The W-Process for software product evaluation: A method for goal-oriented implementation of the ISO 14598 standard

Software Quality Journal, 12, 137–158, 2004

The W-Process for Software Product Evaluation: A Method for Goal-Oriented Implementation of the ISO 14598 Standard

TEADE PUNTER punter@iese.fraunhofer.de
Fraunhofer IESE, Sauerwiesen 6, 67661 Kaiserslautern, Germany

ROB KUSTERS, JOS TRIENEKENS, THEO BEMELMANS and AARNOUT BROMBACHER
Eindhoven University of Technology, Den Dolech 2, 5600 MB Eindhoven, The Netherlands

Abstract. The importance of software product evaluations will grow with the awareness of the need for better software quality. The process to conduct such evaluations is crucial to get evaluation results that can be applied and meet customers' expectations. This paper reviews a well-known evaluation process: the ISO 14598 standard. The review focuses on the difficulties in selecting and evaluating the appropriate evaluation techniques. The review shows that the standard has problems in applying evaluation processes in practice due to insufficient attention to goal definition and to relationships between activities being implicit. Also, the standard ignores the trade-off between goals and resources and pays insufficient attention to feedback. To address these deficiencies, the W-process is proposed. It extends the standard through an improved process structure and additional guidelines.

Keywords: software product evaluation, ISO 14598, software quality, ISO 9126, ISO 25000, Goal–Question–Metric paradigm

1. Introduction

As the importance of software increases, it is expected that software product evaluations will play an important role in judging software quality. Software product quality is expressed as a set of quality characteristics, like Reliability or Usability (ISO 9126, 1991), or properties, like Fault proneness, Cohesion or Structuredness. Software product evaluation aims at determining to what extent a particular software product fulfills its specified characteristics or properties, or systematically examines the extent to which the software is capable of fulfilling the specified criteria (ISO 8402, 1994)[1]. A variety of approaches for software product evaluation exist: architecture assessment (Kazman et al., 1998), conformity assessment (Pivka and Potocan, 1996), and static code analysis (Mayrand and Coallier, 1996).

Surveys conducted by others (Miller and Dunn, 1997) and ourselves (Punter and Lami, 1998) show that about 50% of the organizations that conduct software product evaluations are dissatisfied with their results, since the evaluation did not result in answers that fit the expectations. Furthermore, organizations are not able to apply the evaluation results for defining effective actions.

The origin of the dissatisfaction with evaluations is often diagnosed as being a measurement problem. The problem is then defined as finding the appropriate software

metrics to predict or determine the quality characteristics or product properties. Based on this perception, much research has been conducted on finding appropriate metrics. One example is the work of (Briand et al., 1999) where the relationships between coupling and cohesion metrics and the fault proneness of object-oriented (OO) classes are explored. Other examples are (Koshgoftaar and Allen, 1998) and (Cartwright and Shepperd, 2000).

We agree with those researchers that evaluation should be measurement based. This is also implied by the term 'systematic examination' stated in the ISO definition on evaluation presented above. However, focusing upon this aspect in isolation is insufficient to improve software product evaluations. It is also necessary to manage the expectations of the stakeholders during goal formulation as well as to involve people during evaluation activities. Therefore, the focus should also include the evaluation process instead of measurement only.

Many descriptions of evaluation processes exist in literature, see, e.g., (Mayrand and Coallier, 1996; Kazman et al., 1998). Often, the primary focus is not the evaluation process as such, but the application of a specific tool or technique, or the description of a working procedure. A publication that focuses on the evaluation process explicitly is the ISO 14598 standard, titled: 'Software engineering—product evaluation.' Because of its explicit reference to the definition of the software product evaluation process, we take this publication as the starting point for our analysis on software product evaluation processes.

This paper starts with a review of the ISO 14598 standard. Problems with practical application of the standard are described in Section 2. After that, Section 3 proposes the improvements for the standard by introducing the so-called W-process. Experiences with the W-process application are reported in Section 4.

2. Review of the ISO 14598 standard

This section introduces the ISO 14598 standard and describes four problems encountered when applying the ISO 14598 standard in evaluation practice.

2.1. Introduction to the standard

The ISO 14598 standard (ISO 14598, 1999) can be regarded as the successor of ISO 9126 (ISO 9126, 1991), which previously defined an evaluation process. Nowadays, both standards complement each other: the ISO 9126 standard provides the vocabulary for defining software product quality, the ISO 14598 presents the evaluation process. That both standards are closely related is illustrated by the current discussions of ISO's joint task committee 1 to harmonize both standards into a new standard, ISO 25000, in which the ISO 14598 parts will be addressed as ISO 2504X (Azuma, 2001).

The ISO 14598 standard consists of six parts. Part 1 presents the general evaluation process, while Part 2 is about planning and management. Parts 3, 4 and 5 of the standard each stress a specific focus, namely the developer's view (14598-3), the acquisition view (14598-4) and the evaluator's view (14598-5) on evaluation, and provide additional guidelines to the phases and activities that are presented in Part 1.

According to the standard, there is an evaluation process with four phases[2] with additional activities within these phases, namely (ISO 14598, 1999):

- Establish evaluation requirements—during this phase, the requirements for the evaluation are transferred into quality characteristics that are conformant to the ISO 9126 quality model. So-called evaluation levels express the importance of the characteristics by expressing the expected use of the product and related risks. Three activities are conducted during this subprocess, namely: establish purpose of evaluation, identify types of products, and specify quality model.
- Specify the evaluation—during the second phase, metrics for the evaluation should be selected. The selection should be based upon the evaluation requirements defined in the previous phase. Other activities during the second phase are establishing the rating levels (which define when measured values will be unacceptable and when they will be acceptable) and establishing criteria for assessment. Criteria could be defined for different quality characteristics or a weighted combination of subcharacteristics is issued to summarize the measurement results.
- Design the evaluation—the third phase concerns documenting the procedures that will be used by the evaluator to perform the measurement. The required resources—e.g., people and techniques—are specified as well as their allocation to the different activities during the evaluation execution phase. The result of the design phase is an *evaluation plan* that describes the evaluation procedure and the schedule of the evaluator's actions.
- Execute the evaluation—during this phase, the actual evaluation is done in a measurement-based way: the actual values for the specified metrics are collected. These values are interpreted by considering the specified target values and the criteria. Finally, an evaluation report is produced. For this phase, the standard also provides guidelines.

