# Software Product Evaluation

*- Current status and future needs for customers and industry -*

Teade Punter, Rini van Solingen, Jos Trienekens

Keywords: software product evaluation, software product quality, user needs, evaluation of information technology, ISO 9126, ISO 14598

**ABSTRACT**

*Evaluation of Information Technology is mostly considered as decision making on Information Technology investment proposals. However a selected investment proposal still has to be implemented. Actual realisation of such proposals is not as straightforward as it seems. Therefore activities must be started to manage conformance to the initial proposal. One of such activities is 'Software Product Evaluation'. Software Product Evaluation is defined as the assessment of software product characteristics according to specified procedures. During a Software Product Evaluation the fit between the software product and the users' needs of that product are determined. This fit concerns both explicit and implicit needs about the product. This is often referred to as 'software product quality'. Software Product Evaluation becomes more popular both in industry and academics. Several evaluation techniques are already available, and others are being developed. This paper presents the current status of software product evaluation and points out the future needs. We conclude that three main areas can be distinguished on which more progress is necessary: building a quality profile, selecting appropriate actions during software implementation to address the quality profile, and designing and executing the evaluation activities.*

**CONTENTS**



**About the authors:**

- Teade Punter is a researcher at Eindhoven University of Technology. He is working on a PhD assignment for situated evaluation of software products. For KEMA Nederland he is involved in the SPACE-UFO project.
- Rini van Solingen is a Quality Engineer at Schlumberger Retail Petroleum Systems involved in quality measurement and improvement of embedded products. In parallel he is researcher at Eindhoven University working on a PhD Assignment for reliability improvement of embedded products and development processes.
- Jos Trienekens is an associate Professor 'Development of IT products' at Eindhoven University of Technology. He is responsible for a number of practice-oriented IT-research projects at the Frits Philips Institute for Quality Management (FPIQM) and Eindhoven University of Technology. He is as a consultant to KEMA Nederland involved in the SPACE-UFO project.

**Contact:**
Teade Punter
Eindhoven University of Technology, Faculty of Technology Management, section Information and Technology
PAV D-11, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
E-mail: tpu@tm.tue.nl

# Software Product Evaluation

*- Current status and future needs for customers and industry -*


## ABSTRACT

*Evaluation of Information Technology is mostly considered as decision making on Information Technology investment proposals. However a selected investment proposal still has to be implemented. Actual realisation of such proposals is not as straightforward as it seems. Therefore activities must be started to manage conformance to the initial proposal. One of such activities is 'Software Product Evaluation'. Software Product Evaluation is defined as the assessment of software product characteristics according to specified procedures. During a Software Product Evaluation the fit between the software product and the users' needs of that product are determined. This fit concerns both explicit and implicit needs about the product. This is often referred to as 'software product quality'. Software Product Evaluation becomes more popular both in industry and academics. Several evaluation techniques are already available, and others are being developed. This paper presents the current status of software product evaluation and points out the future needs. We conclude that three main areas can be distinguished on which more progress is necessary: building a quality profile, selecting appropriate actions during software implementation to address the quality profile, and designing and executing the evaluation activities.*


## 1. INTRODUCTION

In our society, software is important. It is even impossible to imagine today's society *without* software. Banks, insurance companies, governments and many other organisations fully depend on software. Software creates the possibility to transform large amounts of information, present it in an informative way, and transport it to the other side of the world in a split second. Many companies invest significant resources in software development. Also the amount of products that partially contain software is high. An example is a television set, of which the effort spend to develop a new generation consists for more than 70% of software development resources (Rooijmans et al., 1996).

*Evaluation of Information Technology* is considered as decision making on Information Technology investments. To take the decision to invest is one thing, but actual implementation is another thing. Because of the increasing size, complexity, quality needs and market demands for software, problems arise that are specific for software development. Examples of such problems are planning difficulties, exceeding budgets, unknown or bad product quality, projects that are stopped or never ended, milestones that are reached months or years too late, or developers who are working mostly unstructured, under high stress. Those problems should be addressed by increasing management effort to control implementation[1].

*Software Product Evaluation* can be regarded as an instrument which will support such control. It is defined as the assessment of software product characteristics according to specified procedures (definition is based upon ISO14598, 1996). During a Software Product Evaluation the fit between the software product and the needs of that product are determined. This fit concerns both explicit and implicit needs about the product, often referred to as *software product quality*. By on the one hand examining the needed level of product quality, and on the other hand examining whether a product meets that level of quality, *fitness for use* is evaluated. This can be done during several phases of development and use, which results in increased control during the transformation from investment decision to actual implementation.

