that UPC only refines *Throughput* metric of [ISO/IEC 9126-2] for mail servers.

Inapplicable metrics

Part of proposed [ISO/IEC 9126-2] metrics cannot be applied, because of the evaluating requirements and the application methods proposed by the standard. Examples of these metrics are Usability related metrics, where [ISO/IEC 9126-2] recommends conducting user tests. User test according to [ISO/IEC 9126-2] means monitoring and interviewing sample users of the product software. The standard recommends that for reliable results at least eight users should be tested. In absence of users, the evaluator can take that role; however, the issue here is that the evaluator has usually better IT skills than the typical application user. The relevance of the results is also questionable in this case because only one user is participating. We evaluated several understandability metrics, in a way that we were executing functions as described in the product documentation and demonstration. Example of these metrics is *completeness of description* and *demonstration accessibility*, where [ISO/IEC 9126-

2] proposes a method of user test. Our evaluation was that the evaluator was checking the product documentation and looking for demonstrations.

Another example of metrics inapplicable for external evaluation are suitability metrics where [ISO/IEC 9126-2] assumes that the evaluator posses requirements specification document of the product. Requirements specification documents are usually not publicly available for commercial product software, so external evaluator probably will not be able to evaluate suitability sub-characteristic as described in [ISO/IEC 9126-2]. With suitability metrics, we tried to redefine them in the usable manner so instead of evaluating all the functions from requirements specification we evaluated the main commercial features.

Similar issue is the one with the maturity sub-characteristic, where most of the metrics are related to detected and resolved problems. This fact makes the evaluation of external evaluators almost impossible, unless the producers are willing to share this information.

Another remark related to [ISO/IEC 9126-2] metrics is that these metrics provide results in numbers, where the number is usually between 0 and 1. We used the same approach but observed that, in some cases, numbers were impractical. This was especially the case with suitability metrics as “Functional adequacy”.

Applicable Metrics

Recoverability metrics are widely applicable, but their implementation requires monitoring the product software for a longer period. Examples of this kind of metrics are A*vailability, Mean Down Time, Restartability, and Mean Recovery Time*. We did not evaluate these metrics since we did not have the product software running for a longer period. Our opinion is that these metrics can be evaluated on base of the application and operating system log files on a system that is in use. In that case the evaluator can get information about restarts and crashes of the system caused by the product software.

Efficiency metrics for resource utilization can also be applied in many different domains. Some of them contain imperfections that can make the evaluation complex, but if the evaluators obtain the point of the metric, they can redefine it in a more useful way. Example of this metric is *maximal memory utilization* that we redefine it to *memory usage*, which is easier to measure.

Another group of usable metrics is installability metrics like *rase of installation* and *ease of Setup Retry* that are general and applicable for various product software applications.

Conclusion about [ISO/IEC 9126-2] Metrics

Our conclusion is that [ISO/IEC 9126-2] metrics can be used for quality evaluation. First part of the metrics (named as “Applicable Metrics”) can be used as they are provided by [ISO/IEC 9126-2], second part of metrics (named as “General Metrics”) can be used as a guideline on defining domain specific metrics based on the [ISO/IEC 9126-2] metrics and third part of metrics can be used only during internal evaluation.

Our opinion is that [ISO/IEC 9126-2] standard is not “ready to use” so the evaluators should adjust it to their application domain and business objectives.

Defined and derived metrics

Domain specific metrics

During this project, we defined a number of metrics that are not identical to the metrics described in [ISO/IEC 9126-2]. First part of the metrics were domain-specific metrics that are not described by [ISO/IEC 9126-2], but they provide indication about the product software functionality. Defining a domain-specific metric requires not only investigating product and marketing documentation of the evaluated product, but also investigating documentation from related product software application from the same domain. By reading the product documentation, the evaluators can gain a better overview about domain related features and quality characteristics. Based on the overview, we defined metrics like *Support of specific/additional text editor features* for text editors, “Storing/displaying the best scores” for gaming applications, *Support for different programming languages* and *Additional code analyzer features* for code analyzer tools.

Solutions for general metrics

Second part of the metrics ware proposed for [ISO/IEC 9126-2] metrics that were too general. Defining these metrics was based on the product documentation, but we had the [ISO/IEC 9126-2] metrics as a guideline. Example of a metric defined with this approach is *Grammar error correction*, specific text editor metric derived from *Error correction* in [ISO/IEC 9126-2].

Another interesting example redefining metric is *Data preservation in case of abnormal events* derived from Restorability [ISO/IEC 9126-2:2001] for evaluating DB development tool. We analyzed what is important to be preserved for DB development tool in case of abnormal events and we came to conclusion that data should be preserved.

Similar is the example with the security metrics of [ISO/IEC 9126-2], where we defined several metrics for text editor application. The first metric is *Document corruption prevention* derived from *Data corruption prevention* [ISO/IEC 9126-2]. Further we also defined additional metrics as *Open and Modify document protection* and *Macros protection*, that are domain specific, but initially derived from *Access controllability* and *Data corruption prevention* respectively.

We also derived portability metric *Supported Operating Systems* for DB development tool was more specific than *Supported Software Environment* metric provided by [ISO/IEC 9126-2].

Additional metrics

We could not find a metric about hardware requirements in [ISO/IEC 9126-2]. A minimal hardware requirement is important feature mentioned in marketing and technical documentation of product software, and gives quality information.

Therefore, we introduced a metric *Minimal Hardware Requirements* that should check minimal hardware requirements of a product software.

Similar example was with installability metrics, where [ISO/IEC 9126-2] contains *Ease of installation* metric, but it does not contain a metric about uninstallation of software. We consider uninstallation also important feature, so we derived a metric *Ease of uninstallation* metric.

Answers of the Research Questions

At the start of the project, we posed four research questions. At the end of this project, we can provide the following answers:

1) How do we measure product software quality?

We tried to design, develop and execute product software quality evaluation process. Measuring product software quality based on our designed process has the following phases:

- Categorizing the product software, where the product should be assigned to one of the three categories, based on the product software characteristics and usage.
- Specifying the related metrics that will be measured, based on the category quality model. Category quality models are described in Chapter 6 and Appendix C.
- Executing the measurement of the product software. Using the metrics defined in the previous phase, the evaluator should execute the test cases defined by the metrics.

2) Is product software quality domain dependent?

Our analysis in Chapters 5 and 6, pointed that for product software in different categories, different quality characteristics and sub-characteristics are relevant. Thus, we proved that product software quality is category dependent, because different product software categories have different usages and different expectations from users. We assume that if we go to the domain level, we will get more domain specific sub-characteristics and metrics. Thus, we can also prove that product software quality is domain dependent.

3) How can we use the ISO/IEC 9126-1 quality model in different domains?

Our recommendation is to use ISO/IEC 9126 category based quality models in different applications domains. Application domains represent subset of product categories, so our assumption is that category-based quality domains can be used for quality evaluation in different domains. During this project, we created category based quality models; we believe that these models can be reused for product software evaluation in the future.

4) What are the differences between product software quality and tailor-made software quality?

We have not evaluated tailor-made software during this project, but our experience and expectation is that tailor-made software is usually comparable with the product software from the same domain. Our expectation is that tailor-made software should have similar quality requirements as related product software in the same domain, which means that same quality characteristics and sub-characteristics should be relevant. One issue related to the tailor-made software is that there is a single customer, so the probability of detecting all faults in the software is lower, which in turn has an impact on the maturity of the tailor-made software. Consequently, we expect that the tailor-made software should have a lower reliability than the product software. Another assumption is that in the case of the tailor-made software, the communication between a producer and a customer is better, and that the requirements are clearly described to the producer. Because of this, we might expect better suitability and functionality in the case of the tailor-made software. We also expect that portability is less relevant for tailor-made software; this is because tailor-made software is supposed to run on one software and hardware platform. Further, we assume that installability of tailor-made software is not so important, because the producer is usually installing the software.

Conclusion and recommendations

We can evaluate product software quality using ISO/IEC 9126 standard as a framework. Our evaluation on the selected product software applications showed that we could use quality characteristics and sub-characteristics of [ISO/IEC 9126-1] standard and external metrics of [ISO/IEC 9126-2] in product software quality evaluation. These two ISO/IEC 9126 standards are not “ready to use”. Therefore, domain analysis reducing number of relevant quality characteristics and sub-characteristics per product is one of the most important steps in quality evaluation, which should be executed at the start. Further, relation from characteristics to metrics should be established by decomposing sub-characteristics to attributes and proposing relevant metrics for the attributes. When metrics are defined, the evaluator should test the application and modify the metrics that are not applicable.

We conclude that external product software quality is domain or category dependent, because for different categories different quality characteristics and sub-characteristics are relevant. Therefore, we created reduced quality models based on ISO 9126 that can be applied per product software category. Using these reduced quality models, we decreased the evaluation effort per product software. Proof of this concept was evaluating the external quality of four product software applications from different domains.

ISO/IEC 9126 standard is not gaining the deserved attention from the industry. We assume that this is because of the fact that market does not demand ISO/IEC 9126 certification and ISO/IEC 9126 does not offer any certification at all. At the same time different markets require following of other standards and regulations. ISO/IEC 9126 is mainly used by the research institutions and still not extensively used in the industry. Another reason for ISO/IEC 9126 unpopularity is that the standard does not describe the evaluation process, so the companies miss the information how to use this standard. We expect that this process issue will be resolved with the next generation of ISO/IEC 9126 quality standards SQuaRE.

Our recommendation is that providing consulting and certification services for market-requested standards are better business opportunities than evaluating software quality using ISO/IEC 9126. An alternative business opportunity is using ISO/IEC 9126 standard as a guideline for evaluation of specific quality characteristics i.e. usability of application software, provided that there is a demand from the software producers.

Future work

Current work did not focus much on the research question about differences between product software and tailor-made software quality evaluation. We believe that quality for tailor-made software can also be evaluated using [ISO/IEC 9126-1], and ISO/IEC category-based quality models. Tailor-made software evaluation looks like a nice challenge for a future project at LaQuSo.

Quality evaluation of system infrastructure software was not completed in this project, because software producers in the Netherlands do not produce products from this category. Therefore, they do not seem relevant for future evaluation of other products.

In this chapter, we mentioned several issues related to evaluation of product software as grading the product software are establishing evaluation process. We believe that next generation of ISO/IEC 9126 quality standards SQuaRE will address these issues and integrated them in one group of standards. Therefore, researching application of SQuaRE quality standards should be one of the related research activities of software quality scientists in the future.

Software quality and quality are interesting scientific areas containing many interesting research topics. Future challenges in the future can be industrial standards related to software quality processes as Six sigma, CMM (Capability Maturity Model), TMap and ISEB (Information Systems Examinations Board). These standard seem interesting and are widely used by the industry, therefore researching them can provide not only scientific, but also practical benefits.

8. List of Abbreviations

AS	Application Software
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CMM	Capability Maturity Model
EAI	Enterprise Application Integration
ERP	Enterprise Resource Planning
FDA	Food and Drug Administration
FTP	File Transfer Protocol
HIPAA	Health Insurance Portability and Accountability Act, address the security and privacy of health data
ICT	Information and Communication Technology
ISEB	Information Systems Examinations Board
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
LaQuSo	Laboratory for Quality Software
MTBF	Mean Time Between Failures
OECD	Organisation for Economic Co-operation and Development
RU Nijmegen	Radboud Universiteit Nijmegen
SBS	Software Based Services
SDT	Software Development Tools
SIS	System Infrastructure Software
SOX	Sarbanes-Oxley Act
SQuaRE	Software Product Quality Requirements and Evaluation
TMap	Test Management Approach
TU/e	Technische Universiteit Eindhoven, English name is Eindhoven University of Technology
UPC	Universitat Politèchnica de Catalunya
VVSS	Verification and Validation of Software Systems
WWW	World Wide Web

9. References:

Al-Kilidar, Hiyam, Karl Cox, Barbara Kitchenham, *The Use and Usefulness of the ISO/IEC 9126 Quality Standard*, International Symposium on Empirical Software Engineering, pp. 126-132, November 2005,

Boehm, B. W, J. R. Brown, M. Lipow, *Quantitative Evaluation of Software Quality*, Proceding 2nd International Conference on Software Engineering, October 1976, pp. 592-605

Botella, Pere, Xavier Burgues, Juan P. Carvallo, Xavier Franch, Carme Quer, *Using Quality Models for Assessing COTS Selection*, In Proceeding of
5th Workshop on Requirements Engineering (WER) pp. 263-277, Valencia, Spain, November 2002.

Burgues, Xavier, Pere Botella, Juan P. Carvallo, Xavier Franch, Joan A. Pastors, Carme Quer , *Towards a Quality Model for the Selection of ERP Systems*, Cechich et al. (Eds.): Component-Based Software Quality, LNCS 2693, Springer-Verlag Berlin Heidelberg, pp. 225–245, 2003

Grady, R., Caswell, D.. *Software Metrics: Establishing a Company-Wide Program*. Englewood Cliffs, NJ, Prentice-Hall, 1987.

Crosby, Phillip B., *Quality Is Free: The Art of Making Quality Certain*,
McGraw-Hill Companies 1979, ISBN: 978-0070145122

Cote, Marc-Alexis, Witold Suryn and Clode Y. Laporte, *The Evolution Path for Industrial Software Quality Evaluation Methods Applying ISO/IEC 9126:2001 Quality Model: Example of MITRE's SQAE Method*, Software Quality Journal,13, pg. 17-30 Springer Science 2005

Dromey*, R. G, Software Product Quality: Theory, Model, and Practice*, Technical Report, Software Quality Institute, Griffith University, Nathan, Brisbane, Australia 1998

Fitzpatrick, Ronan, *Software Quality: Definitions and Strategic Issues,*_Staffordshire University, School of Computing Report, Staffordshire University, UK, 1996

Garvin D., *What does Product Quality Really Mean?* Sloan Management Review, Fall 1984

Gillies, *Alan C., Software quality: Theory and management*, Chapman & Hall, London, England 1992

Grady *R.,* D. Caswell, Software *Metrics: Establishing a Company-Wide Program*. Englewood Cliffs, NJ, Prentice-Hall 1987

Ince Darrel, *ISO 9001 and software quality assurance*, McGraw-Hill, Berkshire, ISBN: 0077078853, England 1994

ISO/IEC 9126-1, *ISO/IEC 9126-1 Software engineering Product quality Part 1: Quality Model*, ISO/IEC 2001

ISO/IEC 9126-2, *ISO/IEC, ISO/IEC 9126-2 Software engineering Product quality Part 2: External Metrics*, ISO/IEC 2003

Jung Ho-Won, Seung-Gweon Kim, Chang-Shin Chung, *Measuring Software Product Quality: A Survey of ISO/IEC 9126,* IEEE Computer Society 2004

Kececi N, L. Buglione, A. Abran, *An Integrated Graphical Assessment for Managing Software Product Quality*, Universite Québec Montréal, Canada

Kent, *Beck, Extreme Programming Explained: Embrace Change.* Addison Wesley, 2000.

Kitchenham Barbara, Shari Lawrence Pfleeger, *Software Quality: The Elusive Target*, IEEE Software, Volume 13, Issue 1 Pages: 12 - 21 (January 1996)

Lassila Aki, Jani-Pekka Jokinen, Janne Nylund, Petru Huurinainen, Markku Maula, Jyrki Kontio*, Finnish Software Product Business: Results of the National Software Industrial Survey 2006*, Technical report, Centre of Expertise for Software Product Business

Lee Edward A., *Embedded Software,* Advances in Computers
(M. Zelkowitz, editor), Vol. 56, Academic Press, London, 2002

Litkowski Kenneth, Erwin W. Bedarf, *Using Structured Interviewing Techniques*, United States General Accounting Office, Program Evaluation and Methodology Division June 1991

Maiocchi Marco, *Experiences in evaluating software quality within the banking sector*, Achieving Software Product Quality Erik van Veenendaal pg. 109-125/Julie McMullan (eds.)