Part 6 of the ISO 14598 standard provides a format to document ready-made evaluation techniques, which is called evaluation module (EM). The idea behind evaluation modules is to learn from evaluation experiences and avoid developing techniques from scratch each time by being able to select the appropriate techniques from an evaluation library. Such a library should support the selection of appropriate evaluation modules for the specified context (Punter, van Solingen, and Trienekens, 1997).

Because of its specific attention to software product evaluation, the rest of this paper takes the ISO 14598-5 into account. The ISO 14598-5 was applied in several research and industry projects, e.g., (Space-Ufo Consortium, 1998; Starlander and Popescu-Belis, 2002; Colombo and Guerra, 2002).

2.2. Problems with the standard

Although the ISO 14598-5 is an improvement compared to the situation where there is no definition at all, the standard lacks support that would make its application in evaluation practice more successful. This section describes four problems that are not addressed thoroughly by the standard, namely:

1. Insufficient goal formulation;

2. Implicit relationships between activities;
3. No attention to the trade-off between goals and resources;
4. Insufficient attention to feedback.

Insufficient goal formulation. The ISO 14598 starts with the activity 'Establish purpose of the evaluation.' When conducting an evaluation, there will be always the challenge to define goals in a way that is sufficiently abstract to cover their intention as well as define them concretely so that they can be operationalized. However, the ISO 14598 standard does not give much support on how to define evaluation goals in this way.

Another omission is that the standard does not provide support on how to involve stakeholders in goal formulation. Often, several stakeholders are involved, such as project managers, engineers, and customers—each with their own interest in the outcome of the evaluation. Problems occur when those perspectives are in conflict with each other. As an example, we note that during one of our evaluations we found that 9 stakeholders were involved. However, only one had formulated the evaluation goal. This has caused problems in this evaluation getting accepted, because the other parties did not find their goals reflected in the evaluation afterwards. The different interests should be heard to get acceptance for the evaluation, but how to integrate them?

The third shortfall of ISO 14598 is that the standard does not address how to cope with changing goals. Goal formulations that once were mentioned as important might have less weight after a while. This is especially an issue for evaluations that last for longer periods, about a year or longer. It does not concern most product evaluations that are conducted within 1 or 2 months. Therefore, we do not focus much on changing goals during the rest of our paper.

Implicit relationships between activities. The ISO 14598 standard provides a clear overview of the activities that have to be conducted as well as what should be done in those activities. However, the standard does not specify clear relations between the activities. Figure 1 clearly expresses that the activities relate to one of the four phases, but what about the relationships between the activities?

The lack of these relationships cause the activities to be conducted in isolation without taking the output of previous activities into account, while uncertainty exists about whether activity outputs will be applied by successor activities. Process management based on evaluation goal(s) is not possible, because the relations between goal and activities are not specified explicitly.

This might, for example, lead to evaluations in which metrics are selected (during the activity Select metrics) based upon the wrong argumentation, due to insufficient information about the quality properties (defined during the activity Specify quality model).

No attention to the trade-off between goals and resources. Resources are needed to perform one or more evaluation activities. People—e.g., an assessor—and techniques—e.g., a checklist technique—are the main categories of evaluation process resources. Appropriate resources should be chosen to achieve the evaluation goals. A mechanism that could be used is called *trade-off between goals and resources*. For

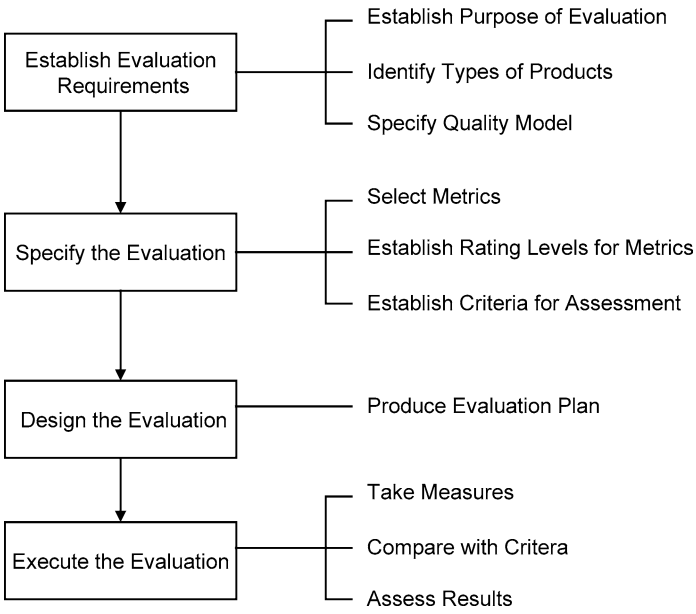

Figure 1. Overview of the ISO 14598 evaluation process.

example, if decreasing the effort for the evaluation is a major ambition of the evaluation, and if a checklist and a metric are both appropriate to execute the evaluation, but the checklist requires less effort, the checklist will likely be selected.

In case there are no suitable techniques at all, the expectations should be tempered and goals should be reformulated. Trade-off should be done continuously during an evaluation process. When the evaluation goal formulations are available and the supporting resources can be denoted, the trade-off should start. But when the trade-off is set, the disturbances in the evaluation process, e.g., changing evaluation goals or absence of evaluators, might also lead to mismatches, which require setting a new trade-off.