[1] *Implementation is regarded as all activities necessary to introduce a software product into an organisation. This includes activities such as development as well as introduction of standard software and product use.*

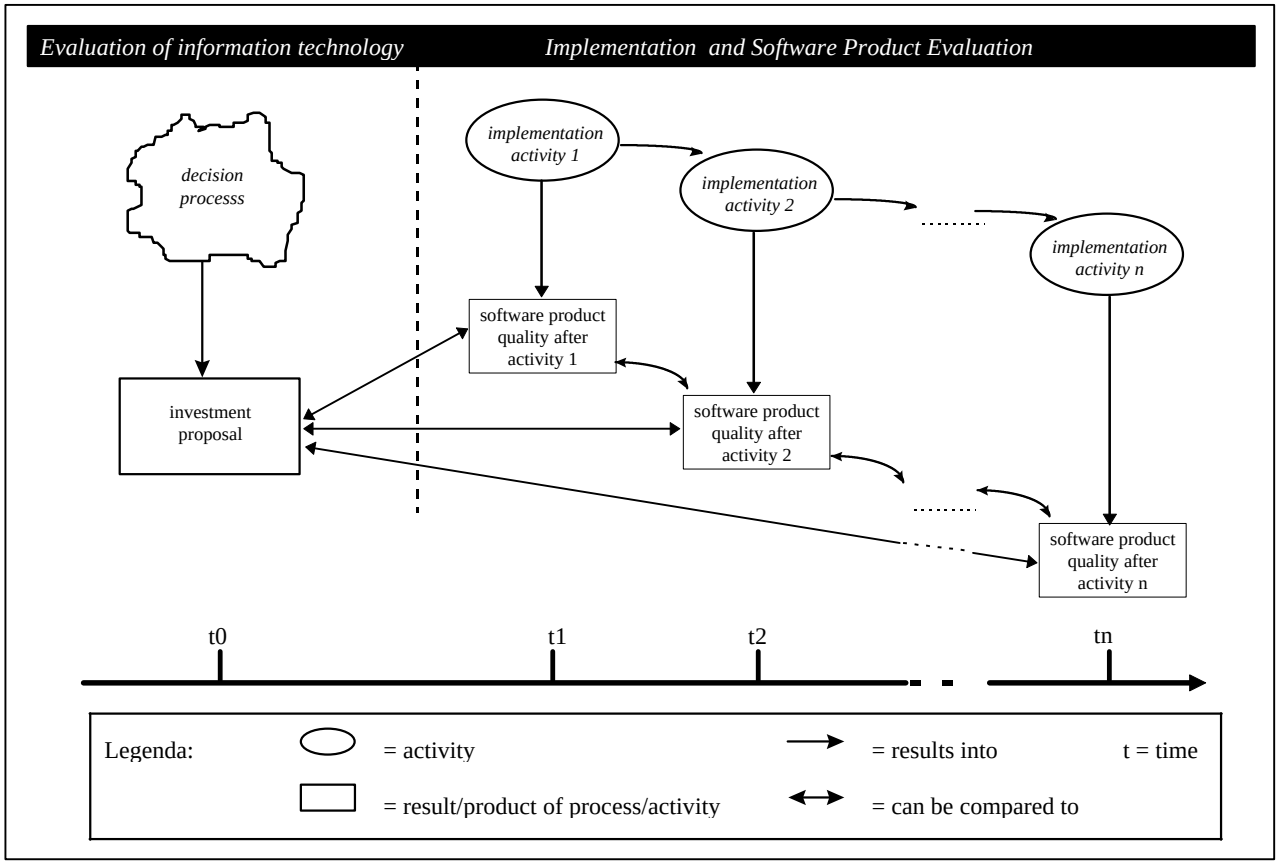


Figure 1: *Evaluation of Information Technology and Software Product Evaluation*

The relation between Evaluation of Information Technology and Software Product Evaluation is visualised in Figure 1. The left side depicts the development of investment proposals during Evaluation of Information Technology. The decision process is not refined and assumed to be a certain process that comes up with a 'best' proposal. On the right side, Software Product Evaluation is depicted as a process during implementation[1]. The result of evaluation is software product quality at different moments. There is a relation between the investment proposal and the actual software product quality. The initial investment proposal is translated into software product quality from the start of implementation. During implementation several versions of the product can be compared to the intended software product quality of the investment proposal.

Software Product Evaluation becomes a growing market within the software industry. Customers and users get the opportunity to have a potential product evaluated to their needs and start demanding such evaluations more and more. On the other hand, certification institutes and evaluation companies push their evaluations into the market and increase revenues because of this new and well received services. And also, producers of these software products are confronted with Software Product Evaluations after their products have been developed. Therefore industry becomes pro-active to such evaluations and change development processes in such a way that the evaluation demands are directly addressed.

Although we have stated that Software Product Evaluation will be an instrument to control implementation of investment proposals this concept cannot be fully elaborated at this moment. For example a procedure for translating an investment proposal into software product quality is not available yet. To elaborate this, research in the area of Software Product Evaluation is required. Therefore this paper focuses upon research in the area of Software Product Evaluation. The current status of this research is described in chapter 2. The needs for improvements and research for Software Product Evaluation are identified in chapter 3.