McCall *J.A.,* J.P. Cavano, A *Framework for the Measurement of Software Quality*, Proceedings of the ACM Software Quality Assurance Workshop, November 1978, pp. 133-139

OECD, Organization for Economic Co-operation, and Development: *The software sector: Growth, structure and policy issues.* OECD Report STI/ICCP/IE(2000)8/REV2 (2001)

Ortega Maryoly, Maria Perez, Teresita Rojas*, Construction of a Systemic Quality Model for Evaluating a Software Product,* Kluwer Academic Publishers. Software Quality Journal, 11, 219–242, 2003

Pfleeger Shari Lawrence, *Software Engineering: Theory and Practice*, 2nd edition, Upper Saddle River, NJ, Prentice Hall, 2001

Rawashdeh Adnan, Bassem Matalkah, *A New Software Quality Model for Evaluating COTS Components*, Journal of Computer Science 2 (4): 373-381, 2006 ISSN 1549-3636

Rubey R. J., R. D. Hartwick, *Quantitative Measurement of Program Quality*, ACM National Conference Proceedings, 1968, pp. 671-677.

Sawyer Erran Carmel, Steve Sawyer**,** *Packaged software development teams: what makes them different?*, Information Technology & People, **11**, 7–19

Suryn Witold, Alain Abran, *ISO/IEC SQuaRE. The second generation of standards for software product quality*, IASTED 2003 - SEA 2003 November 3-5, 2003 Marina del Rey, CA, USA

Xu Lai, Sjaak Brinkkemper, *Concepts of Product Software: Paving the Road for Urgently Needed Research*, Technical report, Institute of Information and Computing Sciences, Utrecht University, The Netherlands

Appendix

A. Appendix ISO/IEC 9126-1 Questionnaire

Introduction

This survey is part of a master thesis project about "Product Software Quality" at Eindhoven University of Technology (TU/e). The project is executed at the Laboratory for Quality Software (LaQuSo). We conduct this research of software product quality characteristics in order to set a product quality model based on the ISO/IEC 9126 standard.

Part of the project is research of a number of Software product companies in the Netherlands (or wider) and to find which software product quality characteristics are important for them.

We selected you as a representative of a Software Product company who can give a significant input for our survey. We are grateful to your organization for participating in this survey. Results of this survey and thesis report will be available for your company.

Your participation in the survey and the name of your company will not be mentioned in the thesis report or in other scientific publications.

The results of this survey will be analyzed and incorporated in the Master thesis report.

Questions

These questions are related to the software product that is produced by your company, the questions are not related to the software development process nor your organization. Questions are based on the ISO/IEC 9126-1 Software Engineering – Product quality- Quality model. The ISO/IEC 9126-1 Quality model is presented on the following figure from [1]:

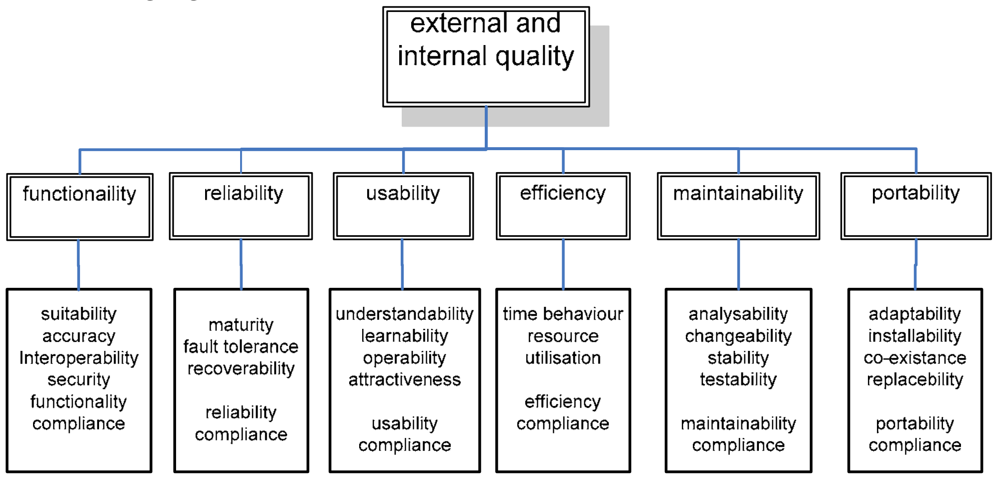


Figure 1 Quality model for external and internal quality [1]

As you notice on Figure 1, ISO/IEC 9126-1 contains 6 main quality characteristics (functionality, reliability, usability, efficiency, maintainability and portability) and 27 subcharacteristics (portability, accuracy …replaceability, portability compliance).

There are two questions for each subcharacteristic. The first question asks about the importance of the sub-characteristics. The second question asks how you assess the sub-characteristics.

Definitions from [1] are provided in *Italic style* to describe every characteristic as specified in the ISO/IEC 9126-1 standard.

Instruction

Please give a grade about the importance of the sub-characteristics scale where 1 = very unimportant, 2 = unimportant, 3 = neutral, 4 = important, 5 = very important

On the question about assessing the sub-characteristics, you should give a number from 1 to 4, where 1 = you do not asses the sub-characteristic, 2 = you assess this sub-characteristics manually, 3 = you asses the sub-characteristics using tools or automated methods and 4 = means that you asses the sub-characteristics manually and using tools or automated methods.

Functionality Questions

*Definition: **Functionality** is the capability of the software product to provide functions which meet stated and implied needs when the software is used under specified conditions.*

The sub-characteristics of **functionality** are:

- ***Suitability** is the capability of the software product to provide an appropriate set of function for specified tasks and user objectives.*

1) How important is the **suitability** of the software product produced by your company?

   1 2 3 4 5

2) How do you assess the **suitability** of the software product?

   1 2 3 4

- ***Accuracy** is the capability of the software product to provide the right or agreed results or effect with the needed degree of precision.*

3) How important is the **accuracy** of the software product produced by your company?

   1 2 3 4 5

4) How do you assess the **accuracy** of the software product?

1 2 3 4

- ***Interoperability** is the capability of the software product to interact with one or more specified systems.*

5) How important is the **interoperability** of the software product produced by your company?

   1 2 3 4 5

6) How do you assess the **interoperability** of the software product?

   1 2 3 4

- ***Security** is the capability of the software product to protect information and data so that unauthorized person.*

7) How important is the **security** of the software product produced by your company?

   1 2 3 4 5

8) How do you assess the **security** of the software product?

   1 2 3 4

- ***Functionality compliance** is the capability of the software product to adhere to standards, conventions or regulations in laws and similar prescriptions relating to functionality.*

9) How important is the **functionality compliance** of the software product produced by your company?

   1 2 3 4 5

10) How do you assess the **functionality compliance** of the software product?

   1 2 3 4

Reliability Questions

Definition: Reliability is the capability of the software product to maintain a specific level of performance when used under specified conditions.

The sub-characteristics of the **reliability** are:

- ***Maturity** is the capability of the software product to avoid failure as a result of faults in the software.*

11) How important is the **maturity** of the software product produced by your company?

1	2	3	4	5

12) How do you assess the **maturity** of the software product?

1	2	3	4

- ***Fault tolerance** is the capability of the software product to maintain a specific level of performance in cases of software faults or of infringement of its specified interface.*

13) How important is the **fault tolerance** of the software product produced by your company?

1	2	3	4	5

14) How do you assess the **fault tolerance** of the software product?

1	2	3	4

- ***Recoverability** is the capability of the software product to re-establish a specified level of performance and recover the data directly affected in the case of a failure.*

15) How important is the **recoverability** of the software product produced by your company?

1	2	3	4	5

16) How do you assess the **recoverability** of the software product?

1	2	3	4

*i. **Availability** is the capability of the software product to be in a state to perform a required function at a given point in time, under stated conditions of use. Externally, availability can be assessed by the proportion of total time during which the software is in up state. Availability is combination of maturity (that covers the frequency of failure), fault tolerance and recoverability* *(which covers the length of down time after each failure).*

17) How important is the **availability** of the software product produced by your company?

1	2	3	4	5

18) How do you assess the **availability** of the software product?

1	2	3	4

- ***Reliability compliance** of the software product to adhere to standards, conventions or regulations relating to reliability.*

19) How important is the **reliability compliance** of the software product produced by your company?

1	2	3	4	5

20) How do you assess the **reliability compliance** of the software product?

1	2	3	4

Usability Questions

Definition: Usability is the capability of the software product to be understood, learned, used and attractive to the user, when used under specified condition.

The sub-characteristics of the **usability** are:

- ***Understandability** is the capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use.*

21) How important is the **understandability** of the software product produced by your company?

1	2	3	4	5

22) How do you assess the **understandability** of the software product?

1	2	3	4

- ***Learnability** is the capability of the software product to enable the user to learn its application.*

23) How important is the **learnability** of the software product produced by your company?

1 2 3 4 5

24) How do you assess the **learnability** of the software product?

1 2 3 4

- ***Operability** is the capability of the software product to enable the user to operate and control it.*

25) How important is the **operability** of the software product produced by your company?

1 2 3 4 5

26) How do you assess the **operability** of the software product?

1 2 3 4

- ***Attractiveness** is the capability of the software product to be attractive to the user.*

27) How important is the a**ttractiveness** of the software product produced by your company?

1 2 3 4 5

28) How do you assess the a**ttractiveness** of the software product?

1 2 3 4

- ***Usability compliance** is the capability of the software product to adhere to standards, conventions, style guides or regulations regarding to usability.*

29) How important is the **usability compliance** of the software product produced by your company?

1 2 3 4 5

30) How do you assess the **usability compliance** of the software product?

1 2 3 4

Efficiency Questions

Definition: Efficiency is the capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions.

The sub-characteristics of the **efficiency** are:

- ***Time behaviour** is the capability of the software product to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions.*

  31) How important is the **time behaviour** of the software product produced by your company

  1 2 3 4 5

  32) How do you assess the **time behaviour** of the software product?

  1 2 3 4

- ***Resource utilization** is the capability of the software product to use appropriate amounts and types of resources when the software performs its function under stated conditions.*

  33) How important is the **resource utilization** of the software product produced by your company?

  1 2 3 4 5

  34) How do you assess the **resource utilization** of the software product?

  1 2 3 4

- ***Efficiency compliance** is the capability of the software product to adhere to standards or conventions regarding to **efficiency**.*

  35) How important is the **efficiency compliance** of the software product produced by your company?

  1 2 3 4 5

  36) How do you assess the **efficiency compliance** of the software product?

  1 2 3 4

Maintainability Questions

Definition: Maintainability is 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.

The sub-characteristics of the **maintainability** are:

- ***Analysability*** *is the capability of the software product to be diagnosed for deficiencies or causes of failures in the software, or for the parts to be modified to be identified.*

    37) How important is the **analysability** of the software product produced by your company?

    1 2 3 4 5

    38) How do you assess the **analysability** of the software product?

    1 2 3 4

- ***Changeability*** *is the capability of the software product to enable a specified modification to be implemented.*

    39) How important is the **changeability** of the software product produced by your company?

    1 2 3 4 5

    40) How do you assess the **changeability** of the software product?

    1 2 3 4

- ***Stability*** *is the capability of the software product to avoid unexpected effects from modifications of the software.*

    41) How important is the s**tability** of the software product produced by your company?

    1 2 3 4 5

    42) How do you assess the s**tability** of the software product?

    1 2 3 4

- ***Testability*** *is the capability of the software product to enable modified software to be validated.*

    43) How important is the **testability** of the software product produced by your company?

1 2 3 4 5

44) How do you assess the **testability** of the software product?

1 2 3 4

- ***Maintainability compliance*** *is the capability of the software product to adhere to standards or conventions related to maintainability.*
  45) How important is the **maintainability compliance** of the software product produced by your company

  1 2 3 4 5

  46) How do you assess the **maintainability compliance** of the software product?

  1 2 3 4

Portability Questions

Definition: ***Portability*** *is the capability of the software product to be transferred from one environment to another.*

The sub-characteristics of the ***portability*** *are:*

- ***Adaptability*** *is the capability of the software product to be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered.*
  47) How important is the **adaptability** of the software product produced by your company?

  1 2 3 4 5

  48) How do you assess the **adaptability** of the software product?

  1 2 3 4

- ***Installability*** *is the capability of the software product to be installed in a specified environment.*
  49) How important is the i**nstallability** of the software product produced by your company?

1 2 3 4 5

50) How do you assess the i**nstallability** of the software product?

1 2 3 4

- ***Co-existance** is the capability of the software product to co-exist with other independent software in a common environment sharing common resources.*

51) How important is the **co-existance** of the software product produced by your company?

1 2 3 4 5

52) How do you assess the **co-existance** of the software product?

1 2 3 4

- ***Replaceability** is the capability of the software product to be used in place of another specified software product for the same purpose in the same environment.*

53) How important is the **replaceability** of the software product produced by your company?

1 2 3 4 5

54) How do you assess the **replaceability** of the software product?

1 2 3 4

- ***Portability compliance** is the capability of the software product to adhere to standards or conventions related to portability.*

55) How important is the **portability compliance** of the software product produced by your company?

1 2 3 4 5

56) How do you assess the **portability compliance** of the software product?

1 2 3 4

ISO/IEC 9126-1 Ranking Questions

57) Which of the six high level characteristics (Functionality, Reliability, Usability, Efficiency, Maintainability and Portability) is the most important to you?
58) Which of the six high level characteristics is the least important?
59) If you cannot choose the most and the least important, please try to estimate their meaning dividing 100% in six pieces.

60) Which three low level (sub-characteristics) are the most important?
61) Why they are most important?
62) Which three sub-characteristics are the least important?
63) Which low-level (sub-characteristics) does your company assess/measure currently?

General questions about ISO/IEC 9126-1 and the questionnaire

64) Which of these 28 quality sub-characteristics are obsolete for assessing the quality of your product software?
65) Which additional quality characteristics are relevant for assessing your product software?
66) What do you think about this questionnaire?
67) Which questions would you remove from the questionnaire?

Reference:

[1] ISO/IEC 9126-1 Software engineering Product quality

B. Appendix Short Questionnaire Version

ISO/IEC 9126-1 Questionnaire

Introduction

This research is carried out as a part of a Master Thesis on "Product Software Quality". We research product software quality characteristics in order to design quality models based on the ISO/IEC 9126 standard. We expect that different application domains demand different quality models. Hence, we intend to construct a domain-based quality models.

Your participation in the survey and the name of your company will not be mentioned in the thesis report or in other scientific publications.

ISO/IEC 9126 – 1 Software Product Quality Model

These questions are related to the software product that is produced or evaluated by your company, the questions are not related to the software development process nor to your organization. Questions are based on the ISO/IEC 9126-1 Software Engineering – Product quality – Quality model. The ISO/IEC 9126-1 Quality model is presented on the following figure:

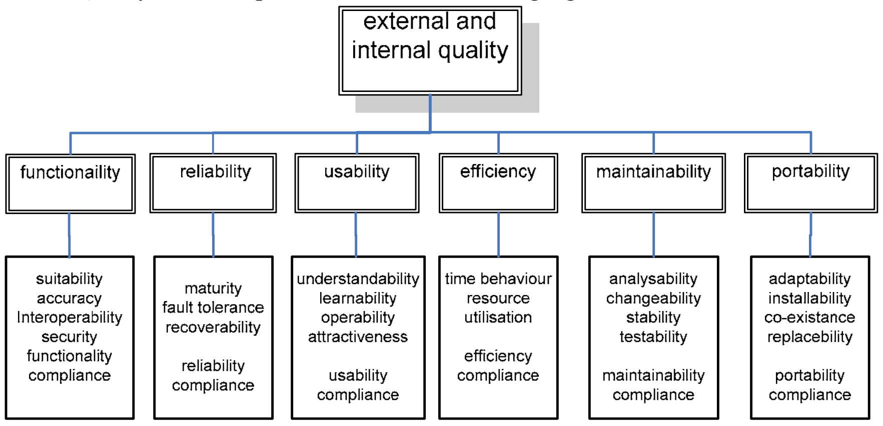


Figure Quality model for external and internal quality ISO/IEC 9126 - 1

As you notice on Figure 1, ISO/IEC 9126-1 contains 6 main quality characteristics (functionality, reliability, usability, efficiency, maintainability and portability)

The definitions of the main quality characteristics are presented in *Italic* font below:

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

Reliability *is the capability of the software product to maintain a specific level of performance when used under specified conditions.*

Usability *is the capability of the software product to be understood, learned, used and attractive to the user, when used under specified condition.*

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

Maintainability *is 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 *is the capability of the software product to be transferred from one environment to another.*

Questions:

1. Do you assess functionality of product software?
   Yes □ No □
2. Do you assess reliability of product software?
   Yes □ No □
3. Do you assess usability of product software?
   Yes □ No □
4. Do you assess efficiency of product software?
   Yes □ No □
5. Do you assess maintainability of product software?
   Yes □ No □
6. Do you assess portability of product software?
   Yes □ No □

7. Rank the six high level characteristics (Functionality, Reliability, Usability, Efficiency, Maintainability and Portability) starting with the most important characteristic first, please try to estimate their meaning dividing 100% in six pieces.