The ISO 14598 standard addresses the concept of resources with the concept of evaluation modules. However, the standard does not handle trade-off as such, because it is often impossible to trace the evaluation goals to the evaluation modules. For example, knowing that an evaluation module is about determining maintainability of the software is not enough to match it to a goal formulation that expresses Maintainability as a focus. An explicit relationship with goals is necessary also to address the issues of cost, time, and effort of the evaluation module.

Insufficient attention to feedback. Conducting evaluations successfully also requires monitoring of its processes. Feedback is necessary to find out if the process activities result in the specified outputs so that it is likely that the evaluation goal will be achieved. If deviations are discovered, then steering actions are required to achieve the desired result(s) after all.

Different events might disturb an evaluation process. An extraordinary example during a code metric evaluation we conducted was the replacement of a development

platform that had caused problems in the parsing activity. Therefore, the parser became useless and it was impossible to continue this evaluation. The feedback action that was needed to take here was selecting and installing a new parser. More ordinary disturbances during the evaluation process are unavailability of metrics to conduct code measurement and assessors that are scheduled to conduct the evaluation but are not able to do so anymore (e.g., due to sickness).

The ISO 14598 does not explicitly address monitoring and feedback in its definition of the evaluation process. Nor have we seen such feedback in other evaluation literature so far. This causes evaluations whose results are only evaluated at the end of the process. In case the customer's expectations regarding the evaluation are not fulfilled, it is then too late to steer the process in the right direction.

The review of ISO 14598 has detected four problems with this standard. We denote these combined problems as a lack of goal-orientation in the ISO 14598 standard. Our analysis shows the need for a new process structure, the W-process, and additional concepts to address the problems and conduct software product evaluation more effectively.

3. Extending ISO 14598 with the W-process

This section describes the W-process, which is a method to conduct evaluation processes in a goal-oriented way.

The W-process starts with a restructuring of the ISO 14598 process in order to address the problems of insufficient implicit relationships between activities and insufficient explicit trade-offs between goals and resources. The restructuring is done on three levels, namely: the Goal, Question and Metric levels. The GQM principle (Basili and Weiss, 1984) was used for this. This principle states that measurement goals should be formulated as questions, which need to be answered to fulfill the goals. In turn, each question has to be refined into a set of associated metrics. With the GQM principle we recognize that several trade-offs exist in a measurement process. First, goals are operationalized by questions whose answers should enable us to achieve the specified goal. Second, questions are operationalized by metrics whose obtained measurement values should provide data to answer the questions.

These two trade-offs between the Goal, Question and Metric level are applied to restructure the ISO 14598 evaluation process. The result of this restructuring is presented in Figure 2. Because of its shape, the process is called *W-process*. The W-process consists of two V's that are in fact two different and complementary movements: namely, the definition-V and the execution-V. The *definition-V* starts with the activity "Determine evaluation goal" and ends with the activity "Produce evaluation plan." The *execution-V* starts with the activity "Produce evaluation plan" and ends with "Determine results." Both movements start at the goal level and go down to the metric level. Finally, both return to the goal level.

One of our assumptions in restructuring the ISO 14598 process is to preserve the existing ISO activities as much as possible. Activities are only changed when required by our review. The activities are described below.

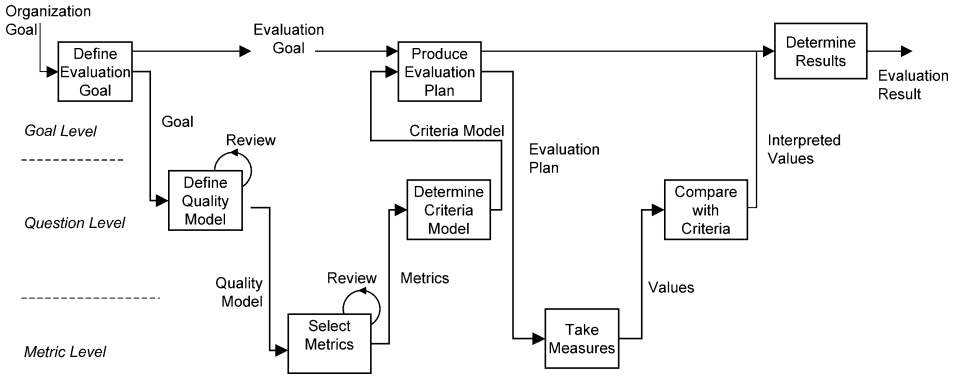

Figure 2. The W-process.

3.1. Define evaluation goal

Mainly, this activity has the same purpose as the activity 'Establish purpose of evaluation' defined by ISO 14598; however, to address the problem of insufficient goal definition, additional guidelines are formulated, namely:

- Identify business goals.
- Identify and involve stakeholders.
- Define the evaluation goal according to a measurement goal template.
- Prioritize evaluation goals.

Identify business goals. Evaluation goals should be derived from business goals (the goals that define what the organization wants to achieve) to assure that the evaluation makes sense: that its results will be used by the organization. Defining the business goals is dedicated to the activity 'Define evaluation goals' only. However, in case those business goals are not explicit at the start of the evaluation, they have to be defined during this activity. Some guidelines for refining business goals into subgoals are provided by (Park et al., 1996), e.g., the identification of business goals is best done in team settings (with managers participating) and the prioritisation of business goals is best done by using techniques such as structured brainstorming.