## 2. CURRENT STATUS

This chapter presents the current status of Software Product Evaluation. Its origin is explained by highlighting important aspects of evaluation in a historical order as visualised in Table 1.

| *Year* | *Developments in Software Product Evaluation* | *Source/Origin* | *Paragraph* |
|---|---|---|---|
| 1991 | user oriented quality model - quality characteristics | ISO 9126 | 2.1 |
| 1994 | evaluation process | Scope-project, elaborated in ISO CD 14598 | 2.2 |
| 1994 | evaluation techniques - bricks / evaluation modules | Scope-project (1994), elaborated in ISO CD 14598-6 | 2.3 |
| 1995 | specifying quality requirements - quality profile | Eindhoven University of Technology / KEMA Nederland | 2.4 |

Table 1: *Overview of development in Software Product Evaluation - overview of chapter 2*

The first aspect of evaluation is expressing the software product as a set of quality characteristics. The ISO 9126 standard founded a user oriented set of quality characteristics in 1991 (2.1). Evaluation is a process which should be reproducible and repeatable. The Scope-project established an evaluation process in 1994, which is elaborated in the ISO CD 14598 standard (1996) (2.2). To execute evaluation requires evaluation techniques. A first investigation and experiences with those techniques were made during the Scope-project (1994). Requirements for the concept of evaluation module -which is a documented evaluation techniques are now defined in a proposal for standardisation (ISO CD 14598 part 6, 1996) (2.3). To start up effective and efficient evaluation requires specification of quality requirements. Eindhoven University of Technology and KEMA Nederland. provided the concept of quality profile in 1995 to determine those specifications. This shall be elaborated during the SPACE-UFO project (2.4).

### 2.1 Expressing software quality

As described before, Software Product Evaluation addresses software product quality. *Quality characteristics* are used as attributes to describe a software product. During the short history of software engineering several quality models -in which the relations between the quality characteristics are determined- are presented. Examples can be found in (McCall et al., 1977), (Boehm, 1981) and (Quint, 1991). In this paper we apply the ISO 9126 quality model, because it is an international standard that addresses user-needs of a product explicitly. ISO 9126 (ISO 9126, 1991) distinguishes six quality characteristics: Functionality, Reliability, Usability, Efficiency, Maintainability, and Portability.

Each of these quality characteristics is split in several sub characteristics. For example quality characteristic ‘maintainability’ is divided into four quality sub characteristics: analysability, changeability, stability and testability. Below an overview of the ISO 9126 standard is presented. Despite criticism about ISO 9126 -see for example: (Mellor, 1992)- the advantage of ISO 9126 is that it provides a common vocabulary to express quality of *user needs*.

***ISO 9126 (1996)*** [2] ***Quality characteristics***

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

- Suitability -the capability of the software to provide an appropriate set of functions for specified tasks and user objectives.
- Accuracy - the capability of the software to provide right or agreed results or effects.
- Interoperability - the capability of the software to interact with one or more specified systems.
- Security - the capability of the software to prevent unintended access and resist deliberate attacks intended to gain unauthorised access to confidential information, or to make unauthorised modifications to information or to the program so as to provide the attacker with some advantage or so as to deny service to legitimate users.

Reliability - the capability of the software to maintain the level of performance of the system when used under specified conditions

- Maturity - the capability of the software to avoid failure as a result of faults in the software.
- Fault tolerance - the capability of the software to maintain a specified level of performance in cases of software faults or of infringement of its specified interface.
- Recoverability - the capability of the software to re-establish its level of performance and recover the data directly affected in the case of a failure.

Usability - the capability of the software to be understood, learned, used and liked by the user, when used under specified conditions.

- Understandability - 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.
- Learnability - the capability of the software product to enable the user to learn its application.
- Operability - the capability of the software product to enable the user to operate and control it.
- Attractiveness - the capability of the software product to be liked by the user.

Efficiency - the capability of the software to provide the required performance, relative to the amount of resources used, under stated conditions.

- Time behaviour - the capability of the software to provide appropriate response and processing times and throughput rates when performing its function, under stated conditions.
- Resource utilisation - the capability of the software to use appropriate resources in an appropriate time when the software performs its function under stated conditions.

Maintainability - the capability of the software to be modified.

- Analysability - 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.
- Changeability - the capability of the software product to enable a specified modification to be implemented.
- Stability - the capability of the software to minimise unexpected effects from modifications of the software.
- Testability - the capability of the software product to enable modified software to be validated.

Portability - the capability of software to be transferred from one environment to another.

- Adaptability -the capability of the software to be modified for different specified environments without applying actions or means other than those provided for this purpose for the software considered.
- Installability - the capability of the software to be installed in a specified environment.
- Co-existence - the capability of the software to co-exist with other independent software in a common environment sharing common resources.
- Replaceability - the capability of the software to be used in place of other specified software in the environment of that software.