8. What kind of product software do you (your organization) produce/test?
   a) System infrastructure software
   b) Software development tools
   c) Application software
   d) Software based services
   e) Other, please specify ____

9. What is your area of expertise within the company/institution? Please choose the function that best describes you.
   a) Software Engineering
   b) Software Testing and Quality Assurance
   c) Project or Line Management
   d) Other, please specify________________________

10. Are you interested in a longer questionnaire or interview about product software quality?
    Yes □ No □

If yes, please leave your contact information or give your business card with this questionnaire:

Name:_______________________________ Phone:_______________________________

Email:_______________________________ Company:_____________________

Please leave the filled copies of this questionnaire at the reception.

C. Appendix Product Software Evaluation

Operating System Attributes and Metrics

Functionality Attributes and Metrics of Operating System (OS):

We can define the following functionality attributes related to operating systems functionality some of the attributes are related to other characteristics:

Suitability attributes of OS:

- Managing system resources, (these attributes are also related to performance)
  - Multitasking or switching processes quickly
- Managing system memory
  - Virtual memory management, increases the amount of memory available for each process by using the disk storage like main memory
  - Managing virtual addresses, preventing different processes from interfering with each other's memory
- Supporting scripting/programming languages

Suitability metrics of OS:

- Functional adequacy of major functions [ISO/IEC 9126-2]. Evaluator should verify if major functions are adequately functioning.
- Functional implementation completeness of major functions. Evaluator should verifying that major functions are completely functional
  - Management of system processes, evaluator should check the state of the system processes and try to stop/start system processes
  - Management of system memory and virtual memory, evaluator should check memory status with the commands vmstat and memstat
  - Supporting/including compiler and runtime environment for common programming languages i.e. C, Java, C++, evaluator should check which of the compilers are present in the OS.
  - Supporting/including scripting languages and shells (Born, bash, Korn shell, c shell, PERL), evaluator should check which of the shells and scripting languages are supported.

Interoperability attributes of OS:

- Supporting different hardware peripherals
- Supporting software Internet/networking services
- Interoperability with other Operating Systems (i.e. Linux and Windows)

Interoperability metrics of OS:

- **Data exchangeability with other systems and peripherals**

Evaluator should check the support of these hardware peripherals**.**

- Support of CDROM/DVD ROM
- Support of CD/DVD writing devices CDRW/DVD RW
- Support of on-board audio
- Support of floppy disk
- Support of USB devices
- Support of Wireless Network Interfaces

- Support of Ethernet Interfaces
- Support of Printer devices
- Supporting of common networking protocols (telnet, ssh, FTP, WWW, etc) as part of the Operating system

Security attributes of OS:

- Providing internal security
  - Protection the computer's resources from the programs concurrently running on the system
  - Providing auditing of security events
- Providing external Security
  - Providing secure access to the services for the authorized users
  - Providing Network Security

Security metrics of OS:

- Access controllability [ISO/IEC 9126-2]. Evaluator should check whether OS has mechanisms to allow/deny access. Then verify if illegal operations are possible, under illegal operations we mean unauthorized users access.
- Access auditability [ISO/IEC 9126-2]. Evaluator should check that the OS has logging of access. If access login exists, then is it recording all accesses?
- Providing System files integrity, comparable to data corruption prevention [ISO/IEC 9126-2]. Evaluator should check frequency of system files corruption events.
- Protecting system processes, Boolean metric, Evaluator should check whether the OS contains mechanisms for protecting system processes from application processes and non-system users.
- Providing software firewall Boolean metric, Evaluator should check whether the OS contains software firewall. Evaluator should check OS technical documentation or help whether firewall is part of the OS. If firewall is part of the OS then evaluator should be able to enable it.

Reliability attributes and Metrics of OS:
Maturity attributes of OS:

- Existence and quality of stability indicators

Maturity metrics of OS:

- Mean Time Between Failures [ISO/IEC 9126-2]

Fault tolerance attributes of OS:

- Error Avoidance
- Performance in case of failure

Fault tolerance metrics of OS:

- Breakdown avoidance [ISO/IEC 9126-2]. Evaluator should analyze system files and count the breakdowns with respect to failures.
- Incorrect operation avoidance [ISO/IEC 9126-2]. Evaluator should execute number of test cases that can cause critical failures and count how many of the failures were avoided.

- Support for clustering this is Boolean metric whether OS supports clustering or not. Evaluator should check OS technical documentation whether clustering is supported by the evaluated OS.

Recoverability attributes of OS:
- Recoverability from hardware faults
- Recoverability from software/application faults

Recoverability metrics of OS:
- Availability of OS [ISO/IEC 9126-2]. Evaluator should analyze log files of a production system, calculate operation time To, and repair time Tr.
- Mean Downtime of OS [ISO/IEC 9126-2]. Evaluator should analyze log files of a production system and calculate down time during a specified trial period.
- Mean recovery time [ISO/IEC 9126-2]. Evaluator should analyze log files of a production system and calculate recovery time for each time the system was brought down. After calculating the recovery times the evaluator should calculate average value.
- Restartability of OS [ISO/IEC 9126-2]. Evaluator should analyze log files of a production system and counts the number of successful restarts and total number of restarts.

Efficiency attributes and Metrics of OS
Resource utilization attributes of OS:
- CPU efficiency
- Memory efficiency
- Memory swapping
- Networking performance
- File system performance
- Application performance
- Hardware system requirements like CPU and amount of system memory

Resource utilization metrics of OS:
- Networking device utilization related to I/O devices utilization [ISO/IEC 9126-2]
- Networking loading limits related to I/O loading limits [ISO/IEC 9126-2]
- Memory usage. Evaluator should check the amount of memory used by the system processes. This can be done running system tools or commands i.e. Task manager on Windows OS or UNIX system commands (top or prstat) on a UNIX system.
- Maximum CPU Utilization caused by system processes, similar as maximum memory utilization [ISO/IEC 9126-2], but in this case evaluator should measure the maximum of CPU utilization that can have values from 0 to 100%.
- Amount of CPU time the operating system spends fulfilling system calls, similar as previous time, it this case evaluator should calculate time spent on system processes and divide with total uptime of the system

Time behaviour attributes of OS:
- Response and processing times

Time behaviour metrics of OS:

- Boot time. Evaluator should measure the time that a system needs to boot starting from moment when the power button is pressed to the moment when the user successfully logged in. This test case should be executed ten times and average value calculated.
- Response time for operations derived from [ISO/IEC 9126-2], i.e. time needed to open terminal application. Evaluator should execute this test case should be executed ten times and average value calculated.

Network Management Application Attributes and Metrics

Functionality Attributes of Network Management Application (NMA)

We can define the following functionality attributes related to OpenView Operations

Suitability attributes:

- Processing events from different systems on the network
- Presenting events from different systems on the network

Suitability metrics:

- Event filtering, prioritizing and grouping of messages, this is Boolean metric, Evaluator should check that event filters are working properly
- Event correlation/time this metric, evaluator should count the number of correlations that happened within the time period
- Buffering messages if management system is down this is Boolean metric, evaluator should verify buffering messages. Test should be executed on various client systems, at least one client running different OS.
- Presenting the events in different colors that indicates the severity of the event, Boolean metric, evaluator should confirming whether the events are presented with different colour/severity
- Providing event specific action that operator should execute Boolean metric, evaluator should open an alarm and check whether it contains operator action

Interoperability attributes:

- Collecting events/alarms from different systems on the network,

Interoperability metrics:

- Alarms/events arrived related to data exchangeability of [ISO/IEC 9126-2]. Evaluator should check that alarms related to data exchange between the server and client nodes arrived.
- Collecting information from application and system log files of client systems. Boolean metric, evaluator should check the alarms or application logs of NMA whether they contain alarms or entries from application and system log files.
- Collecting system messages of client systems. Boolean metric, evaluator should check the alarms or application logs of NMA whether they contain system messages entries from clients
- Collecting SNMP traps and variables of client systems. Boolean metric evaluator should check the alarms or application logs of NMA whether they contain SNMP trap messages entries from clients

- Collecting OS resource/performance metrics of client systems. Boolean metric, evaluator should check the alarms or application logs of NMA whether they contain performance alarms or entries from clients.

Security attributes:

- Securing communication with the agents (client systems)

Security metrics:

- Controlled secure access communication from agents (i.e. using HTTPS and SSL). Evaluator should check whether the communication with the client system is secure using secure protocols HTTP and SSL
- Auditing of secure access communication with agents similar to [ISO/IEC 9126-2] metric security auditing. Evaluator should check NMA application logs, whether they contain auditing information about client connections.
- Communicating via proxy servers and firewalls. Evaluator should check/analyze if events arrive via proxy servers and firewalls.

Reliability attributes and metrics of Network Management Application
Maturity attributes of Network Management Application

- Certification for Operating System i.e. Certificate for Microsoft Windows, was announced by HP as reliability improvement
- Existence and quality of analyzing, configurations and tracing tools

Maturity attributes of Network Management Application

- Existence and quality of stability indicators. Boolean metrics check that the application contains stability indicators.
- Readability of stability indicators. Boolean metrics check that stability indicators can give quality of software configuration tools. Boolean metric that checks existence of configuration tools.
- Existence of software updates

Fault tolerance attributes of OS:

- Error Avoidance

Fault tolerance metrics:

- Incorrect operation avoidance [ISO/IEC 9126-2]
- Breakdown avoidance [ISO/IEC 9126-2]

Recoverability attributes of Network Management Application

- Recoverability from software faults

Recoverability metrics:

- Availability of Network Management Application [ISO/IEC 9126-2]. Evaluator should analyze NMA log files of a production system, calculate operation time To, and repair time Tr of NMA.
- Mean downtime of Network Management Application [ISO/IEC 9126-2]. Evaluator should analyze NMA log files of a production system and calculate down time during a specified trial period.
- Restartability of the complete system caused by NMA [ISO/IEC 9126-2]. Evaluator should analyze log files of a production system and counts the

number of successful restarts (providing running NMA application) of complete system and total number of restarts.
- Restartability of NMA services derived from [ISO/IEC 9126-2]. Evaluator should analyze log files of a production system and counts the number of successful restarts (providing running NMA application) of NMA services and total number of restarts of NMA services.

Efficiency attributes for Network Management Application
- Event processing
- CPU efficiency
- Memory efficiency
- Hardware system requirements

Efficiency metrics for Network Management Application
Time behavior metrics for Network Management Application
- Response time of starting the Administration Interface. Evaluator should measures time needed to start NMA Administrator User Interface several times (i.e. ten times) and calculate the average value.
- Response time of starting and stopping the application processes. Evaluator should measures time needed to start NMA application processes several times (i.e. ten times) and calculate the average value.
- Amount of CPU time the system spends executing Network Management Application processes. In this case we will have time spent on NMA application processes/ total uptime of the system

Resource utilisation metrics:
- Number of events processed per time unit. Evaluator should generate test events and calculate how many of created test events are processed by the system.
- CPU Utilization caused by Network Management Application processes. Evaluator should run OS performance commands (*top* command on HP UX or *prstat* command on Solaris OS) that present CPU usage by different processes.
- Memory usage caused by Network Management Application processes. Evaluator should run OS performance commands (*top* command on HP UX or *prstat* command on Solaris OS) that present memory usage by different processes

DB development tool

We assessed the following versions of TOAD:
- TOAD Oracle Freeware version 8.5.0.50 g, we started with this version of TOAD, but since we did not have Oracle Database in our testing environment, we switched to version of TOAD for MySQL
- TOAD for MySQL version 2.0.3.795

Functionality attributes and metrics of development tool
Suitability attributes of DB development tool:

- Editor functionality for programmers
- Tools for building SQL queries
- Database reporting in different formats – HTML, PDF, XLS, JPG, RTF
- View and edit data types
- Compilation, debugging and execution of stored procedure, triggers, function and types
- Generation of Schema and Database scripts

Conversion of SQL statements to programming and scripting languages

Suitability metrics of DB development tool:

- Adequacy of the listed suitability attributes. This metric is derived from functional adequacy [ISO/IEC 9126-2:2001]. The metric is defined by [ISO/IEC 9126-2:2001] as X=1 –A/B, where:
    - A=Number of functions in which problems are detected in evaluation. We give the following grades 1 for unsuccessful, 0,5 for partly successful and 0 for successful.
    - B=Number of evaluated functions

  Functions that succeeded are counted as 1, not succeeded functions are counted as 0 and partly succeeded functions are counted as 0,5.

We evaluated the following functions defined:

- Editor functionality for programmers Succeeded, editor can be started from the standard toolbar.
- View and edit data types – Database browser functionality Succeeded, we were able to open and edit MySQL Databases
- Tools for building SQL queries Succeeded, we executed trivial SQL queries as `use liverepository; SHOW TABLES;`.
- Database reporting in different formats – HTML, PDF, XLS, JPG, RTF Partly Succeeded we received DB report in HTML format, other formats were mentioned in the marketing documentation, but they were not available in the application.
- Compilation, debugging and execution of stored procedure, triggers, function and types Partly Succeeded it was possible to create stored procedure and function.
- Conversion of SQL statements to programming and scripting languages. Not Succeeded option is not available in the freeware version of TOAD for SQL.

We calculated the following numbers: A=2 B=6 X=0,667

- Completeness of the listed suitability attributes. This metric is derived from functional implementation completeness [ISO/IEC 9126-2:2001]. In the absence of functional requirement specifications, we could not execute completeness tests. Therefore, we focused on the functional adequacy in the previous paragraph.
- Supported Databases by Database Management tool. We can define this metric like X=A/B, where A is number of supported Databases and that is 1 per tool, B is total number of Databases, where we count most often industry used databases (Oracle, SQL Server, Sybase and DB2) is 5, so X=A/B=0,2. The remark is that TOAD versions for other database exist, but it is always one database per tool. This metric may be defined in other way like Boolean metric called "Support for different databases" in that case we will have 0 for tool supporting only one database type and 1 for tool supporting different database types. This is relatively low value, but

the fact is that most of the DB management tools will score low because they dedicated to one database only.

Interoperability attributes:

- Connection to databases
- Network/OS tools support

Interoperability metrics:

- Database update, metric that verifies that the tool is capable to execute basic Database operations like “add” and “delete” records. Metric is defined as X=A/B where A is number of supported operations and B is number of total operation. In this case we will have A=2, B=2 and X=1
- Connection to Database and possibility to open Database files. TOAD for MySQL can connect to database and open database files. So the values will be A=2, B=2 and X=1. TOAD for Oracle can connect to database only in this case we will have the following values A=1, B=2 and X=0,5
- Support/contain Operating System and Network tools. E.g. FTP, telnet, rexec, ping, tnsping, UNIX crontab interface Windows Service Management. Metric that verifies presence of these tools in a way X=A/B, where A is number of tool supported = 0 (for these freeware versions) and B is 7, thus X=0

Other Functionality metrics not applicable for TOAD but applicable for the other similar products on the market:

- Data export, exporting data to one of the common formats MS Access, MS Excel (or csv), MS WORD, PDF. This metric can be defined as X=A/B where A is number of supported common export formats and B is total number of common formats
- Data extract (backup of the meta data and tables). Boolean metric that will verify whether these functionality is present.
- Data Import MS Access, MS Excel and other popular formats to database tables. This metric can be defined as X=A/B where A is number of supported common import formats and B is total number of common formats
- Data/DB Comparer, compare and synchronize context/structure of databases. Boolean metric that will verify whether these functionality is present and working properly

Reliability attributes and metrics for DB development tool

Recoverability attributes for DB development tool:

- Data preservation from e faults
- Frequency of software faults during the usage

Recoverability metrics for DB development tool:

- Data preservation in case of abnormal events, derived from Restorability [ISO/IEC 9126-2:2001]

This metric is defined by [ISO/IEC 9126-2:2001] as X=A/B where
A is number of abnormal events cases when data was not damaged
B is number of abnormal events cases tested as per requirements

We executed three test cases two times per case thus in total six times.