Identify and involve stakeholders. A stakeholder is any individual, team or organization with interest in or concerns regarding an evaluation (IEEE 1471, 2000). The ISO 14598 addresses the variety of stakeholders during an evaluation by distinguishing different perspectives on the evaluation process, namely developer's, acquirer's and evaluator's perspective; see Section 2. For goal formulation, it is important to assure that all stakeholders of an evaluation are consulted; one of them is the customer (the one who pays) of the evaluation. Each of the stakeholders will have their own perspective and interest in the evaluation that might result in different and perhaps conflicting evaluation goals, but that might also enrich goal formulation. Therefore, it is important first to make diversity in goals explicit and then to define a goal on which as many stakeholders as possible agree. For example, based on our experiences, we think that the following stakeholders are important for software product evaluations:

- Software supplier—is concerned about the evaluation result: what is the opinion of the evaluator regarding our product? A software supplier might order the evaluation himself, e.g., to distinguish his product in the market. The supplier might also be committed to conducting the evaluation. These opposite motivations will influence the supplier's involvement in the evaluation.
- Software customer—is interested in the evaluation result, more specifically, the quality of the software. The software customer might be part of the evaluation, when the customer is asked for his/her opinion about the product.
- Evaluator—is concerned about the technical issues of the evaluation (e.g., which evaluation techniques have to be selected) and the requirements of the evaluation customer regarding the evaluation.
- Customer of the evaluation—is concerned about the cost of the evaluation and the applicability of the information that will result from the evaluation, e.g., can the evaluation results be used to take (improvement) actions.

These stakeholders should be involved in goal formulation. At least the customer of the evaluation should express his/her perspective, but also the other stakeholders have to be consulted as much as possible. Asking the other stakeholders means that their perspectives should be taken into account seriously. It does not mean that their view will be included in the evaluation goals. In fact, this is a matter of negotiation.

Define the evaluation goal according to a measurement goal template. This guideline is to assure that goal formulation will be as explicit as possible. The measurement goal template is a template proposed by the Goal–Question–Metric approach (Solingen and Berghout, 1999) that describes a goal by addressing five elements, namely: object, purpose, quality focus, viewpoint, and context. Using this template will facilitate the definition of evaluation goals.

Prioritize evaluation goals. Depending on the amount of evaluation goals that are defined as well as on the information about the available time and effort that can be spent on evaluation, the goals should be prioritized. A structured brainstorming with the stakeholders who were consulted will be one approach to do so.

3.2. Define quality model

This activity conducts the definition of the product properties (quality attributes, characteristics) that are to be fulfilled by the software product. The activity should provide the same output that is defined by its equivalent in the ISO 14598 process, namely a quality model that breaks software quality down into attributes. The well-known phrase 'quality is in the eyes of its beholder' has an impact on how a quality model should be defined. At least the different perspectives relevant for the product—the stakeholders—should be involved. Based on this, we basically distinguish two approaches to defining a quality model, namely by quality profile or by abstraction sheet.

A *quality profile* presents the quality model as a set of ISO 9126 quality (sub-)characteristics and their degree of importance: the so-called evaluation level, which applies to one of the following four categories: low (D), medium (C), high (B), and excel-

lent (A), to assure safety or to assure approval (Rae et al., 1995). With the Space-Ufo method, a quality profile for a specific software product can be defined. The method takes three factors into account, namely: business process, user, and environment of the product. Details on the method can be found in (Space-Ufo Consortium, 1998)[3]. The Space-Ufo method is suitable when appropriate evaluation techniques exist that can be matched to the quality characteristics that are defined in the profile. However, this is not always possible.

Often, customization to specific (measurement) requirements is needed. Therefore an alternative approach is proposed, namely defining a quality model according to the format of abstraction sheets. An *abstraction sheet* summarizes the main issues and dependencies of an evaluation goal on four quadrants in a sheet, namely:

- The quality factors: what are the measurable properties of the object as specified in the goal template?
- Baseline hypothesis: what is the current knowledge with respect to the measured properties?
- Variation factors: what will influence the quality factors?
- Impact on baseline hypothesis: how do these variation factors influence the quality model?

Abstraction sheets originate from the GQM method where they are applied to support communication between stakeholders when defining questions and hypotheses. With abstraction we are able to propose quality factors to the stakeholders so that a discussion can start on their relevance for the product under evaluation. The advantage is that the sheets can be reused to define quality models for new evaluations. Defining a quality model with the help of abstraction sheets can be compared to the quality profile. One difference is that in addition to quality attributes, hypotheses and their impact are defined. This measurement knowledge facilitates the transfer to the next activity 'Select metrics.'

3.3. Select metrics

During this activity metrics should be defined for the quality characteristics that were defined in the previous activity. The issue is how to get those metrics. Looking at existing measurement plans and getting ideas for metrics based on similar factors is a first approach. A metrics database, like the one defined in the Squid project (Boegh et al., 1999) and (Punter, 1998) is another possibility. However, less measurement experience is available as well, as we found that it is hard to get metrics from metrics databases that are directly applicable. Often, the metrics have to be tuned to the context, for example, for a particular programming language.

If there are no references at all, the metrics and characteristics have to be defined as a define-your-own-model (Fenton and Pfleeger, 1996), like the GQM method. The factors in the abstraction sheets are then regarded as equivalent to questions. For example, the factor Analyzability is related to questions like: 'What is the changeability of the software?'

Another issue for metrics selection is subjective versus objective measurement. In software measurement, metrics are often regarded as being code metrics only. How-

ever, especially for product evaluations, checklist items can be appropriate metrics too. Of course, they should fulfill basic measurement requirements: reproducibility and validity. The advantage of checklist items is that they can often address a broader scope (concerns: object, quality focus) than code metrics. Therefore, we add the guideline of being aware of combining different kind of metrics during metric selection.

3.4. Determine criteria model

This activity is about determining the criteria for the metrics that are specified in the previous activity. Setting criteria is necessary to be able to interpret the actual values of the metrics during the execution-V in the activity 'Compare with criteria.' This activity covers the ISO 14598 activities 'Establish rating levels' and 'Establish criteria for assessment.'