## 2.2 Evaluation process

Evaluations of software products must be objective - based upon observation, not opinion. They should also be reproducible. Evaluation of the same product to the same evaluation specification by different evaluators should produce results that can be accepted as identical and repeatable. To do so, procedures for project control and judgement are necessary. On the main level an evaluation process should be defined. During the *Scope-project*[3] such an evaluation process is defined. The process was originally presented in five steps: analysis of evaluation requirements, specification of the evaluation, design and plan the evaluation, perform the evaluation and reporting (Robert, 1994). This process is the basis for the international standard ISO CD 14598, which consists of four steps, as visualised in Table 2. The standard is also different from the Scope-project in distinguishing three perspectives on evaluation: developer, acquirer and evaluator. The headlines of current draft standard are expressed in Table 2.

| | Analysis | Specification | Design | Execution |
|---|---|---|---|---|
| Process for developers 14598-3 | definition of quality requirements and analysis of their feasibility | quantification of quality requirements | planning of evaluation during development | monitoring of quality and control during development |
| Process for Acquirers 14598-4 | establishing purpose and scope of evaluation | defining the external metrics and corresponding measurements to be performed | planning, scheduling and documentation of evaluation | evaluation shall be performed, documented and analysed |
| Process for Evaluators 14598-5 | describing the objectives of the evaluation | defining the scope of the evaluation and the measurements | documenting the procedures to be used by the evaluator | obtaining results from performing actions to measure and verify the software product |

Table 2: *ISO CD 14598 definition of evaluation process. Source: (ISO 14598, part 3-5 (1996)*

These activities contain the do's and don'ts of evaluation. The fact that the standard provides guidance and requirements for three different situations implies that evaluation is not only a technical affair of the evaluator. Developers and acquirers are also involved in evaluation. Therefore they must take the requirements into account as well can get guidance for their part of the evaluation.

## 2.3 Evaluation techniques

Evaluation techniques are necessary to evaluate software products. An *evaluation technique* is a measurement procedure, which also contains the *pass/fail criteria* for the measurement results. For example an evaluation technique

[2] *The new version of ISO 9126: ISO CD 9126 (1997) has added some sub characteristics to ISO 9126 (1991). For example 'attractiveness' which is a sub characteristic of 'Usability'.*

[3] *Software CertificatiOn Programme in Europe (Esprit II, P2151)*

for ‘analysability’ which measures failure analysis time should be explicit if 1 hour or 1 week is acceptable. Those pass/fail criteria are necessary to decide about accepting the product. The pass/fail criteria of an evaluation technique must be applied to a restricted range of software products: only measures for which the criteria can be applied are valid for the product. Therefore it is necessary to specify quality first, before evaluation is conducted.

*Quality characteristics and evaluation level*
Evaluation techniques are related to quality characteristics or quality sub characteristics. The level of evaluation concept was introduced by the Scope-project to correlate the stringency of the application of an evaluation technique to the context of the use of the software product (Robert, 1994). Products with different application risks must not be evaluated with equal stringency. For example a word processor has lower application risk than the security system of a nuclear power plant. a. Therefore the word processor requires less thorough evaluation. Four levels are distinguished by increasing risk: D, C, B and A. Table 3 presents a proposal -taken from the Scope-project- of evaluation techniques for these four levels and quality characteristics (Robert, 1994).

| | Level D | Level C | Level B | Level A |
|---|---|---|---|---|
| Functionality | functional testing | review (checklists) | component testing | formal proof |
| Reliability | programming language facilities | fault tolerance analysis | reliability growth model | formal proof |
| Usability | user interface inspection | conformity to interface standards | laboratory testing | user mental model |
| Efficiency | execution time measurement | benchmark testing | algorithmic complexity | performance profiling analysis |
| Maintainability | inspection of documents (checklists) | static analysis | analysis of development process | traceability evaluation |
| Portability | analysis of installation | conformity to programming rules | environment constraints evaluation | program design evaluation |

Table 3: *Evaluation techniques for various levels and quality characteristics. Source: Robert (1994).*

The concept of evaluation level defines the depth or thoroughness of the evaluation in terms of the evaluation techniques to be applied and evaluation results to be achieved. Different evaluation levels give different levels of confidence in the quality of a software product. It is widely accepted that different software products with different application risks should not all be evaluated equally thorough. Also the notion of Software Integrity Level in ISO 1508 (ISO 1508, 1996) is related to this idea.

*Evaluation modules*
To stress the restricted application of evaluation techniques -because of their limited validity- and to support their selection for suitable software products the concept of evaluation module was introduced[4]. *Evaluation module* is defined as a structured set of instructions and data used for evaluation (ISO CD 14598-6, 1996). It specifies the evaluation techniques for a quality characteristic and identifies the required evidence. Also the format to report the measurement results is presented. This means that an evaluation module contains the evaluation technique plus description for necessary input data (evidence) and resulting output (format for report).

## 2.4 Specifying quality requirements - quality profile

Table 3 shows the Scope-proposal for categorising evaluation techniques according to the dimensions ‘quality characteristics’ and ‘evaluation levels’. The problem for evaluation was however which quality characteristics are most relevant and should be evaluated. This is due to the fact that applying all techniques is not suitable and too expensive. Therefore the quality profile was introduced.

A quality profile presents the relevant quality characteristics and the evaluation levels for the software product. The profile reflects the notion of quality for a certain software product and makes quality tangible for both developers and users. The profile is based upon information about customer/user, business process and the software product itself (Trienekens, 1994) and (Eisinga, Trienekens and Van der Zwan, 1995). This is depicted in figure 2.