1) First case was stopping TOAD for MySQL application via task manager,
2) Second case was shutting down the system from a power button while tool has connections with databases
3) Third case was shutting down the system from power button while tool has connections with databases

In all six cases, we had a successful restoration and the connection to the databases at the moment of stopping the application was remembered, that results in: A=6, B=6 and X=1.

Usability attributes and metrics of DB development tool

Understandability attributes of DB development tool:

- Product demonstration
- Help menu
- Self-explanatory and consistency of user interface

Understandability metrics for DB development tool:

- Existence of product manual / help as part of the product software. Boolean metric that checks whether user manual exist as part of the product software. We executed these tests with both TOAD versions.

TOAD for Oracle:

We have checked the documentation links from TOAD Windows program menu (**Start> Programs> Quest Software> Toad Oracle Freeware> Documentation**) as shown on the figure bellow:

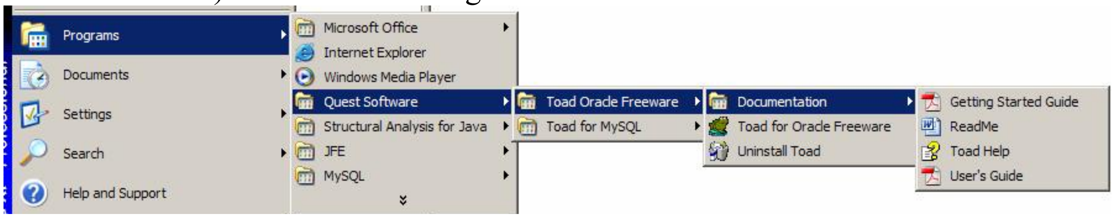
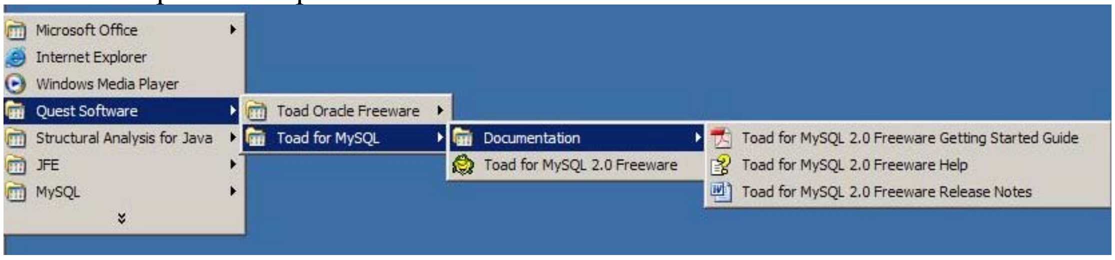


Figure 2 TOAD for Oracle Documentation

So the value for this metric, existence of product manual/help X=1

TOAD for MySQL:

We have a similar result for TOAD for MySQL, where documentation is included as part of the product software



Microsoft Office
Internet Explorer
Windows Media Player
Quest Software
Structural Analysis for Java
JFE
MySQL
Toad Oracle Freeware
Toad for MySQL
Documentation
Toad for MySQL 2.0 Freeware
Toad for MySQL 2.0 Freeware Getting Started Guide
Toad for MySQL 2.0 Freeware Help
Toad for MySQL 2.0 Freeware Release Notes

Figure 3 TOAD for MySQL documentation

The value for this metric, existence of product manual/help X=1

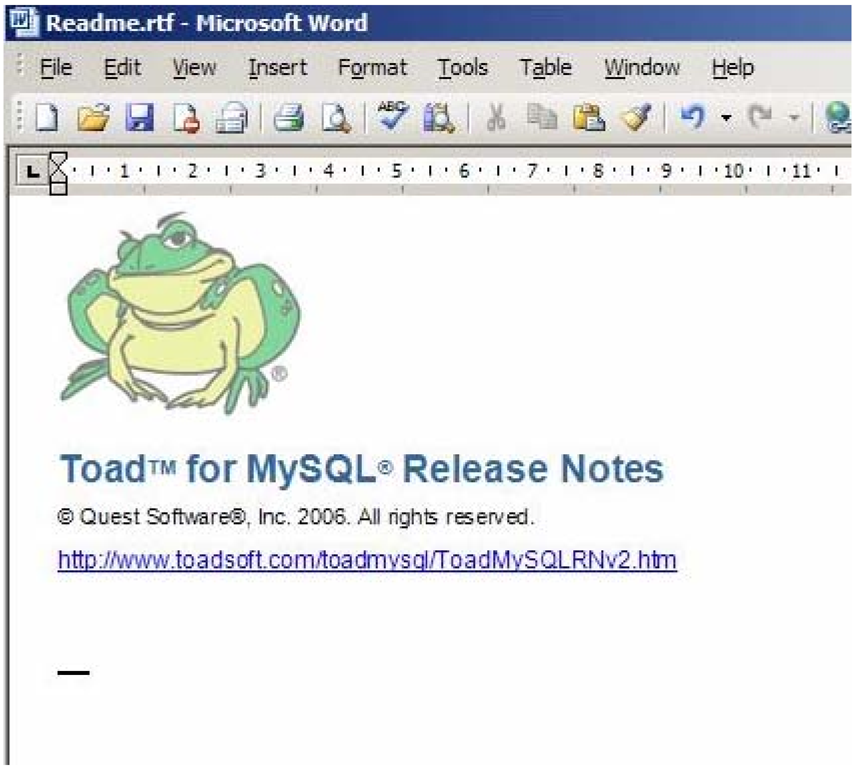
- Accessibility of product manual as part of the product software.

TOAD for Oracle:
From the four specified documents, only two were available. Therefore, we will have the following values:

- A number of existing product manuals = 2
- B number of total product manuals = 4
- X accessibility of product manual = A/B = 0,5

TOAD for MySQL:
We received the following values:

- A=Number of accessible product manuals/help = 2,5, grade of 0,5 was assigned to TOAD for MySQL release notes, because the document was almost empty containing a link to URL where the actual release notes are


Readme.rtf - Microsoft Word
File Edit View Insert Format Tools Table Window Help

Toad™ for MySQL® Release Notes

© Quest Software®, Inc. 2006. All rights reserved.

http://www.toadsoft.com/toadmysql/ToadMySQLRNv2.htm

Figure 12 TOAD for MySQL Release notes (Readme file)

- B=number of total product manuals =3
- X accessibility of product manual = A/B =0,83

- Existence of product manual on Internet.

TOAD for Oracle:
Product documentation for TOAD for Oracle exists on internet but is not publicly available. In order to get the documentation the user should register on the Quest Software (TOAD producer) web site. So the grade for this metric of TOAD for Oracle will be 0,5, because the User guide cannot be downloaded without registration, providing personal information on the web site of Quest Software.

TOAD for MySQL:

Product manual for TOAD for MySQL does not exist on the Quest Software web site. Therefore, the grade existence of product manual of TOAD for MySQL will be 0.

- Existence of demonstrations. Boolean metric check whether demonstration tutorials exist as part of the product software or on Internet. These tutorials do not exist for both of TOAD versions so our grade for this metrics will be 0.
- Self-descriptive User Interface (UI) functions. We define this metric as X=A/B, where
  A=Number of UI functions that are self descriptive to the user=11,5
  B=Total number of UI functions=18
  For this metric, we evaluated the following UI functions on the figure:

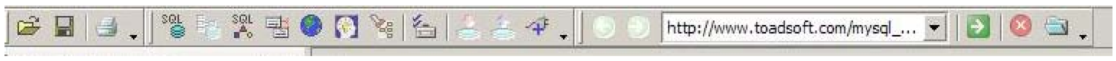


Figure 13 User Interface Functions

We have calculated the following numbers that are subjective:
A=11,5 B=18 and X=0,639

- Understandable input and output provided by the tool [ISO/IEC 9126-2:2001]. This metric cannot be executed as [ISO/IEC 9126-2:2001] described therefore we will try to give a subjective grade of 0,8 of 1, because it seems clear to us what the input and output for this product software are.

Learnability attributes of DB development tool:

- Product Manual

Learnability metrics of DB development tool:

- Ease of learning to perform a task in use [ISO/IEC 9126-2:2001], this is a bit difficult to measure in absence of the dedicated users that will be monitor and time needed to learn a function measured. Our impression after executing few operations on base of the Help is that we can grade this metric with 0,8 of 1
- Correctness of user documentation. We evaluated correct description of functions in one of the manuals. Metric is defined as X=A/B. Where:
  A= Number of functions correctly described
  B= Total number of the function described

Functions that succeeded are counted as 1, not succeeded functions are counted as 0 and partly succeeded functions are counted as 0,5.

For this metric, we evaluated "TOAD for Oracle Getting Started Guide Version 9.0" and "TOAD for MySQL Getting Started Guide version 2.0".

We executed the following procedures from the TOAD for Oracle Getting Started Guide Version 9.0" and we received the following results:

o Installation	Succeeded
o Installation log (Install. Log) creation	Succeeded
o Silent Install	Succeeded
o Uninstall (also executed in ease of uninstallation)	Succeeded

- Trial Version Registration Not succeeded, option is not available for the freeware version that we evaluated

So we calculated the following numbers for "TOAD for Oracle Getting Started Guide v. 9.0" A= 4, B=5 X=0,8

We executed the following procedures from "TOAD for MySQL Getting Started Guide Version 2.0"

- Online help, help selection Partly Succeeded, manual says that it is on the **Tools | Options | Interface | General,** but it is actually on **Tools | Options | Interface | Help System**.
- Online help, context sensitive Succeeded
- Online help, general information Succeeded
- Online help, keyword searching Succeeded
- Release Notes Succeeded
- Installing TOAD Partly succeeded getting started guide does not contain some screens that appear during the installation
- Uninstall TOAD for MySQL Succeeded
- Files Installed Succeeded
- Registering TOAD for MySQL Partly succeeded, Getting started guide does not specify how to get to the authorization key menu

Calculated number for "TOAD for MySQL Getting Started Guide Version 2.0" A= 7,5 B=9 and X=0,833

- Help accessibility [ISO/IEC 9126-2:2001]. The metric is defined by [ISO/IEC 9126-2:2001] as X=A/B, where

A number of tasks we choose here more for number of screens for which correct online help is located

B total number of screens that can appear

Example of screen without help button is TOAD server Login started with File > New Connection, presented on the following figure:

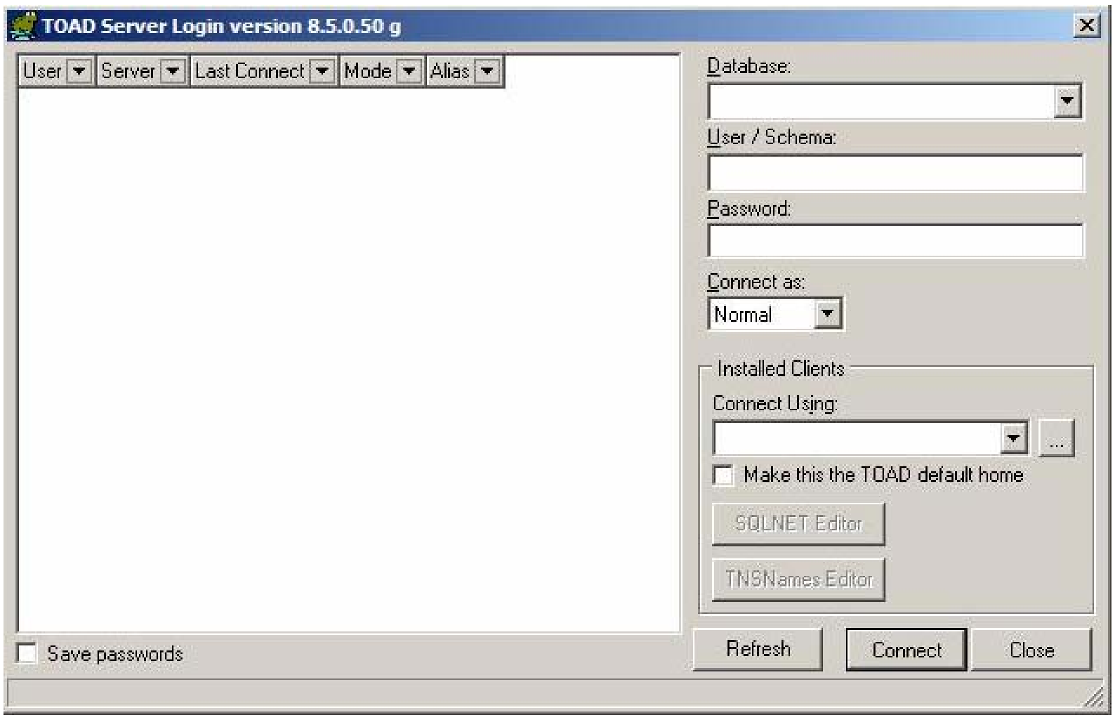
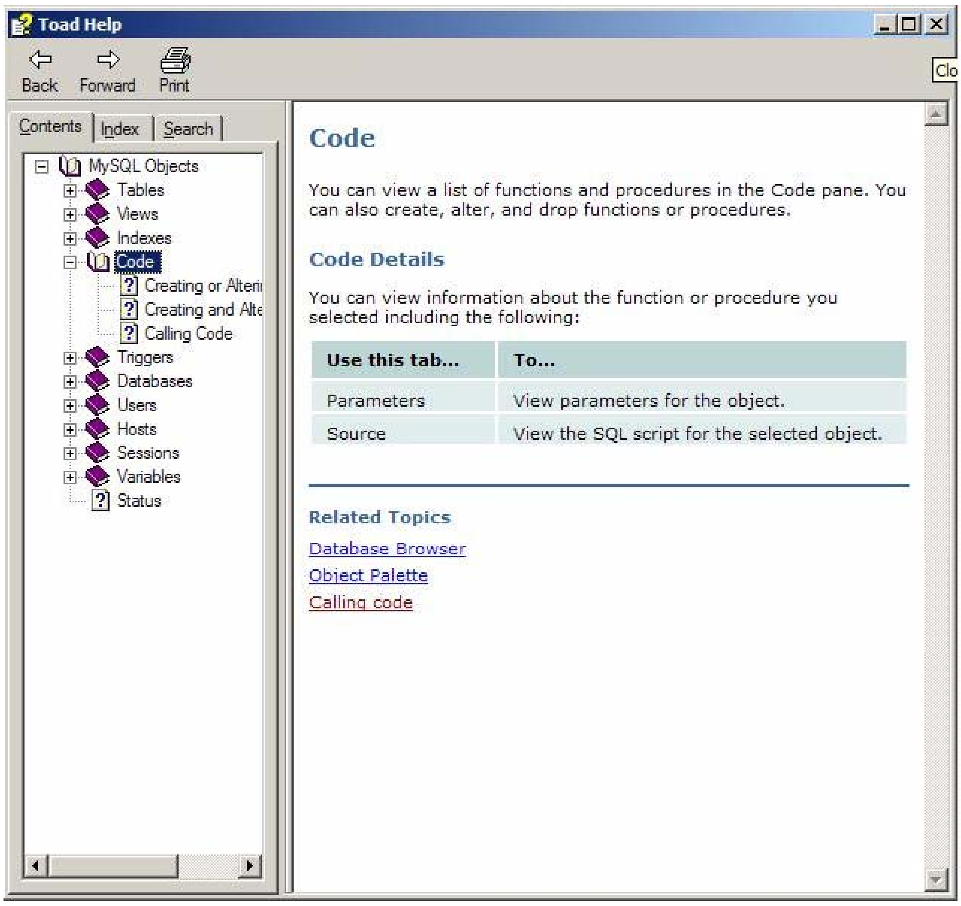


Figure 14 TOAD new connection

However, after pressing the F1 button the appropriate help menu appears.
We executed this operation on the 12 basic screen using TOAD for MySQL and we received appropriate help screen. The only remark is the help screens for Procedures and functions, where we received the following screen:



Code

You can view a list of functions and procedures in the Code pane. You can also create, alter, and drop functions or procedures.