Criteria can be determined by setting hypotheses for the quality factors, defining indicators (Park et al., 1996), and then starting a measurement pilot to determine whether those criteria make sense for the product or not. Such pilots can often not be afforded in evaluation practice because of restricted cost and time. Therefore criteria are often set implicitly (vague), so that they can be determined during execution. Another possibility is to reuse criteria from previous evaluations. In the Squid project (Boegh et al., 1999), a structure for reusing metrics as well as target values (the criteria) was proposed. However, this proposal has not been applied on a large scale yet and its economical feasibility is unknown. Therefore, a criterion setting is still a matter of trial and error: start a pilot and analyze data seriously.

3.5. Produce evaluation plan

During this activity, the evaluation goal(s), quality model, metrics and criteria model are presented in concert. Besides this information, those data collection or assessment techniques are selected that will be applied during the executing-V. The techniques are also assembled in a sequence so that the evaluation can be executed optimally. The activity will result in a description of evaluation techniques and the schedule of the evaluator actions. ISO 14598 Parts 3–5 provide useful guidelines to take into account during this activity.

The trade-off between goals and resources should be set during this activity. The first step is to apply the information that is specified in the evaluation goals. Evaluation goals that are formulated according to the measurement goal template contain such information by addressing object, purpose, quality focus, and viewpoint of the evaluation. The second step is to store and retrieve the evaluation resources according to a similar format. The ISO 14598-6 provides a format for this purpose. However, additional information is needed to make matching with the evaluation goal possible. The object and quality characteristic should be described in a way similar to the one one in the measurement goal template. Also, information about the effort needed to apply the techniques is necessary for the trade-off. This has resulted in the following format to describe evaluation techniques:

- Object—the object that is addressed by the technique, e.g., code, design, implemented system, documentation (Hausen and Welzel, 1993; Fenton and Pfleeger, 1996).
- Quality characteristic—the properties of the product that are addressed by the technique. This might be done according to ISO 9126 (ISO 9126, 1991), but also alternative schemes might also be applied when useful (Dromey, 1996).
- Evaluation level—defined as the ability of the technique to achieve a correct outcome regarding the property that has to be evaluated. Often, this will be related to the plans that the customer has with the evaluation results: is an intensive and fault-free evaluation required? This concerns the purpose of the evaluation, too. Attributes to characterize evaluation levels are: power of the technique and measurement scale (Kitchenham et al., 1995). We think that separating evaluation levels into 4 categories as proposed in (Rae et al., 1995) is difficult to maintain in evaluation practice. Therefore, two levels are proposed:
  - A: the technique is appropriate for evaluating a characteristic and is independent of the evaluator's opinions.
  - B: the technique is appropriate, but its execution depends largely on the evaluator's opinions.
- Effort—the experiences concerning the time and the required skills of the people in applying the technique. Sub-characteristics to define effort are: time, number of persons involved, skills (capabilities) of the involved people.

The trade-off mechanism in which goals and resources are matched is expressed in Figure 3. The figure shows two main techniques (which as such are a subcategory of evaluation resources), namely: abstraction sheets (that are applicable dur-

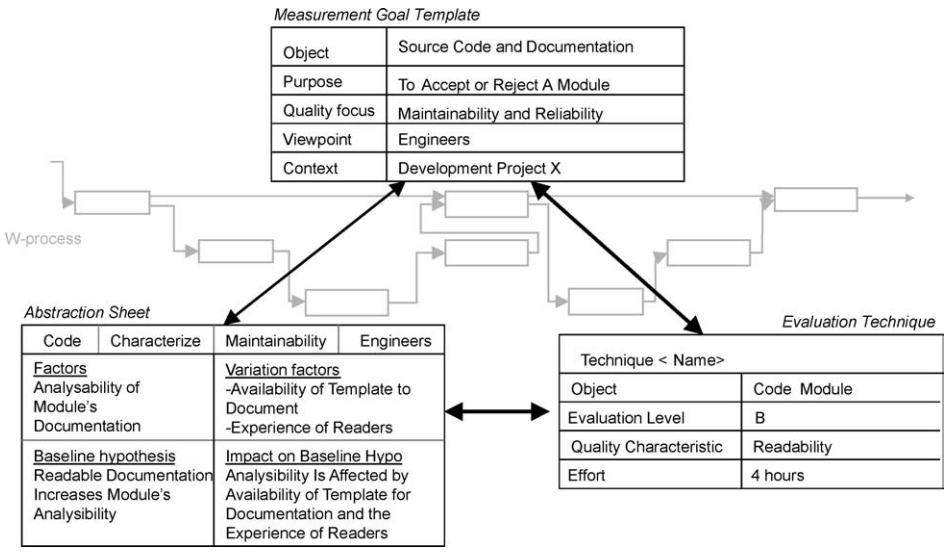

Figure 3. Information about goals and resources necessary for trade-off.

ing the definition-V) and assessment techniques (applied during the execution-V of the W-process).

With the activity 'Produce evaluation plan,' the definition-V of the W-process is finished. At least at this point feedback should be given to the customer. If there is no match between the expectations of the customer and the contents of the evaluation plan, the definition-V should be conducted again, until there is a match.

3.6. Execution-V

The W-process has a special impact on the first phases of the ISO 14598 process: the definition-V. Therefore, its activities were changed compared to the ISO 14598 process. Meanwhile, the activities depicted in the second V in the W-process (the execution-V) are similar to the ones presented by the ISO standard. We do not propose changes for those activities.

The activity 'Take measures' is about data collection. The activity 'Compare to criteria' concerns the interpretation of the data found during the previous activity. Finally, the questions/goals that originally initiated the evaluation are to be answered. This is done during the activity 'Determine results.' The ISO 14598 standard parts 3–5 provide guidelines to conducting these activities successfully. Additional guidelines for data collection and interpretation can be found, e.g., in (Fenton and Pfleeger, 1996).