[4] *By the Scope-project originally addressed as 'brick'.*

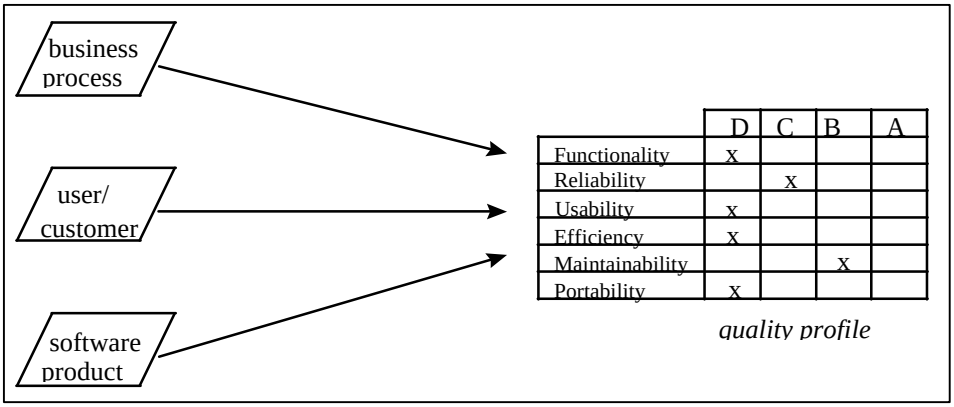


Figure 2: *Determining a quality profile*

The question how to create a quality profiles for a software product, according to deterministic procedures is one of the subject which are elaborated by the SPACE-UFO-project.

***SPACE-UFO-project***

The SPACE-UFO project is part of the SPACE -Software Product Advanced Certification and Evaluation- initiative aiming at the emergence of mature evaluation and certification of IT products. SPACE-UFO -User Focus- is a new CEC ESPRIT project that will provide an enhanced user-oriented method for IT product quality requirements specification, which can be used as the 'Quality Target' for both IT product evaluation processes and IT development processes.

SPACE-UFO will define quality requirement specification methods and techniques to improve development processes, including those loosely defined. The specification methods and techniques will be related to IT product quality measurement and evaluation, which are also useful for organisations that are progressing towards process standardisation and optimisation.. The project will also operational methods, evaluation instruments and training materials to support rapid and effective adoption of the SPACE-method by the IT industry, also in areas of low process maturity. The operational set of methods, evaluation instruments and training facilities will be validated in detailed experiments. A dedicated European user group will soon be established.

A major result of the project will be the SPACE Method for IT product quality specification. The method will be widely published. The entire method will also be implemented in a hypermedia training tool. The method will be supported by a set of specification and evaluation instruments. Existing methods and instruments will be adopted (and possibly adapted to establish user orientation) whenever possible. The method, instruments will be extensively validated on three or more experimental sites, in an industrial environment. Three experimental sites are partner in the project. The lessons learnt from the experiments will be used to tune the methods and instruments, and to identify further opportunities.

The SPACE-UFO Consortium consists of: KEMA Nederland, Brameur QPI United Kingdom, Etnoteam, Italy, IMK (Institute for Small and Medium Enterprises), Netherlands, Italtel, Italy, PSTI-Evaluation, France, Philips, Netherlands, Schlumberger RPS, France, The London City University, United Kingdom. More information can be found on: '*http://www.brameur.co.uk/qpi/space.htm*'

## 3. FUTURE NEEDS, QUESTIONS AND PROBLEMS FOR RESEARCH

The first part of this paper handled the current status regarding software product evaluation. Several standards, international projects, methods and approaches are available that support evaluation of software products. However, current trends show an increasing demand for software product evaluation, while the available methods, techniques and tools are not available or not sufficient. Therefore, this paper addresses the main needs for software product evaluation and points out the most important problems and research questions in this area. This chapter handles those future needs, questions and problems divided over the three main areas of software product evaluation in the SPACE-UFO framework (Figure 3):

- building a *quality profile*, which is based on business needs, user requirements and product issues,
- improving *software implementation* to address this quality profile fully, while being most efficient and effective,
- *design and execution of evaluation activities* that are most efficient and effective, but check a product to the quality profile completely.

These three areas are based upon the conceptual level of the SPACE-UFO reference model for evaluation (SPACE-UFO, 1997). The reference model reflects the notion that software product quality -expressed as a quality profile- should be specified before the product can be evaluated. Therefore the reference model presents two parts of a transformation process. During the first part a quality profile is derived from data about business process, user expectations and software product. The second part uses the quality profile to derive the quality measurement specification and the evaluation plan. The quality measurement specification contains quality specifications which can

be used during the development of the software product. The evaluation plan describes evaluation techniques or evaluation modules and acceptance criteria that will be used to evaluate the product against the quality profile.