Code Details

You can view information about the function or procedure you selected including the following:

Use this tab...	To...
Parameters	View parameters for the object.
Source	View the SQL script for the selected object.

Related Topics

Database Browser

Object Palette

Calling code

Figure 15 Help Menu for Function and procedures

Therefore, we will grade these two menus with 0,5 and the numbers will be A= 11 (10*1 +2*0,5), B=12 and X= 0,917

Efficiency attribute and metrics of DB development tool

Time behaviour attributes:

- Server and session statistics
- Processing times of functions

Time behaviour metrics:

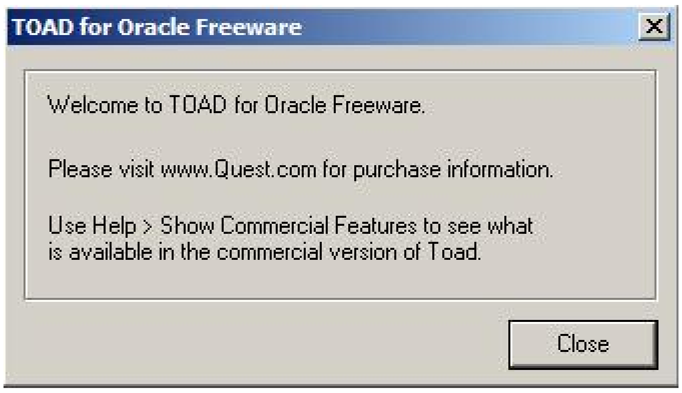
- Tool start time is defined by T= in seconds. We count the average value of start time we received the following values for

TOAD for MySQL:

T=4sec.

TOAD for Oracle is:

T=4 sec. with the remark that for TOAD for Oracle we should press Close button on a pop-up window to continue.


TOAD for Oracle Freeware

Welcome to TOAD for Oracle Freeware.

Please visit www.Quest.com for purchase information.

Use Help > Show Commercial Features to see what
is available in the commercial version of Toad.

Figure 16 Pop-up window appearing during TOAD start-up

- Time needed to open a small database of 40k. We executed this metric using TOAD for MySQL only. The result was:

T= 2,5sec.

Resource Utilization attributes:

- Hardware resource

Resource Utilization metrics:

- Minimum hardware requirements, we found these information in Getting Started guide and in the web page of the producer Quest Software

TOAD for Oracle has following hardware requirements

  - Required space on disk for installation 75MB
  - Required RAM memory 512 MB, 1GB recommended

TOAD for MySQL has the following hardware requirements:

  - Required space on disk for installation 44MB
  - Required RAM memory 256 MB, 512 MB recommended
  - Required CPU frequency 233MHz (minimal), 300MHz or higher recommended

- Memory usage, we checked the memory usage of the TOAD process toad.exe in the windows Task Manager as shown on the following figure:

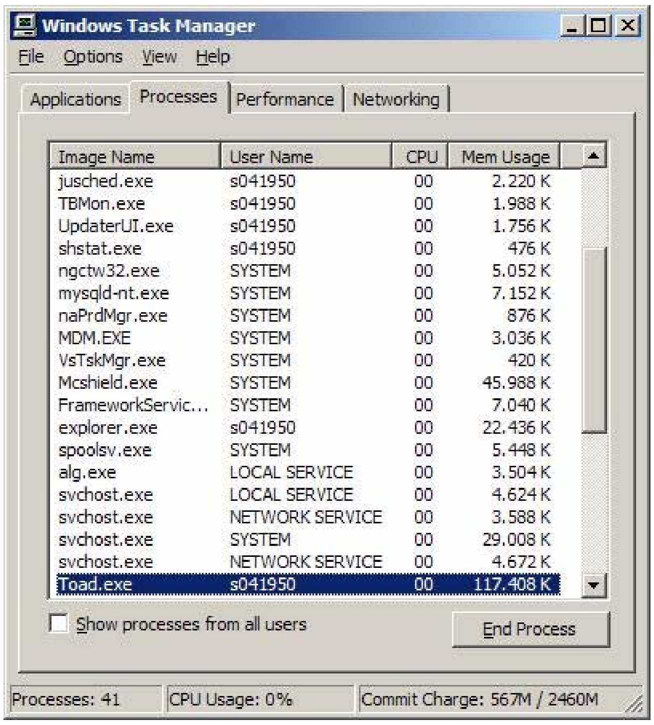


Image Name	User Name	CPU	Mem Usage
jusched.exe	s041950	00	2.220 K
TBMon.exe	s041950	00	1.988 K
UpdaterUI.exe	s041950	00	1.756 K
shstat.exe	s041950	00	476 K
ngctw32.exe	SYSTEM	00	5.052 K
mysqld-nt.exe	SYSTEM	00	7.152 K
naPrdMgr.exe	SYSTEM	00	876 K
MDM.EXE	SYSTEM	00	3.036 K
VsTskMgr.exe	SYSTEM	00	420 K
Mcshield.exe	SYSTEM	00	45.988 K
FrameworkServic...	SYSTEM	00	7.040 K
explorer.exe	s041950	00	22.436 K
spoolsv.exe	SYSTEM	00	5.448 K
alg.exe	LOCAL SERVICE	00	3.504 K
svchost.exe	LOCAL SERVICE	00	4.624 K
svchost.exe	NETWORK SERVICE	00	3.588 K
svchost.exe	SYSTEM	00	29.008 K
svchost.exe	NETWORK SERVICE	00	4.672 K
Toad.exe	s041950	00	117.408 K

Processes: 41 | CPU Usage: 0% | Commit Charge: 567M / 2460M

Figure 17 TOAD memory usage

TOAD for Oracle has memory usage of 39 MB when running empty, without any connections to databases. We cannot give further numbers because we did not have an Oracle DB available at the testing environment.
TOAD for MySQL has memory usage of 13 MB when running empty, without any connections to databases. After opening a small database, the memory usage became 108MB. After opening, a second connection to an almost empty database memory usage becomes 120MB.

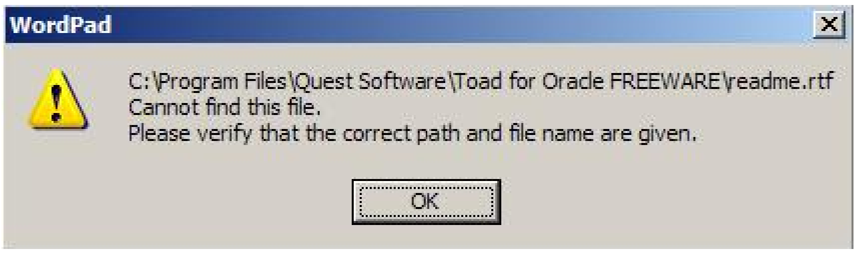
Portability attributes and metrics of DB development tool

Installability attributes of DB development tool:

- Installation package availability
- Supported Software Environments
- Supported connecting database and supported versions of these databases
- Database client requirements

Installability metrics of DB development tool:

- Ease of manual installation [ISO/IEC 9126-2:2001]. Installation is *easy* as described in [ISO/IEC 9126-2:2001] that is 3 of 4 on [ISO/IEC 9126-2:2001] scale. On the end of the installation, we select that readme file should be open, but the installation program could not find this file because it was not on the specified location. So we received the following error:


WordPad

C:\Program Files\Quest Software\Toad for Oracle FREEWARE\readme.rtf
Cannot find this file.
Please verify that the correct path and file name are given.

Figure 18 TOAD installation cannot find readme file

Therefore, our grade for ease of installation for TOAD for Oracle will be 0,65 of 1.

TOAD for MySQL:

When installing this product we did not have this kind of issues, so the grade value will be 0,75

- Ease of manual uninstallation, on base of the "Ease of user's manual install operation" metric of [ISO/IEC 9126-2:2001] we defined this metric.

TOAD for Oracle:

Based on the scale as defined in [ISO/IEC 9126-2:2001] we can grade ease of uninstallation as *easy*, it contains uninstall button from the Windows program menu. So the grade will be X=0,75 of 1.

TOAD for MySQL:

We can also grade ease of uninstallation of TOAD for MySQL as easy with a remark that this application does not contain uninstall link in the Windows program menu. Therefore, the grade for ease of uninstallation will be 0,65.

- Ease of Setup Retry [ISO/IEC 9126-2:2001]

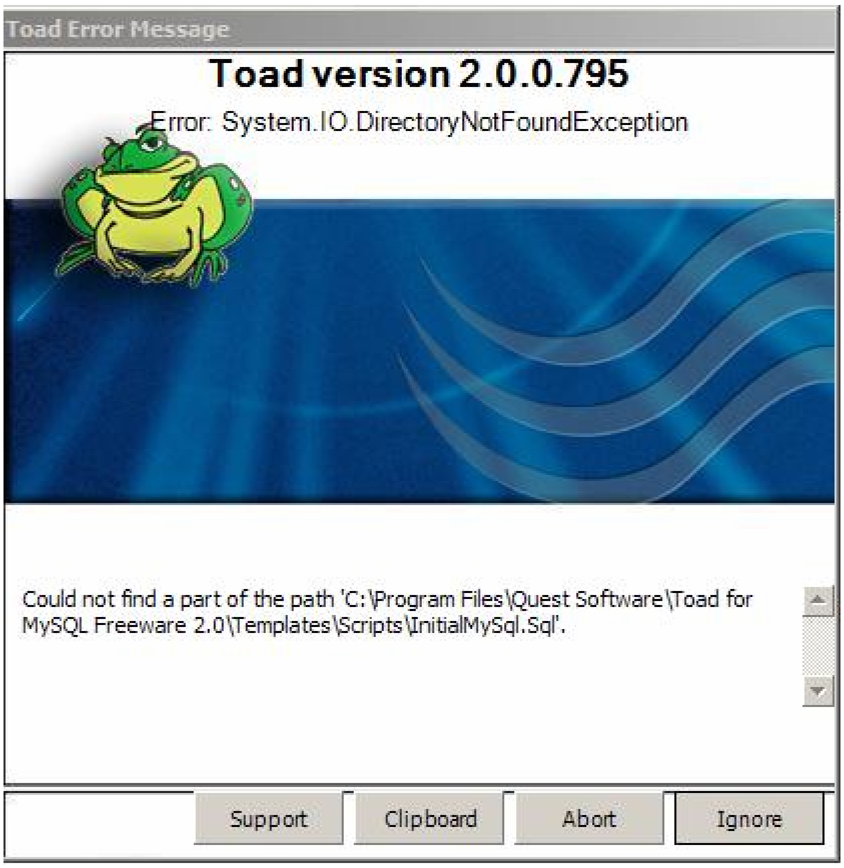
This metric is defined by [ISO/IEC 9126-2:2001] as X=1-A/B where

A is number of cases in which user fails in re-trying set-up during set-up operations

B total number of cases in which user attempt to re-try setup during set-up operations

TOAD for MySQL:

We executed 4 cases using TOAD for MySQL thus B=4. Only one was partly unsuccessful when installation path was not on the default path "C:\Program Files\Quest Software\Toad for MySQL Freeware 2.0" A=3,5 and then X=0, 875. In that case, after opening a database we received the following error message that indicates hard coded paths in the tool:


Toad Error Message

Toad version 2.0.0.795

Error: System.IO.DirectoryNotFoundException

Could not find a part of the path 'C:\Program Files\Quest Software\Toad for MySQL Freeware 2.0\Templates\Scripts\InitialMySql.Sql'.

Figure 19 Error message after installing TOAD on non-default path

- Supported Operating Systems. Check product documentation, which Operating Systems are supported. We define this metric as X=A/B where:

A=Number of supported Operating Systems =1 Windows only

B=Commonly used Operating Systems families: Windows, Linux, Solaris, UNIX (POSIX)=4

Consequently X=A/B=0,25

SA4J

Functionality attributes and metrics for code analyzer tool

Functionality attributes for code analyzer tool

- Stability analysis of application structure
- Anti-pattern detection of application packages
- Package analysis of application packages
- What-if impact analysis of components

Functionality metrics for code analyzer tool:

- Completeness of the listed major functions. This metric is derived from functional implementation completeness [ISO/IEC 9126-2:2001]. In order to verify this function we need the functional requirements specifications, which is not publicly available. Therefore, we cannot execute this test. An alternative is to verify application features mentioned in the marketing materials.

- Verification of the major product functions:
    - Analyse of randomly selected application
    - Anti-patern detection of randomly selected application
    - What-if impact analysis of randomly selected application

To verify these features we downloaded random applications from Internet JavaFE and Smart Chat, we were able to executed analysis with SA4J. Because of this, we could verify the above-mentioned functionality attributes.

- Supported input in the case of code analyzer tool the input can be binary files or code, so we can define a metric as X=A/B where
    - A= number of supported input=1 since SA4J does not analyze code it analyses Java classes only
    - B=total number of inputs = 2
    - X=A/B=0,5

- Support for different programming languages, this metric can be define in two ways:
    - Boolean X=0 it supports only one language, X=1 supports more than one programming language. The result of SA4J in this case will be X=0
    - In a way X=A/B where A is number of supported programming languages and B is total number of programming languages, that is kind of vague to count, we can eventually focus on most popular languages like Java, C++ and Visual Basic. The result of SA4J in this case will be X=1/B or X=1/3=0,333

Other code analyzer metric not applicable for SA4J, but applicable for the other products in the same domain:

- Detection of duplicated code. Run the tool using sample code containing duplicated code and check whether the tool can detect duplicated code.
- Removal of duplicated code. Run the tool using sample code containing duplicated code and check whether the tool can remove duplicated code.
- Coding standard check, check source code against definable coding standard. Boolean metric that verifies the coding standard functionality. Define couple of coding standard rules in the tool. Run the tool using sample code containing violations of defined coding standards and check whether the tool can detect violation of the defined coding standards.
- Integration with development tools (e.g. Borland JBuilder, Oracle9i JDeveloper e.t.c). The metrics can be defined as X=A/B, where A is number of supported development tool and B is number of commonly used tools
- Code metrics based analysis (e.g. Number of statements per method, Number of statements per class, Static Path count, Code Nesting, Cyclomatic Complexity e.t.c.). This metric can be defined as X=A/B, where A is number of supported metrics by the tool and B is total number of metrics
- Generation of reports in the standard formats like text, HTML, pdf, csv. This metric can be defined as X=A/B, where A is number of supported report formats, B is number of common report formats =4
- Compatibility check of binaries and code with the older releases of the programming language. Boolean metric that checks whether code analyzer

can execute a compatibility check with older version code and binaries from the same programming language

Reliability attributes and metrics for code analyzer tool:

Recoverability attributes for code analyzer tool:

- Data preservation

Recoverability metrics for code analyzer tool:

- Data preservation in case of abnormal events, derived from Restorability [ISO/IEC 9126-2:2001]

This metric is defined by [ISO/IEC 9126-2:2001] as X=A/B where
A is number of abnormal events cases when data was not damaged
B is number of abnormal events cases tested as per requirements
We executed three test cases two times per case thus in total 6 times.