3.7. Feedback and trade-off moments

Normally our knowledge will grow when conducting evaluations and passing the W-process from left to right. It will be preferable to conduct feedback during several stages of the process as well as focusing upon the trade-off between goals and resources. Figure 4 provides an overview of those moments where it is most likely to give feedback or set a trade-off. This overview is meant to plan feedback and trade-off when conducting an evaluation process.

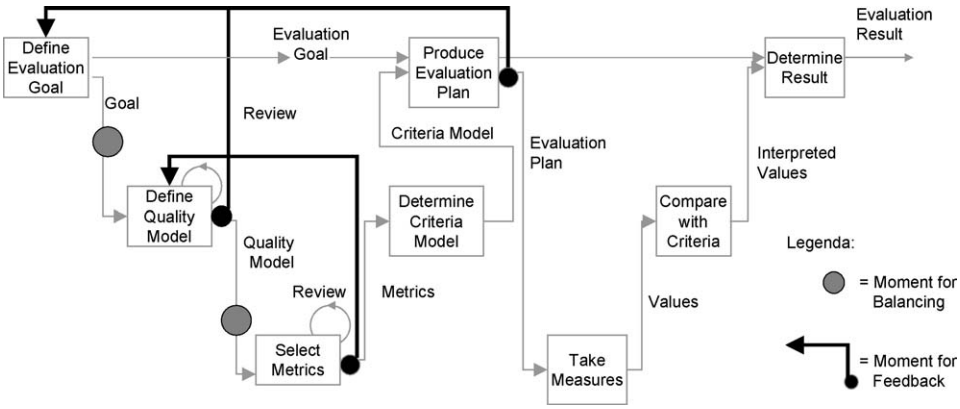

Figure 4. Feedback and balancing moments in the W-process.

4. Applying the W-process

This section presents the experiences with the application of the W-process during three case studies (Punter, 2001). First, the context of the case studies is described. Then, the work done during the evaluation activities is illustrated. Finally, the satisfaction of evaluation participants with the W-process is presented.

4.1. The case studies

Boston evaluation—the first case concerns the development of a source code evaluation. The code modules are developed for the Boston system by Océ Technologies in Venlo, the Netherlands. Boston is an internal code-name of a specific copier. The question that had to be answered during the case study was: 'Is it possible to apply code metrics to predict the quality of software modules?' The software was developed internally. Because of the growing importance of software in copiers, the management at Océ has defined software control and improvement as business goals for the company. To achieve this, static code analysis (Bache and Bazzana, 1994) was defined as one of the actions. The focus of the Boston evaluation was on the definition-V of the W-process. The defined evaluation was conducted several times for experimental purposes.

Omega evaluation—the second case study is about the development of an evaluation for the core system of Omega 2050. Omega 2050 is a retail automation system for gas stations, developed by Tokheim RPS (former Schlumberger RPS) in Bladel, the Netherlands. The core system is produced for the European market. At the time of the start of the case study, a static analysis test procedure was already in place. The evaluation that was to be developed aimed at a measurement-based evaluation of the code modules, especially for determining maintainability of the modules.

IS audit—the third case concerns the improvement of an existing audit. Originally, this audit was developed and applied by Cap Gemini Information Systems Management. Its purpose was to get a well-founded and clear insight into the technical maintainability and manageability of information systems, e.g., a mortgage information system. The case study was done to find new insights, especially criteria, to improve the existing audit.

4.2. Define evaluation goal

The evaluation goal in the Boston case study was defined according to the measurement goal template: "analyze the source code of the modules with the purpose of determining the appropriateness of code metrics with a quality focus on maintainability and reliability from the viewpoint of engineers in the context of a specific copier project at Océ." The goals in the other cases were defined in a similar way; see Table 1.

During two evaluations, the customer of the evaluation formulated the goals. For Boston, it was the project management of the research group static analysis and for Omega, the Quality assurance (QA) manager. Note that the engineers also participated in goal formulation during those evaluations. During the IS audit, the evaluator

Table 1. Evaluation goals of the three evaluations

	Boston evaluation	Omega evaluation	IS audit
Object	Source code	Source code	Information system
Purpose	Determine applicability of code metrics	Accept or reject code modules	Determine the changeability of the system
Quality focus	Maintainability, reliability	Maintainability	Maintainability
Viewpoint	Engineers, research project "Static Analysis"	Engineers, quality assurance	Customer of the audit
Context	Boston	Omega	All kinds of (large) information systems

stakeholder, namely the auditors themselves, formulated the goals. Formulating the goals according to the measurement goal template was found useful: "a simple and good overview."

The goal formulation in these case studies went rather smoothly and easily. This is due to the fact that business goals were clear and not many parties were involved in goal formulation. In case of an evaluation with many more stakeholders, like the one referred to in Section 2, goal formulation would have been more complex by focusing on reducing conflict in goals and increasing communication with the stakeholders.

4.3. Define quality model

The three evaluations started the definition of the quality model by looking at the quality focus as defined in the evaluation goals. Because those foci were clear in all cases, there was no particular need to verify whether the right quality characteristics were chosen and therefore, there was no specific need to define a quality profile. Instead, it was necessary to specify the meaning of the chosen quality characteristics in their specific product context. Mainly, this was about what Maintainability means. For the IS audit, several Maintainability models of large information systems, like (Afotec, 1996), were consulted. The resulting list was compared to the quality factors applied by the original evaluation. When factors in the new list did not cover the original list, they were added to the audit. The quality factors are presented in Table 2.

The quality models for the Omega and Boston evaluation were defined in a different way. The starting point was the idea that, in general, valid quality models for code evaluations cannot be defined. Instead, it was assumed that when engineer opinion matched code metrics, this would result in context dependent quality models. Therefore, the quality models were defined by taking existing models and apply static analysis tools to them (Bache and Bazzana, 1994).