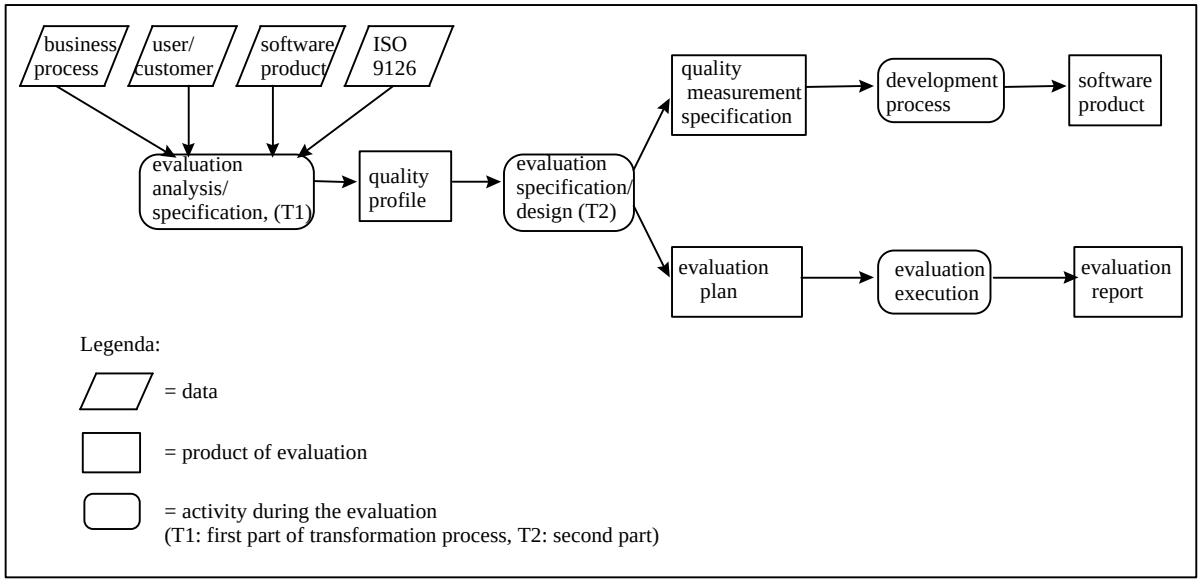


Figure 3: *SPACE-UFO reference model, according to ISO 14598 terminology; based upon (SPACE-UFO, 1997)*

## 3.1 Building a quality profile

Several difficulties exist in creating a quality profile. This paragraph addressed these problems and research directions. When developing a quality profile, the user expectations on product quality are the ultimate requirement to be fulfilled. This means that a quality profile must either:

a. address all requirements as needed by all users involved, or
b. it should make clear that some requirements cannot be fulfilled within the existing constraints.

Building a quality profile exists of two phases: *identification of quality needs* and *specification of quality characteristics*.

1. *Identification of quality needs*. Quality needs of for example users, customers, and management are interrelated with the needs of businesses or companies. Users' quality needs of a software product have to be justified by the impact a software product has on, for example:
   - the business system: the goals that have to be reached,
   - the characteristics of that business system. For example: complexity or flexibility,
   - the tasks of the user. For example: routine-based or ad-hoc.

   To identify quality needs in a structured way, the different business aspects have to be modelled. Based on such a model the meaning and the importance of quality needs can be identified and priorities can be determined. Regarding the modelling of quality characteristics in relation to business characteristics there are still no well-structured specification methods. Therefore no guarantee can be given that really all requirements are included.
2. *Specification of quality characteristics*. Quality characteristics have to be specified in an operational way. This means that for each quality sub characteristic not only a good definition should be given, but that it is also important to connect metrics to these definitions. The aim is that quality characteristics can be interpreted and quantified in a consistent and complete way. There are still no practical theoretical constructs or mechanisms to link quality sub characteristics to metrics. Filling out a quality profile is not supported sufficiently, which makes it difficult to show that really all requirements are included.

Although important steps have been made regarding the structured development of quality profiles, several research questions should still be solved. Two key issues in developing quality profiles are *consensus building* and *learning from previous experiences*.

1. *Consensus building*. During implementation of software different users are involved. Differences between users occur related to for example a persons job, responsibility or role and their specific use of the system. Even people that will not actually 'use' software like the manager or the person who is responsible for buying the software, are involved. Individual preferences also exist based on intellectual skills, background or interests. The result of these

differences is that ‘quality’ is often defined differently among the different users. However, during development there is a clear need for consensus, because decisions on different design alternatives are made continuously. The best way to make this consensus explicit is by specifying a prioritised set of quality characteristics. However, reaching such consensus between all stakeholders is difficult. Possible support might be found in structured group discussions.

2. *Learning from previous experiences*. Evaluation of software product quality is a young profession. Much more experience on developing quality profiles, specifying metrics and the control of the effectiveness of design actions is needed to make this profession more mature. Experiences have to be stored in a structured way to make reuse of previous work possible. Some examples are listed:
    - reusing (standard) quality profiles for evaluating specific types of software packages.
    - reusing metrication results to tune indicators and measurement scales of specific measures.
    - using statistical information about the effectiveness of specific design activities to restructure development processes.

## 3.2 Selecting appropriate actions to improve software implementation

For product development various activities are done. A typical breakdown of activities for developing software is: initiation, specification, design and coding. The development activities create the product and will determine the product quality. Within the development process, several activities can be noted that specifically contribute to the quality of the system. By making these activities explicit, they can be regarded as control actions on all activities within the development process. Depending on both the type of quality required as well as the level required, some actions are selected to result in the required product quality.