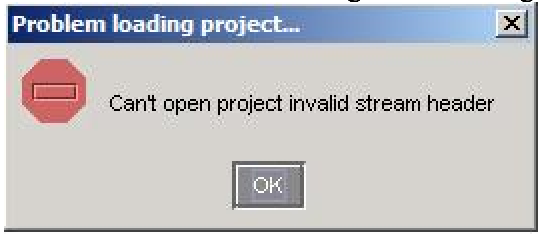
1) First case was stopping SA4J application via task manager,
2) Second case was shutting down the system from a power button while application was running.
3) Third case was shutting down the system while application was running

In all 6 cases we had no data damage restoration that results in:
A=6, B=6 and X=1

Fault tolerance metrics for code analyzer tool:

- Incorrect operation avoidance

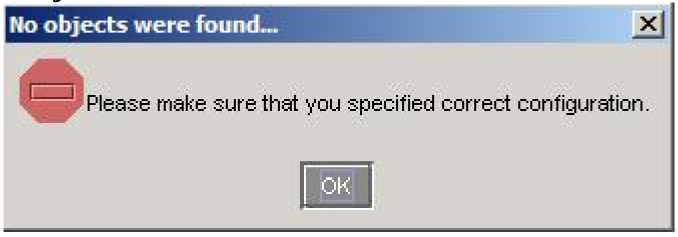
This metric is defined by [ISO/IEC 9126-2:2001] as X=A/B where
A is number of avoided critical and serious failures occurrences
B is number of executed test cases of incorrect operating pattern
Example of these cases is trying to open a project file with wrong file extension. Then we will receive the following error message:


Problem loading project...

Can't open project invalid stream header

Figure 20 Opening project file with wrong extension

Another example is specifying wrong code files, when running **File> New Java Project**:


No objects were found...

Please make sure that you specified correct configuration.

Figure 21 Creating new project with wrong file

Third example is when we try the option **File > Open Project,** directories that do not contain files with .saj extension are not listed, as shown on the following figure:

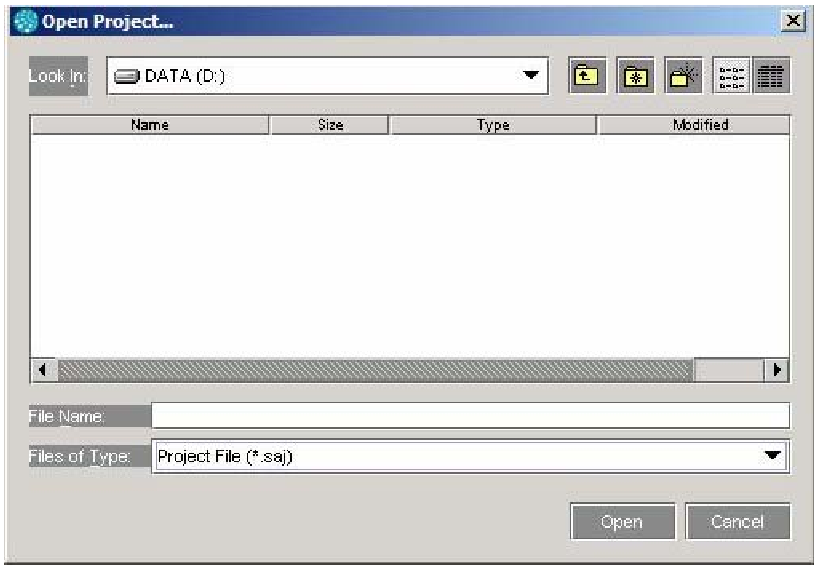


Figure 22 Open Project in SA4J

On base of three two examples, we can give the following numbers:
A=3, B=3 and X=1

Usability attributes and metrics of analyzer tool

Understandability attributes of code analyzer tool:

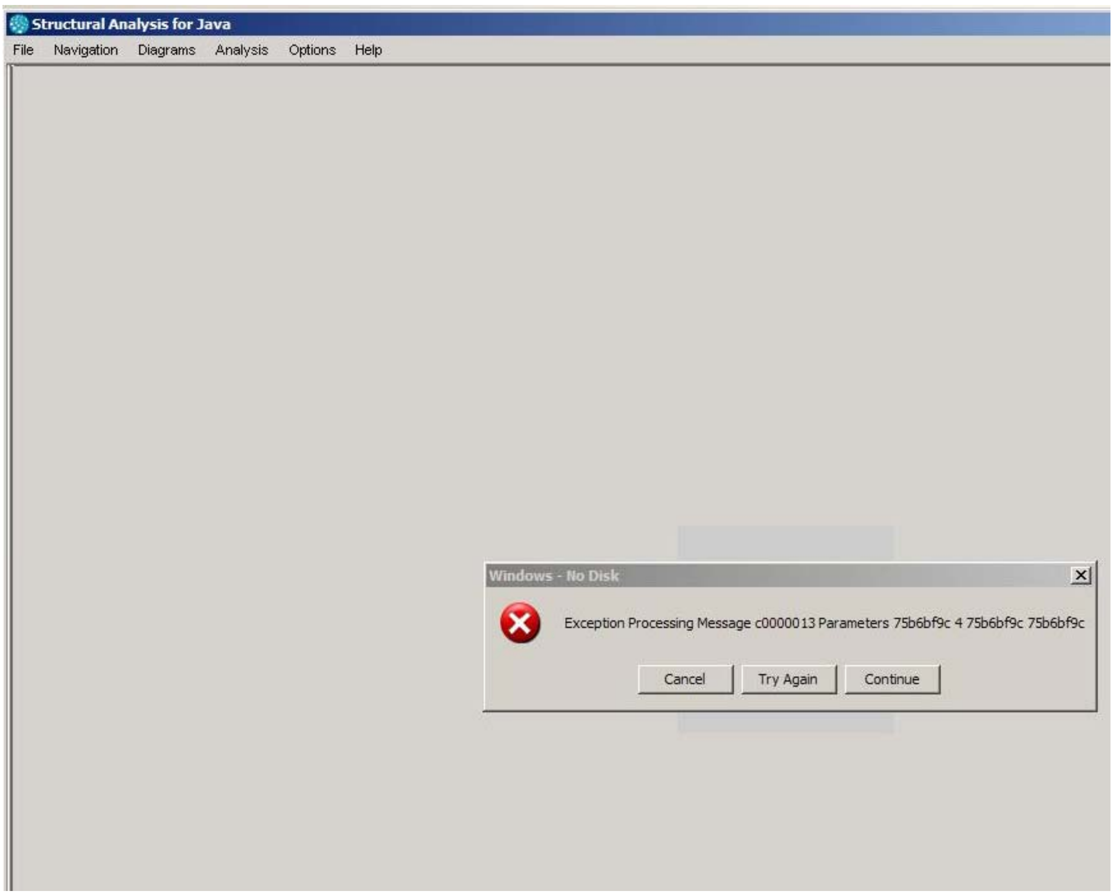
- Product demonstration
- Help menu

Understandability metrics of analyzer tool:

- Completeness (covering all available operations) of product manual [ISO/IEC 9126-2:2001]

Evaluation criteria: Product manual used was "Structural Analysis for Java Tutorial" supplied with the application. Metric is defined in [ISO/IEC 9126-2:2001] X=A/B where the value of X closer to 1 means high quality.
A= Number of functions understood=15
B=total number of functions = 18
X=0,833

Example of a function available but not properly described is opening a new project. When we execute this function, the following pop-up window appears:


Structural Analysis for Java
File Navigation Diagrams Analysis Options Help
Windows - No Disk
Cancel Try Again Continue

Exception Processing Message c0000013 Parameters 75b6bf9c 4 75b6bf9c 75b6bf9c

Figure 4 SA4J issue opening new project

After we press, the button "Continue" for three times the application continues.

Another example of this kind of issue is Java > Diagram option presented on the following figure of the manual:

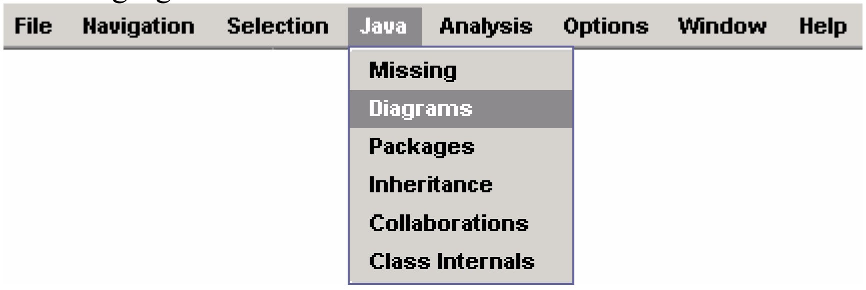


Figure 5 SA4J interface as described in the tutorial

The Java menu does not exists in the latest version of the tool. The interface now has the following menus:

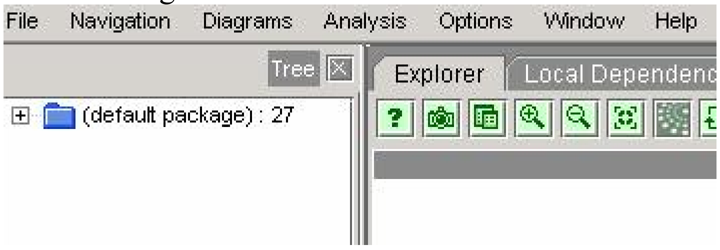


Figure 6 Actual SA4J interface

- Understandable user interface functions, “function understandability” in [ISO/IEC 9126-2:2001]
Metric is defined in [ISO/IEC 9126-2:2001] X=A/B
A=14=Number of UI function whose purpose is correctly understood by the user
B=21=Number of UI functions available from UI
X=0.667

- Understandable input and output provided by the tool [ISO/IEC 9126-2:2001] X= A/B where A is number of input and outputs that user successfully understands A depends of the user knowledge of structures, thus we cannot measure this metric objective

Learnability attributes of analyzer tool:

- Product Manual

Learnability metrics of analyzer tool:

- Ease of function learning [ISO/IEC 9126-2:2001] in our case it does not take too long to learn a function, as a subjective case we can give a grade X=0.7 of 1
- Existence of product manual/help. Boolean metric, checks whether user manual/help exist as part of the product software or on Internet. X=1 product manual exist as part of the software installation package.
- Ease of learning to perform a task in use [ISO/IEC 9126-2:2001] this is also a subjective function same as Ease of function learning and also the output is in time not very relevant for getting an appropriate result.
- Effectiveness of the user documentation [ISO/IEC 9126-2:2001]

Here we will use the same data as for completeness of user documentation While evaluating we gave 1 for a function that is completely good described, 0.5 for a function that is available but on other position in user interface as a result of this classification we received the following numbers. A=13; B=18
X=0,72

- Help accessibility on application screens derived from “help accessibility” [ISO/IEC 9126-2:2001]

Metric is defined by [ISO/IEC 9126-2:2001] as X=A/B, where
A number of tasks we choose here more for number of screens for which correct online help is located = 7. Our impression is that only the basic screens (tabs) of the application (Explorer, Local Dependencies, Global Dependencies, Skeleton, What if and Summary has available help, rest of the screens usually do not have help.
B total number of screens that can appear=15
X=0,467

Efficiency attributes and metrics

Time behaviour attributes:

- Processing times of functions and operations

Time behaviour metrics:

- Response time to execute application operations derived from [ISO/IEC 9126-2:2001]

  Here we can measure a response time for opening an example project and response time to start the application. On a system with 1GB of Ram and Pentium 4 CPU on 3 GHz, the response time to open an example project was 2sec, time needed to start the application was 7sec, and these values are average after 10 measurements.

Task \PC configuration	1 GB RAM, P4 3GHz	500 MB RAM, AMD 2,2 GHz
Start application	7 sec.	7 sec.
Open Project	2 sec.	2 sec.

Table 1 Response times

Resource utilization attributes for code analyzer tool:

- System Memory

Resource utilization metrics for code analyzer tool:

- Memory usage in use, this metric is similar to maximum memory utilization. We tried kind of stress usage scenario, opening many different windows having an open project, in this case. When the project is not open application uses, about 30MB of memory after opening a project a memory usage goes to 60MB. Memory usage caused by SA4J processes was about 92 MB with one project open and many options active. We manage to get a memory usage of 165 MB reopening the same project several times. Our impression is that this tool can use up to 200 MB of memory or even the usage can extend further, because application has an indicator that gives 133MB as maximum by default, when we had memory usage of 164 MB in use this indicator was reporting 113M of 148M.

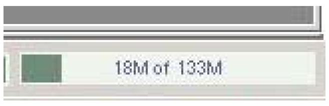


Figure 7 Memory usage indicator SA4J

Portability attributes and metrics of code analyzer tool

Installability attributes of code analyzer tool

- Supported Software environment
- Other software requirements i.e. Acrobat for reading the documentation

Installability metrics for code analyzer tool:

- Supported Operating System

  Here we can have the following metric X=A/B, where:

  A=Supported operating systems for this product are Windows, Solaris and Linux =3

  B if we say that total expected OS to be supported are these OS versions and other UNIX versions we can derive =4

  X=0,75
- Supported web browsers

For this metric, we can use the similar approach X=A/B where:
A supported web browsers for this product are Internet Explorer and Netscape=2
B if we say that total expected browsers to be supported are these browsers versions A+ Mozilla, that is very popular browser at this moment, so B=3
X=0,667

- Ease of user's manual install operation [ISO/IEC 9126-2:2001]

Installation is *easy* as described in [ISO/IEC 9126-2:2001] that is 3 of 4 on [ISO/IEC 9126-2:2001] scale containing *very easy, easy, not easy* and *complicated*. So our grade will be 7,5 of 10 or 0,75 of 1. We can also test further based on [ISO/IEC 9126-2:2001] to try different installation cases. We executed 4 times setup and one was on different location on the file system all 4 times were successful
- Ease of uninstallation, on base of the Ease of installation metric of [ISO/IEC 9126-2:2001] we defined this metric. Based on the scale as defined in [ISO/IEC 9126-2:2001] we can give 6,5 of 10 or 0,65 of 1 because on the program menu we do not have uninstall link, so on a windows system we should use Add or Remove Programs menu to uninstall this application.
- Ease of Setup Retry [ISO/IEC 9126-2:2001]

This metric is defined by [ISO/IEC 9126-2:2001] as X=1-A/B where
A is number of cases in which user fails in re-trying set-up during set-up operations
B total number of cases in which user attempt to re-try setup during set-up operations
We executed 4 cases thus B=4. Only one was unsuccessful when the application was open setup cannot be executed, so in that case it was unsuccessful this is expected behavior but we can say A=1 and then X=0, 75.