Engineers were asked for their opinion of the factors that determine the Maintainability and Reliability (only for Boston) of the modules in their software. For the Boston evaluation, the approach was extended with a look at historical data on defects that were found in software and effort spent during its development. Both data were regarded as indicators of Reliability and Maintainability, respectively. The engineers' opinion was gathered in a structured way by analyzing a list of potentially relevant factors. This was a similar list as the one used for the IS audit. Factors were selected that

Table 2. Maintainability models (factors, weights, metrics) for the three case studies

	Boston evaluation	Omega evaluation	IS audit
Factors	Analyzability, changeability, stability and testability (of code modules)	Function size, lay-out, nesting, variable naming, code complexity, appropriate comment in code, appropriate documentation, implementation dependency (of code modules)	Completeness, accuracy, analyzability, structuredness (of functional and technical documentation and programs) Age, size, testability and stability (of system) Consistency between system documentation and code
Weights	Weights set for metrics	Weights set for factors, see Table 3	No weights set
Metrics	5 Code metrics (see Section 4.4) and 17 questions to score engineer opinion about four categories	12 Code metrics (see left column in Table 3)	91 questions, see Section 4.4

Table 3. Maintainability model for the Omega case study: relating code metrics to factors; weights of the factors are given in parentheses ( )

	Factors							
Metrics	Function size (0.9)	Nesting (0.9)	Layout (0.8)	Variable naming (0.7)	Code complexity (0.9)	Comment in code (0.9)	Appropriate documentation (0.7)	Implementation dependency (0.6)
Lines of code	+				+			
Statements count[a]	+				+			
Cyclomatic complexity					+			
Nested level count[b]		+	+					
Local variable count				+				
Function call count					+			
Return count[c]			+					
Unconditional jump statement count		+						
Comments per line						+		
Average statement size	+		+					
Vocabulary frequency							+	
Number of hard–software switches								+

[a] Statement count is defined as the number of statements between braces, selection (e.g., if, switch) and iteration statements.
[b] Nested level count is defined as the number of open braces after each other without a closing brace.
[c] Return count is defined as the number of function exits.

were deemed to be important by the engineers. Weights (of importance) were added to the factors. The first row in Table 3 presents the factors with their defined relative weights.

During the three evaluations, the quality models focused on refining the factors for the high-level quality characteristics that were defined in the evaluation goals. This has

resulted in abstraction sheets presenting the quality factors and baseline hypotheses. However, the variation factors and impact on the baseline hypotheses were not made explicit for all. Another experience with the quality modeling was that this activity take up most of the time during the definition-V. The models were defined during several sessions and their construction and review took considerable effort.

Table 2 presents an overview of the factors and metrics that were applied during the three case studies to measure Maintainability.

4.4. Select metrics

The metrics were selected for the factors that were already defined in the quality models. For the IS audit this meant that checklist items were defined. One example is the question: "Have all the functions (described in the technical documentation) been implemented in the system?". For this question, two possible answers were defined, namely: "Yes, functions are described in dataflow diagrams or input and output is described" and "No." Other questions on the checklist might have three or even more possible answers. Values were assigned to the possible answers that will become actual values in case the answer is applicable. Those actual values will each contribute to the total score when filling in the checklist. The items were organized according to the needs for the audit, namely for five topics: functional documentation, technical documentation, source code, system and interface system, and environment. The question quoted before relates to the factor Completeness of Technical Documentation, while another question "Are standard variables applied while programming?" relates to the factor Completeness and Analysability of Program. In total, 91 questions were defined for the IS-audit.

Checklist items were also selected for the Omega and Boston evaluations. This was done to make the scoring of engineer opinions feasible. The questions were selected in a similar way as was done during the IS audit, namely by looking at existing references and discussing their applicability with the stakeholders. But the main focus of the Boston and Omega evaluations was on Code metrics, because of the interest of stakeholders in this metric type. The left column of Table 3 provides an overview of the 12 code metrics that were applied during the Omega case study. During the Boston case study, the following code metrics were applied: Cyclomatic complexity, Nested level count, Nesting, Function call count, Comment density, Program volume. Most metric definitions can be found in (Fenton and Pfleeger, 1996).

In the Boston evaluation, a third type of metrics, namely history data metrics, were also applied. The effort (in hours) already spent on a module was defined as an indicator for a module's maintainability. An overview of the stated hypotheses for the Omega case study is given in Table 3, where the relations between the factors and metrics are depicted by plusses (+).

Reflecting on how the third activity of the W-process was applied in the three evaluations, we conclude that it was, in principle, possible to select metrics by looking at existing references. However, it is not simple to propose a set. Instead, the appropriateness of each metric has to be considered and it should be tuned to the new evaluation as well as making sense for the new measurement.

4.5. Determine criteria model

During the IS audit, the criteria were defined by providing the auditors with guidelines on answering the checklist items by giving additional descriptions. These are phrases in the possible answers, e.g.: the phrase "Functions are described in dataflow diagrams or input and output is described" in the answer choice 'Yes' for the question 'Have all the functions (described in the technical documentation) been implemented in the system?,' see above. It was the auditor's responsibility to take such descriptions into account when determining the answers for a particular software product.

The criteria for the checklists in the Boston and Omega evaluations were set in a similar way. The criteria for the code metrics (target values) of those evaluations were set otherwise, namely by analyzing the code metric measurements with the scores of the engineer opinion about the same code modules. For the Boston evaluation the measurement values were also compared to the defect and effort history. Setting the criteria in this way required actual measurement and interpretation of those data. Therefore, a pilot was conducted, passing through the phases of the execution-V; see below.