To select the right mix of actions from the numerous possible options, and to achieve a closed control loop we need to evaluate actual quality to determine if the proper mix of actions was selected. More in detail, for proper changing software processes that address required quality fully, we need three types of support.

1. assessment of *actual quality*. Product use is the ultimate quality check, but during development specific data collection appears to be applicable to give a representation of product quality. Keeping in mind that control actions will be taken, it seems logical to include data collection activities to track whether those actions give the intended effects.
2. selection of *control actions* during the development processes. To select the appropriate actions to fulfil specific quality requirements, a supporting method is required, which is currently not available.
3. make the *desired quality* explicit while taking into account the multi-dimensional nature of the concept of quality as was discussed in the previous section. This is the quality profile which was described before.

The one thing to be kept in mind is that whenever a product must have high quality, appropriate action must be taken to result in that required level of quality. Selection of actions is a difficult topic, and is done in practice mostly implicit and/or based on experience. Selecting those specific actions (read: customising development processes) is a difficult process, since real effects of actions are rarely known. Therefore, whenever certain actions are taken based on assumptions, then organisations must always evaluate the actual effects.

One research initiative that works in this are is the PROFES project. Main focus of this project is to identify specific changes of the development process that aim directly at increasing product quality. For Software Product Evaluation this implies that the needed software product quality is translated in development process changes by the PROFES method.

***PROFES-project***

The PROFES-project -PROduct Focuses improvement of Embedded Software processes- addresses the market of organisations developing embedded software systems, in various sectors such as telecommunication, medical systems, retailing systems and avionics. The PROFES project addresses the need for an integrated approach supported with operational guidelines and tools to plan and carry out software process improvement to improve final product quality. The PROFES methodology provides support to the integrated use of process assessment, product and process modelling, goal-oriented measurements and experience factory; these approaches are usually applied separately and even perceived as alternatives. PROFES provides support to improve the final product quality by improving the software process; none of the existing approaches proves relationships between process and product quality improvement.

The PROFES methodology handbook will provides guidelines to improve the final product quality in a measurable way by improving and measuring the software process. The PROFES supporting tools provide operational support to the PROFES approach, which make it easy to apply the PROFES approach. Presentation and training material will be provided as a major means to disseminate the PROFES approach and to make it accessible to a large user community.

The PROFES Consortium consists of: Dräger Medical Electronics, The Netherlands, Ericsson, Finland, Etnoteam S.p.A., Italy, FhG IESE (Fraunhofer Institute for Experimental Software Engineering), Germany, Schlumberger Retail Petroleum Systems, France, University of Oulu, Finland, and VTT Electronics, Finland. More information can be found on '*http://www.ele.vtt.fi/profes/*'.

## 3.3 Designing and executing evaluation activities

Evaluation of different kinds of software products should be objective and reproducible -see 2.2. Together with the requirement of customers and evaluators for cost-effective evaluation this results in the need for a method to select evaluation techniques. Selecting appropriate techniques for different kinds of software products is addressed as *situated evaluation*. It is based on the idea that a set of evaluation techniques can be selected from an evaluation base by using the quality profile.

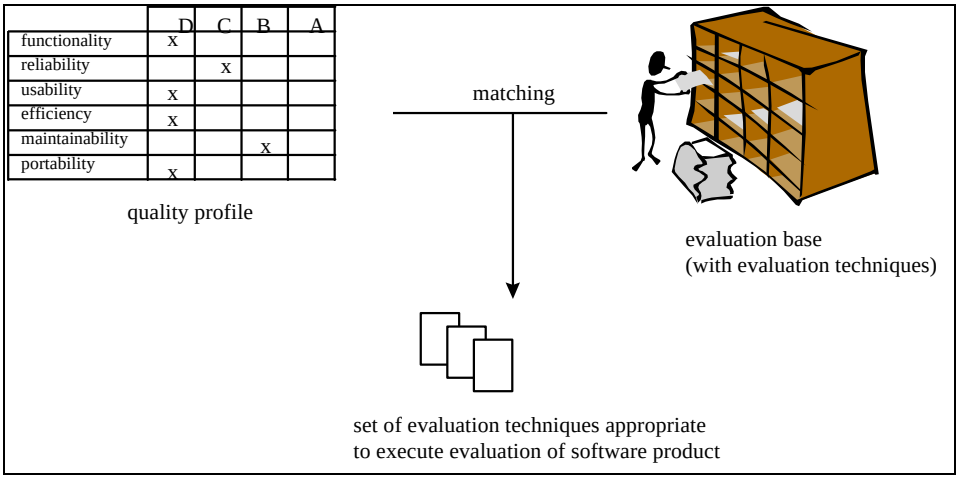


Figure 4: *Situated evaluation*

Situated evaluation addresses several problems in the area of design and execution of evaluation:

1. *Combining evaluation techniques*. To design an evaluation by using techniques from the evaluation base requires consistent techniques for which input and output of evaluation techniques should be defined. The concept of evaluation modules will benefit for this purpose. Also information about experiences with the technique and effort and cost to apply is necessary. To deal with these issues a categorisation of evaluation techniques should be developed which pays attention to technical aspects, like data about input and output for the technique, but also to business aspects, such as: effort, cost and benefits.
2. *Developing and validating evaluation techniques.* Quantitative -based upon metrics- as well as qualitative techniques -like checklists- should be developed (Punter, 1997). Different kind of software products requires other techniques e.g. 3rd generation language products require a *customised evaluation* that should differ from a 4th generation language products. The continuous development of new software products -and the need to evaluate them- requires new techniques. Those techniques should be validated to prove objectivity and reproducibility of their results. Continuous development and validation of evaluation modules for new kinds of software products is required.