Microsoft Word

Functionality attributes and metrics of Office application

Functionality attributes of Office application:
Suitability attributes of Office application:

- Documents editing
- Document formatting
- Spelling and grammar checking of documents
- Supporting other files into the documents i.e. jpg pictures, excel sheets
- Convert documents do other format i.e. PDF, XML, rtf

Suitability metrics of Office application:

- Evaluate adequacy of major project functions, metric derived from functional adequacy [ISO/IEC 9126-2]. Evaluator should verify whether suitability functions are running properly. The metric is defined as X=1-A/B where, A is number of functions where problems are detected during evaluations, B is number of functions evaluated. We evaluated the following functions:
    - Documents editing
    - Document formatting
    - Spelling and grammar checking of documents
    - Supporting other files into the documents i.e. jpg pictures, excel sheets

    - Convert documents to other format i.e. PDF, XML, rtf

  We received the following results: A =4,33 B=5 X=0,935. We graded with 0,33 the document conversion function because evaluated MS Word version 2003 does not support the conversion to PDF format.
- Completeness of the listed suitability attributes, metric derived from functional implementation completeness [ISO/IEC 9126-2]. Evaluator should evaluate if all functions described in the requirement specification are implemented. This metric is defined same as the previous one X=1-A/B with the only difference that B is number of functions described in requirement specifications. In absence of the official requirements specification document, we cannot evaluate this metric.
- Support of specific/additional text editor features. Evaluator should check help or product documentation whether these additional features are supported by Office application. If features are supported then evaluator should verify that they are running properly. This metric can be defined as X=A/B, where:
    - A is number of text editor features supported
    - B is total number of standard text editor feature

    We identified the following list of additional text editor features available on Wikipedia:
    - Autocomplete involves the program predicting a word or phrase that the user wants to type in without the user actually typing it in completely
    - Autoreplace automatic replacement of a particular string with another one, usually one that is longer and harder to type, as "myname" with the full name of the author
    - Text search for searching a computer-stored document
    - Grammar checker design feature or a software program designed to verify the grammatical correctness or lack of it in a written text

    On base of this we can give the grade B=4, A=3,5 because autoreplace feature is not present as automatic replace of words or names, but it can be configured using the other tools as macros. Consequently X=0,875

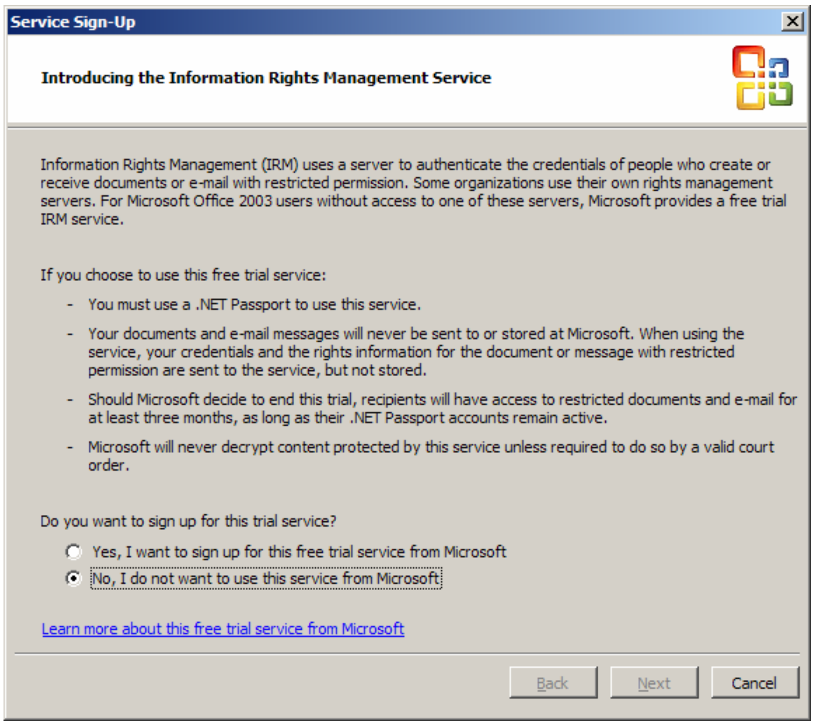
Security attributes of Office application:

- Security vulnerabilities brought as part of the macros
- Confidential sharing of documents
    - digital signature,
    - documents in reading mode
    - assigning permissions on documents

Security metrics of Office application:

All security metrics should be evaluated in a same manner; evaluator should check help or product documentation whether related features are supported by Office application. Then he should verify if related features are working properly.

- Assigning permissions of shared documents. We can define this metric as Boolean. In order to use permission MS Words requests installation of the latest version of Windows Rights Management Client. Thus, application cannot be used immediately. After installation of the latest Windows Rights Management Client, Word requires Sign-up and usage of .Net Passport for this service.


Service Sign-Up

Introducing the Information Rights Management Service

Information Rights Management (IRM) uses a server to authenticate the credentials of people who create or receive documents or e-mail with restricted permission. Some organizations use their own rights management servers. For Microsoft Office 2003 users without access to one of these servers, Microsoft provides a free trial IRM service.

If you choose to use this free trial service:

- You must use a .NET Passport to use this service.
- Your documents and e-mail messages will never be sent to or stored at Microsoft. When using the service, your credentials and the rights information for the document or message with restricted permission are sent to the service, but not stored.
- Should Microsoft decide to end this trial, recipients will have access to restricted documents and e-mail for at least three months, as long as their .NET Passport accounts remain active.
- Microsoft will never decrypt content protected by this service unless required to do so by a valid court order.

Do you want to sign up for this trial service?

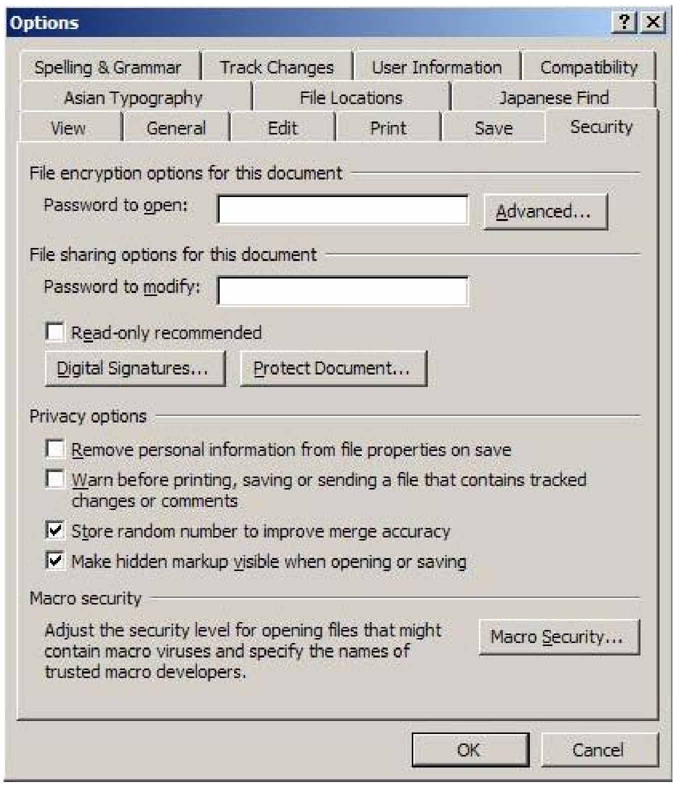
Yes, I want to sign up for this free trial service from Microsoft [✓ No, I do not want to use this service from Microsoft]

Learn more about this free trial service from Microsoft

Figure 23 Sign-up for Information Rights Management Services

Without signing in and providing .NET passport credentials, it is not possible to use this feature. The whole procedure to use this feature seems to us complicated and relying on usage of other Microsoft tools, therefore we will grade it with X=0,5.

- Document corruption prevention, derived from data corruption prevention [ISO/IEC 9126-2]. We will use the same metric definition as [ISO/IEC 9126-2] X=1-A/N, where A is number of times that a major data corruption event occurred and N is Number of test cases tried to cause data corruption event. [ISO/IEC 9126-2] defines additional formula for minor corruption events that can be the following Y=1-B/N, where B is number of times that a minor data corruption event occurred. We executed 4 test cases that were shutting down the application and the complete system with MS Word application open and document saved and unsaved. In test cases with saved document did not we have any data corruption events. In the other two test cases with unsaved document, we have minor data corruption events. So our numbers will be N=4, A=0 B=2, values for X and Y will be X=1 Y=0,5.
- Open and Modify document protection. Metric can be defined as X=A/B, where A is protection that editor support and B is total number protection = 2.


Options

Spelling & Grammar | Track Changes | User Information | Compatibility
Asian Typography | File Locations | Japanese Find
View | General | Edit | Print | Save | Security

File encryption options for this document
Password to open:
Advanced...

File sharing options for this document
Password to modify:

☐ Read-only recommended

Digital Signatures... | Protect Document...

Privacy options
☐ Remove personal information from file properties on save
☐ Warn before printing, saving or sending a file that contains tracked changes or comments
☑ Store random number to improve merge accuracy
☑ Make hidden markup visible when opening or saving

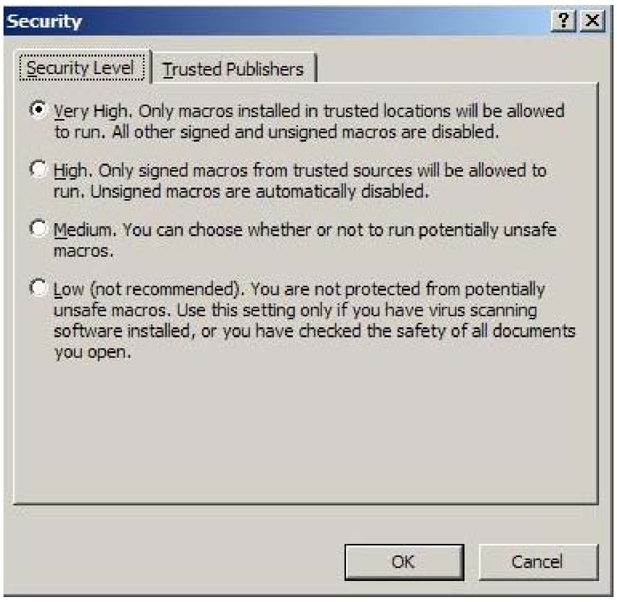
Macro security
Adjust the security level for opening files that might contain macro viruses and specify the names of trusted macro developers.
Macro Security...

OK | Cancel

Figure 24 Open and Modify permission

We tried this protection mechanism and we received the following numbers A=2 B=2 and X=1

- Macros protection. We will define this metric as Boolean, verifying whether the product offers a protection from macros. In case of MS Word 2003, macro protection exists,


Security

Security Level | Trusted Publishers

◉ Very High. Only macros installed in trusted locations will be allowed to run. All other signed and unsigned macros are disabled.

○ High. Only signed macros from trusted sources will be allowed to run. Unsigned macros are automatically disabled.

○ Medium. You can choose whether or not to run potentially unsafe macros.

○ Low (not recommended). You are not protected from potentially unsafe macros. Use this setting only if you have virus scanning software installed, or you have checked the safety of all documents you open.

OK | Cancel

Figure 25 Macro security of MS Word 2007

so our grade for this metric will be 1.

Usability metrics of Office application

Understandability attributes:

- Self-descriptiveness of the user interface
- Product demonstrations

Understandability metrics:

- Evident tools initially derived from "Evident functions" of **[ISO/IEC 9126-2]**. Evaluator should check if toolbar functions are clear to him on base of the toolbar icons. This metric can be defined as X= A/B, where
  - A is number of tools identified by the user
  - B is total number of tools.

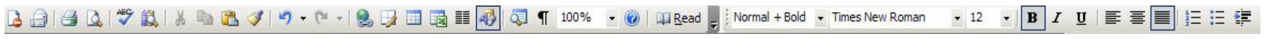


Figure 26 MS Word 2003 Tools

For MS Word 2003 we calculated the following numbers: A=14, B=19 X=0,736

- Existence of demonstrations explaining the product functions. Evaluator should check whether demonstration tutorials exist as part of the product software or on Internet. Three values metric, with values:
  - X=0 if demonstration does not exist at all
  - X=0,5 if demonstrations exist on Internet, but does not exist as as part of the product software
  - X=1 if demonstration exist as part of the product software

  In case of MS Word we will have X=0,5.
- Demonstration accessibility [ISO/IEC 9126-2]. Evaluator should check if he could access the demonstration tutorials. We will define this metric same as [ISO/IEC 9126-2] X=A/B, where A is number of demonstrations that user successfully access and B is number of demonstration available. MS Word 2003 contains 18 demonstrations (courses) B=18 that connect to tutorials from Internet. The number A=18 if a user has access to Internet and 0 if the user does not have access to internet. Therefore we will use the average value of these two numbers A=9, consequently X=0,5.

Learnability attributes for Office application:

- User Manual
- Online help

Learnability metrics for Office application:

- Effectiveness of user manual [ISO/IEC 9126-2]. Evaluator should try to execute functions described in the product manual and evaluate if description is correct. Defined by [ISO/IEC 9126-2] as X=A/B Word contains a large manual. We were able to execute 8 tasks about templates, headers described in the manual and all of them were successful, thus A=8 B=8 and X=1
- Help accessibility on application screens, derived from "help accessibility" [ISO/IEC 9126-2]. Evaluator should check how many of the existing screens have correct online help. This metric is defined as X=A/B where:
  - A is number of task for which correct online help is located
  - B is total number of tasks tested

We checked the existence of all the tasks available in the MS Word menu and we received the following numbers A = 24, B = 96 and X=0,25. This is surprisingly low grade, probably related to the fact that Microsoft did not provide help for some common functions like Copy, Paste or Insert Diagram.

Operability attributes of Office application:

- Grammar error correction
- Can user easily recover his/her input
- Self-explanatory error messages
- Existence of "Undo" operation/feature

Operability metrics of Office application:

- Grammar error correction, specific text editor metric derived from "Error correction" [ISO/IEC 9126-2]. Evaluator should measure time needed to correct grammar errors. This metric is defined as T= Tc – Ts, where:
    - Tc is time of completing correction of specified type errors of performed task
    - Ts time of starting correction of specified type errors of performed task

  The official [ISO/IEC 9126-2] metric should be used for specific errors like destroying data, input/output errors or operational situation. We used the metric for correction of typing errors that belongs to the group input/output errors. We identified three kinds of errors in this category:

1) Wrong use of capital letters, in case of these errors correction happens immediately thus time to correct it is T=0sec
2) Misspelled words, in case of these errors time to correct is T=2sec
3) Grammar errors as usage of passive voice, in case of these errors time to correct is T=20sec.

- Can user easily recover his/her input, sub-metric of "Error correction in use" [ISO/IEC 9126-2]. Evaluator should create several errors and then check whether application is correcting his errors. This metric is defined as X=A/B, where:
    - A is number of screens or forms where the input data were successfully modified or changed before being elaborated
    - B is number of screens or forms where user tried to modify or change the input data during observed user operating time

  We executed this metric also on grammatical errors. So from the three test cases (wrong use of capital letter, misspelled words and passive voice), we received the following numbers A=2, B=3 and X=0,667
- Self-explanatory error messages [ISO/IEC 9126-2]. Evaluator should verify whether error messages propose appropriate recovery action. This metric requires user test for a longer period and analysis whether user takes a right action on base of the error message. Therefore, we did not evaluate the product with this metric.
- Existence of "Undo" operation/feature. Evaluator should verify whether "Undo" function exists and it is functioning properly. We can define this metric as Boolean, which verifies whether application contains an undo operation. In case of MS Word the value will be X=1.