## 4. CONCLUSIONS

Software Product Evaluation is an instrument that supports in controlling actual implementation of investment proposals. This is executed by translating the investment proposal into a quality profile, on basis of which the actual software product can be created. Software Product Evaluation provides the procedure to compare different profiles during different implementation moments for the product. A clear description of the translation process from investment proposal into software product quality profile is however needed, but not available yet.

Software Product Evaluation is an emerging area in both academia and industry. It is an area in which both the market grows, and the amount of research questions increases. In this paper we identified the main research areas, problems and questions divided over three aspects:

- building a quality profile,
- improving software implementation by selecting appropriate actions, and
- designing and executing evaluation activities.

The key component of Software Product Evaluation is specifying software product quality in a 'quality profile'. This is based on the starting-point that no evaluation is possible without a proper specification of the needs for the software product. This quality profile is used for two purposes:

a. *designing and executing the evaluation*, because the quality profile points out the quality priorities for the product and therefore describes the quality characteristics that must be evaluated most thorough, and
b. *improving the software development processes*, because specific action must be taken during implementation to make sure the actual software product is created conform the quality profile.

Full support is however not possible yet, but the available methods, techniques and tools are currently identified and expanded in several international research settings. Two European initiatives in this area are the SPACE-UFO project and the PROFES project, which have both been described in this paper.

Software Product Evaluation supports the learning process of Evaluation of IT investment proposals, because it bridges the time between the investment decision and the final implementation. Experiences with proposals in history should be captured and compared to actual implementation. Such an experience base will give support during the evaluation of new IT investment proposals in the future. We believe that Software Product Evaluation is a powerful tool to support this learning process.

## 5. ACKNOWLEDGEMENTS

The authors would like to acknowledge all partners in both the SPACE-UFO and PROFES consortium.

## 6. REFERENCES


- Boehm, B., *Software engineering economics*, Englewood Cliffs NJ, Prentice-Hall, 1981.
- McCall, J.A., Richards, P.K., Walters, G.F., 'Factors in software quality', Vol. I, II, III', In: *US Rome Air Development Center Reports*, 1977.
- Fenton, N.E. and Pfleeger, S.L., *Software Metrics, a rigorous and practical approach*, Thomson Computer Press, 1996.
- ISO 12207, *Software Life-cycles*, 1996.
- ISO 9126, *Software Quality Characteristics*, 1991.
- ISO CD 14598, *Software Product Evaluation*, part 1-6, 1996.
- ISO CD 9126, *Software Quality Characteristics and Metrics*, 1997.
- Juran, J.M., *Quality control handbook*, McGraw-Hill, 1979.
- Kaposi, A., Kitchenham, B., 'The architecture of system quality', In: *Software Engineering Journal*, 1987.
- Kusters, R., Solingen, R. van, Trienekens, J. 'User-perceptions of embedded software quality', In: *Proceedings of the STEP workshop*, IEEE, 1997.
- Mellor, P., *Critique of ISO/IEC 9126*, ISO/IEC JTC1/SC7 Secretary Canada, 1992.
- Punter, T., 'Using checklists to evaluate software', In: *Proceedings of ESCOM conference*, Berlin, 1997.
- Qiu, F., *An expert system approach to modelling and planning software product assessment and certification, Glasgow*, Glasgow Caledonian University, 1995.

- Quint1, *Specifying software quality* (in Dutch), Deventer, Kluwer Bedrijfswetenschappen, 1991.
- Robert, P., *Final report Scope*, Scope consortium, 1994.
- Rooijmans, J., Aerts, H., Genuchten, M. van, 'Software Quality in Consumer Electronic Produkts', In: *IEEE Software*, January 1996.
- SPACE-UFO consortium (Dailey et al), 'The SPACE-UFO project - enabling the IT-industry to specify and demonstrate user-oriented quality', In: *Proceeding of SQM97*, 1997.
- Trienekens, J., *Time for Quality* (in Dutch), Eindhoven University of Technology, Eindhoven, 1994.
- Trienekens, J., Veenendaal, E. van, *Software quality from a business perspective -directions and advanced approaches*, ISBN 90-267-2631-7, Deventer, Kluwer Bedrijfswetenschappen, 1997.

**WWW**

- PROFES www-site: http://www.ele.vtt.fi/ profes/
- SPACE-UFO www-site: http://www.brameur.co.uk/qpi/space.htm