Attractiveness attributes of Office application:

- Attractiveness of the interface

Attractiveness metrics of Office application:

- Attractive interaction [ISO/IEC 9126-2]. The standard [ISO/IEC 9126-2] recommends assessing attractiveness with questionnaire to users, how they experience the product and the user interface. In absence of proof users, evaluator should give a subjective grade.
- Interface appearance customizability [ISO/IEC 9126-2]. Evaluator should count interface parameters that can be customized and total number of interface parameters. We will redefine this metric because [ISO/IEC 9126-2] definition is related to user wishes. Our definition will be X= A/B, where:
    - A is number of user interface elements available for customization
    - B total number of user interface elements available

In case of MS Word, we have toolbar menu:

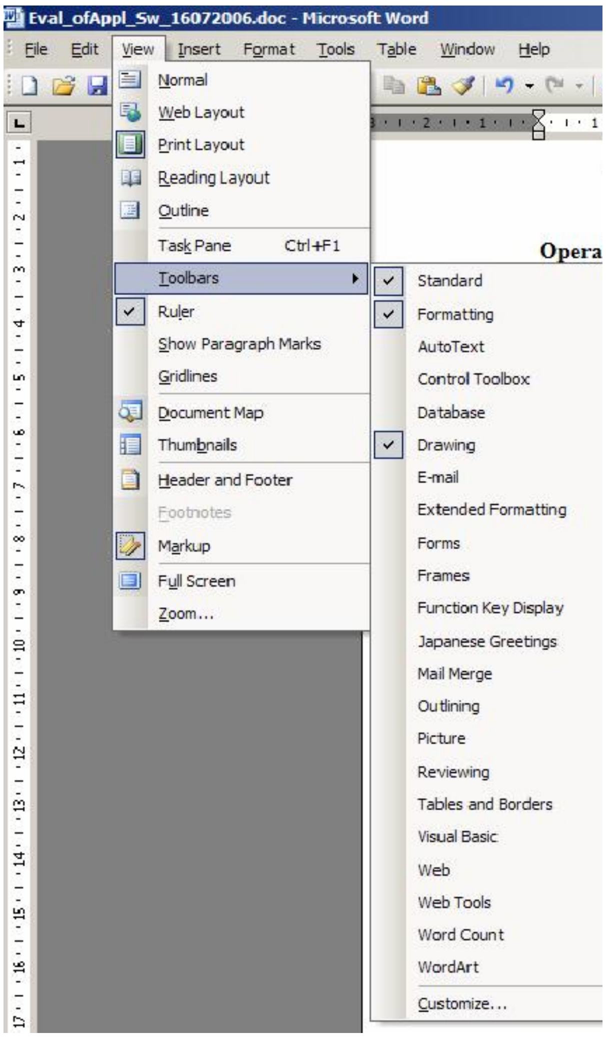


Figure 27 Toolbar menu of MS Word

The menu has an option **Customize** that can customize the commands

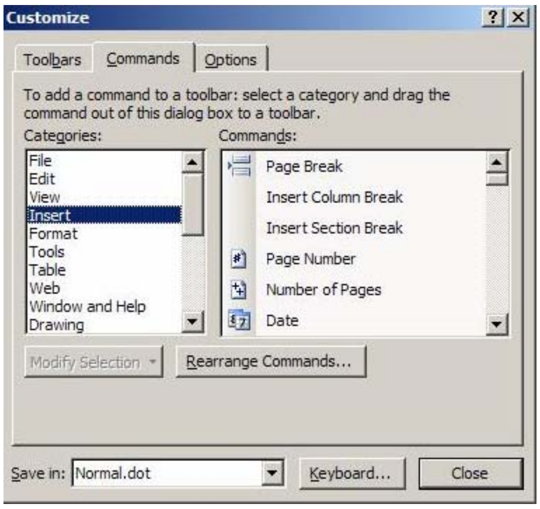


Figure 28 Customize option

Where all commands can be customized. Therefore, we will have the following numbers: A=+-700 B=+-700 X=1

Efficiency attributes and metrics

Time behaviour attributes:

- Time needed to execute system functions/operations

Time behaviour metrics:

- Response time to execute operations derived from [ISO/IEC 9126-2]
    - Time needed to start application. Start-up time is relative short on a today's standard hardware configuration (system with 1GB of RAM and CPU frequency of 3GHz) and it is Tsa=1s
    - Time needed to close application. Close application time is relative short on a today's standard hardware configuration (system with 1GB of RAM and CPU frequency of 3GHz) and it is Tca=1s
    - Time needed to open a document. This metric depends of the size of the document and same configuration (system with 1GB of RAM and CPU frequency of 3GHz). We received the following results:
        - Document with size of 1,5 MB time to open Tod=1s

Resource utilization attributes:

- System utilization
- System Hardware Requirements

Resource utilization metrics:

- Memory usage. Amount of memory used by the application. Evaluator should check the amount of memory used by the application processes. In a normal condition without any files open, MS Word 2003 uses 16MB of system memory. When we opened a file of 500KB memory usage of MS Word was 26MB. When a file of 1,5 MB was open, memory usage went to 37MB.
- Hardware requirements. Evaluator should check help or product documentation about hardware requirements. For Word 2003 we found the following hardware requirements

- Pentium CPU with minimal frequency of 233 MHz
- Minimum 128 MB of memory plus 8 MB for word application

Portability attributes and metrics for Office application

Installability attributes of Office application:

- Installation of the application

Installability metrics for Office application:

- Ease of manual installation [ISO/IEC 9126-2]. Evaluator should execute manual installation and give a grade based on complexity of the installation. This metric is defined with for levels (*very easy, easy, not easy* and *complicated)*, in the case of MS Word we can grade it with *easy*, that is 3rd of 4 levels, equivalent to 0,75 of 1
- Ease of installation of the Word updates. Evaluator should execute manual installation of Word updates and give a grade based on complexity of the installation. We can define this metric with same four levels: *very easy, easy, not easy* and *complicated.* Grade for the ease of installation of updates will be *easy* or 0,75 of 1

Replaceability attributes for Office application:

- Upgrade of the office application
- Support of outputs from previous/new versions

Replaceability metrics for Office application:

- Upgradeability to new product software version. Evaluator should check the product documentation of the newer version to check which older versions can be upgraded to the newer version. We can define this metric as Boolean. Upgradeability of MS Word is defined by business strategy of Microsoft. In case of MS Word 2003, the product can be upgraded to the only later version available on the market now MS Word 2007. Microsoft charges about 130 Euros in the Netherland for this upgrade. This upgrade is not available for older MS Word version like MS Word 97 and MS Word 6.0. Thus, from technical point of view the grade for our metric will be 1, if we take in calculation the financial cost than we can give a lower grade of 0,5.
- Support for files created with previous MS Word versions, derived from continued use of data [ISO/IEC 9126-2]. Evaluator should check whether file could be saved in a format of older application versions. We can define this metric as Boolean. MS Word 2003 supports opening and saving documents (files) created with previous MS Word versions. Grade for this metric will be 1.

Entertainment application Minesweeper

Functionality attributes and metrics of entertainment application

Suitability Attributes for entertainment application:

- Random generation of mines on squares within the game defined field of squares
- Measuring the game duration time. Evaluator should verify that application measures and displays the playing time properly.
- Supplying the square information of the selected square i.e. field is mine of number of the mines on the field bordering fields

Suitability metrics of entertainment application:

- Adequacy of major product functions, metric derived from functional adequacy [ISO/IEC 9126-2]. Evaluator should verify if major product functions are functioning properly.
    - Random generation of mines on squares within the game defined field of squares
    - Measuring the game duration time
    - Supplying the square information of the selected square i.e. field is mine of field is presenting number of the mines on the eight bordering fields

  Metric is defined as X=1-A/B where:
    - A is number of attributes in which problems are detected during the evaluation
    - B number of evaluated attributes

  We evaluated the following functions for Minesweeper and we received the following numbers A=0, B=3 X=1.
- Completeness of the listed major functions, metric derived from functional implementation completeness [ISO/IEC 9126-2]. In the absence of requirements specification, we cannot assess this metric.
- Providing different complexity levels. Boolean metric verifying whether the application contains different levels. Evaluator should verify that application offers different levels of complexity; he should start the application in any of these levels if check, that application runs in these levels and that complexity is different. Minesweeper has the following levels: "Beginner", "Intermediate", "Expert" and "Custom" (where the user can define the complexity of the gaming application). In the case of minesweeper grade for this metric will be 1.
- Storing/displaying the best scores. Evaluator should verify that application shows the best scores. Boolean metric checking whether the application stores and displays the best scores. We reset the scores and played one complete game, after finishing the game we have right to sign in the best scores.

  In the case of minesweeper grade for this metric will be 1.

  Note: Application does not display and stores best scores of the Custom level, but this is expected behavior because application in the custom levels does not have strict level of complexity.

Interoperability attributes of entertainment application:

- Support for network entertainment

Interoperability metrics of entertainment application:
These attributes are not applicable for Minesweeper but are applicable for the modern gaming applications.

- Supported LAN (Local Area Network) gaming. Evaluator should verify that application could be played in the LAN environment, with multiple users playing the same game. Boolean metric checking whether the game can be played in a LAN. In case of Minesweeper application the grade for this metric will be X=0.
- Supported Internet gaming. Supported Internet gaming. Evaluator should verify that application could be played on Internet. Boolean metric checking whether the game can be played on Internet. In case of Minesweeper application the grade for this metric will be X=0.

Usability attributes and metrics of entertainment application

Understandability attributes of entertainment application:

- Self-descriptiveness of the user interface
- Product demonstration

Understandability metrics of entertainment application:

- Existence of demonstrations. Evaluator should check whether demonstration tutorials exist as part of the product software or on Internet. This can be defined as Boolean metric, checking whether application contains demonstrations. In case of Minesweeper application the grade for this metric will be X=0.
- Completeness of description [ISO/IEC 9126-2]. Evaluator should try to understand functions described in the product manual and evaluate if description is correct. Metric is defined by [ISO/IEC 9126-2] in a following way X=A/B where:
  A is number of functions understood
  B total number of functions
  We evaluated help of the application and we have the following numbers A=6, B=6 and X=1
- Evident GUI/toolbar functions derived from [ISO/IEC 9126-2] “Evident functions”. Evaluator should check if toolbar functions are clear to him on base of the toolbar icons. The metric is defined as X=A/B where:
  A is number of GUI/Toolbar functions identified by user
  B total number of GUI/Toolbar functions
  In case of Minesweeper, we received the following numbers A=0 B=1 and X=0 the only GUI/Toolbar:

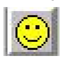


Figure 29 Minesweeper toolbar GUI

That means start new application cannot be identified in that way from our sample user.

This low figure is because of the fact that minesweeper is simple application, so the supplier did not spent much effort in preparing demonstration and evident GUI functions

Learnability attributes of entertainment application:

- User/product manual
- Application help

Learnability metrics of entertainment application:

- Existence of user manual. Evaluator should check whether user manual exist as part of the product software or on Internet. In the case of Minesweeper user manual is provided as part of application package. Thus grade for this metric is X=1.
- Ease of function learning. Evaluator should evaluate how easy is learning functions for a test user. In absence of a test user, evaluator should execute the test user role. The metric is defined by [ISO/IEC 9126-2] is the mean time to learn the function. We decided to measure giving a subjective grade in the range from 0 to 1 where, value closer to 1 means that the functions can be learned very easy and value closer to 0 means that the functions are difficult to learn. In case of Minesweeper we give a grade X=0,9.
- Correctness of user documentation, metric derived form "effectiveness of the user documentation" [ISO/IEC 9126-2]. Evaluator should try to execute functions described in the product manual and evaluate if description is correct. The metric is defined as X=A/B where:
    - A is number successfully completed tasks after accessing online help/or documentation
    - B total number of tasks tested

  In the case of Minesweeper we received the following numbers A=4, B=4 X=1
- Help accessibility on application screens, derived from "help accessibility" [ISO/IEC 9126-2:2001]. Evaluator should check how many of the available screens have correct online help. Metric is defined as X=A/B where:
    - A is number of tasks for which correct online help is located
    - B is total number of tasks tested

The metric is not applicable for minesweeper because the application contains one screen only, with only one online help screen not related to the tasks. However, we find this metric relevant for the complex gaming applications that contain many different screens.

Attractiveness attributes of entertainment application:

- Appearance of the software
- Design of the user interface
- Application content attractiveness
- Software newness

Attractiveness metrics of entertainment application:

- Attractive interaction [ISO/IEC 9126-2]. This metric is should be measured by [ISO/IEC 9126-2] using a questionnaire where user can answer how attractive is the interface to them. In absence of proof users, we will try to give a subjective grade in the range from 0 to 1, where 1 means very attractive and 0 means no attractive interaction. Our grade will be X=0,65
- Age of the software. Check when the software was produced; the assumption is that newer software is more attractive. In case of Minesweeper, we found information on one gaming site (http://www.gamesetwatch.com) that it was first time introduced as part of *Microsoft Entertainment Pack for Windows* in 1990. Thus, we can say that this is medium age application from the early PC age. Our grade for this metric on a 0 to 1 scale will be X=0,5
- Interface appearance customization the name of the metric is from [ISO/IEC 9126-2]. Evaluator should count interface parameters that can be customized and total number of interface parameters. We will try to modify it for this application in a following way X=A/B where:
    - A is number of interface parameters that can be customized (size of the fields and number of mines)
    - B is number of interface parameters that the user wishes to customize

  For Minesweeper, we will have the following numbers: A=3, B=4 X=0,75. We expect that the user would like to modify the color of the application, that customization is not provided. Therefore, number B is greater than number A.

Efficiency attributes and metrics of entertainment application

Time behaviour attributes of entertainment application:

- Time behaviour of the system

Time behaviour metrics of entertainment application:

- Response time to execute operations, derived from [ISO/IEC 9126-2]. This metric is not applicable to Minesweeper application because it is quite light application for today’s hardware configurations. However, we consider this metric relevant for other entertainment applications, because the mother gaming applications are CPU and memory consuming.
- Memory usage. We define this metric as amount of memory used by application process. . Evaluator should check the amount of memory used by the application processes. In case of Windows application, we use Task manager to check the amount of memory used by the process. As you can notice on a figure, bellow Minesweeper uses about 2,7 MB of memory.

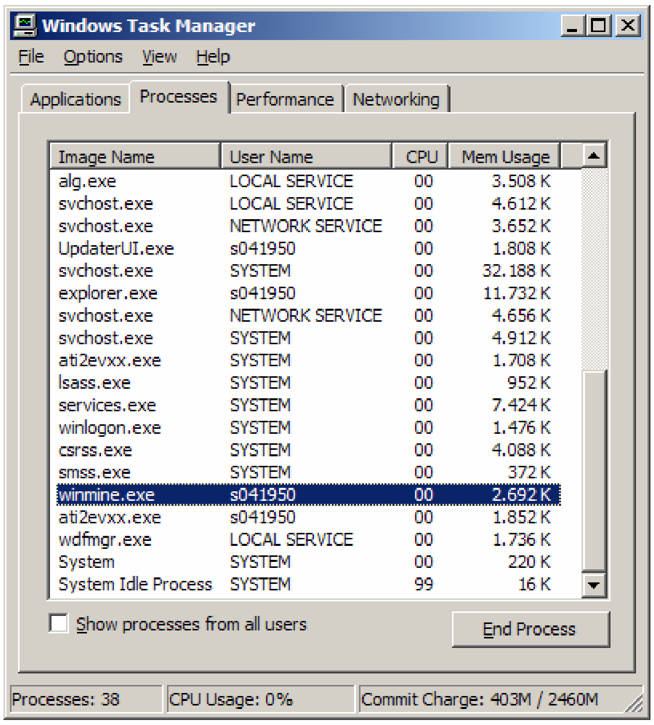


Image Name	User Name	CPU	Mem Usage
alg.exe	LOCAL SERVICE	00	3.508 K
svchost.exe	LOCAL SERVICE	00	4.612 K
svchost.exe	NETWORK SERVICE	00	3.652 K
UpdaterUI.exe	s041950	00	1.808 K
svchost.exe	SYSTEM	00	32.188 K
explorer.exe	s041950	00	11.732 K
svchost.exe	NETWORK SERVICE	00	4.656 K
svchost.exe	SYSTEM	00	4.912 K
ati2evxx.exe	SYSTEM	00	1.708 K
lsass.exe	SYSTEM	00	952 K
services.exe	SYSTEM	00	7.424 K
winlogon.exe	SYSTEM	00	1.476 K
csrss.exe	SYSTEM	00	4.088 K
smss.exe	SYSTEM	00	372 K
winmine.exe	s041950	00	2.692 K
ati2evxx.exe	s041950	00	1.852 K
wdfmgr.exe	LOCAL SERVICE	00	1.736 K
System	SYSTEM	00	220 K
System Idle Process	SYSTEM	99	16 K

☐ Show processes from all users

Processes: 38 | CPU Usage: 0% | Commit Charge: 403M / 2460M

Figure 30 Minesweeper memory usage

Portability metrics of entertainment application

Installability attributes of entertainment application:

- Easy installation of the software

Installability attributes of entertainment application:

- Ease of manual installation [ISO/IEC 9126-2]. Evaluator should execute manual installation and give a grade based on complexity of the installation. This metric is defined with for levels(*very easy, easy, not easy* and *complicated)*, in the case of Minesweeper we can grade it with *easy*, that is 3rd of 4 levels, equivalent to 0,75 of 1
- Ease of manual uninstallation [ISO/IEC 9126-2]. Evaluator should execute manual uninstallation and give a grade based on complexity of the uninstallation. This metric is defined with for levels(*very easy*, *easy*, *not easy* and *complicated)*, in the case of Minesweeper we can grade it with *easy*, that is 3rd of 4 levels, equivalent to 0,75 of 1
- Ease of Setup Retry [ISO/IEC 9126-2].]. Evaluator should execute number of reinstallation (retry set-up) cases and count how many of them were successful. Not applicable for Minesweeper since it is part of Windows operating system and it is not delivered as separate installation package.

Co-existence attributes of entertainment application:

- Application does not conflict with standard applications.

Coexistence metrics of entertainment application:

- Available co-existence [ISO/IEC 9126-2], not applicable for Minesweeper since it is a small application not utilizing system resources and with low hardware requirements. However, we consider this metric relevant for the modern gaming applications.