4.6. Produce evaluation plan

No standard evaluation techniques were selected and applied during the three evaluations, because they were not available. Therefore, the selection of techniques was not conducted as proposed in Section 3 and illustrated in Figure 3. Instead, this activity has focused particularly on the checklist construction. The checklist items—and related criteria—defined during previous activities were arranged in a sequence so that they can be answered in a feasible way. This meant, e.g., that as few leaps in topic as possible occurred and that control questions were added. The questionnaires were implemented in an Excel spreadsheet. For the Boston and Omega evaluations (about 30 items), this was done because engineers had the opportunity to get direct response when scoring their own modules. The IS audit questionnaire was also implemented as a spreadsheet because this was a large and complex one, consisting of about 91 items. With a spreadsheet, the auditor had a better overview, the possibility of automated calculation, and easier navigation through the questionnaire.

During the evaluation plan definition of the Boston and Omega evaluations, a discussion took place about mixing objective (code) metrics and subjective checklist items. Engineers involved in both evaluations were interested in that because they saw this as a possibility to cover their perceptions better than by a code metrics evaluation only. However, in the end it was decided not to continue with this combination because of the prerequisite to conduct the execution of the evaluation automatically with code metrics only.

4.7. Execution-V

The three evaluations defined during the previous activities were applied one or more times. The Omega and Boston evaluations were applied for 8 and 9 modules, respectively. The IS audit was applied once. During all these evaluations, the activities of

Table 4. Different sequences of activities during three software product evaluations

	Boston evaluation	Omega evaluation	IS audit
1	Define evaluation goal	Select metrics	Define quality model
2	Define quality model	Define quality model	Select metrics
3	Select metrics	Define criteria model	Define criteria model
4	Define criteria model	Take measures/compare with criteria	Produce evaluation plan
5	Produce evaluation plan	Determine results	Take measures/compare with criteria
6	Take measures		Determine results
7	Compare with criteria		
8	Determine results		

the execution-V (Take measures, Compare with criteria, Determine results) had a pilot character. The activities were conducted to set criteria (for Boston and Omega) or to proof the applicability of the evaluation (for IS audit).

4.8. Evaluation strategy

During the case studies, we were confronted with different arrangements in which the W-process activities were applied; see Table 4. It even happened that not all of the defined activities were conducted or that two activities were combined into one (Omega evaluation, IS audit). We found that this practice was due to the experience of the evaluator customer, the skills of the evaluator, the availability of evaluation techniques and the availability of references.

Looking at Table 4, two strategies can be distinguished, namely: a linear and an iterative strategy. The linear strategy was applied in the Boston evaluation. In this evaluation, the activities were conducted in the same order as defined in the W-process. This was because the customer had relative low experience with evaluation and preferred guidance by the evaluator. The evaluator preferred to conduct the evaluation in a top-down way. The iterative strategy can be seen when looking at the order of the activities in the Omega evaluation and the IS audit. In those evaluations, the process was not started with goal formulation, because this was obvious for the stakeholders already. Instead, since they had a reference already, they were more interested in improving what they already had in an immature form: the set of metrics (Omega) and the quality model (the IS audit).

4.9. Satisfaction with the W-process

The satisfaction of the stakeholders with the applied W-process was analyzed for the IS audit and the Boston evaluation. The four problems—identified in Section 2—were the basis for this inventory.

Goal formulation: Participants were satisfied with the goal formulation of the evaluation. Especially using the measurement goal template to define the evaluation goals was found useful. Expressing the quality focus as well as the object in one formulation was found clarifying. The applicants had difficulties in how to operationalize these goals into quality models and metrics.

Process structure evaluation: Participants considered the evaluation processes as being structured. The steps to be taken were clearly defined as well as the planned results. Especially the direct connection between the selected metrics and the quality factors defined on the question level and its connection to the evaluation goal was found supportive.

Balancing goals and resources: Participants of both evaluations were quite satisfied with the chosen metrics and evaluation techniques, and the defined resources. It is noted that for both evaluations, the expectations were originally higher than can be assigned to the final evaluation. Nevertheless, the participants had the feeling that the best possible combination of resources was selected. For example, participants of the IS audit accepted that more stringent criteria for this evaluation audit could not be obtained, without reducing the scope of the evaluation goal. The communication about this during the W-process was found informative and participants accepted the situation.

Feedback: For the IS audit and the Omega audit, it was decided to continue applying the new evaluation. For Boston, it was decided to stop. Participants in the Boston evaluation recognized the usefulness of conducting the pilot and that these kind of evaluations do not provide added value to the organization.

5. Conclusion

This paper has presented the motivation, the contents and the experiences with the W-process. The W-process is a goal-oriented implementation of the ISO 14598 standard for software product evaluation. The W-process is necessary to cope with four problems that the ISO standard faces when applied in evaluation practice, namely: insufficient goal formulation, implicit relationships between activities (which makes managing the process more difficult), no attention to trade-offs between goals and resources (which makes it difficult to define an evaluation with the right expectation for the effort, time and money that can be spent), and insufficient attention to feedback (which makes it hard to steer during the evaluation and to learn from experiences). The aim of the W-process is to support the selection and evaluation of appropriate techniques for an evaluation, e.g., to select the right type of metrics to measure software product quality.

The W-process is an extension of the ISO 14598, taking the standard's activities into account, but expressing the relationships between them more explicitly than the standard does itself. The W-process does this by categorizing the ISO 14598 activities in three levels, namely: Goal, Question, and Metric. Distinguishing those three levels makes the trade-off between goals and resources during several stages in the process much more transparent. Goal formulation has improved mainly by adding the concepts of a measurement goal template and that of stakeholder identification. With explicit information about goals and resources—see the activity 'Produce evaluation plan'—as well as suggesting moments to conduct feedback, the information flows in evaluation processes will improve. This has an impact on managing the activities, the trade-off between goals and resources and the feedback issue.

The W-process was applied in three case studies. In two of those case studies, participants were asked about their satisfaction with the method. Most of them were satisfied with goal formulation, the process structure, and the way the evaluation was conducted. The case studies also resulted in the concept of evaluation strategy, denoting that evaluations can be done in a linear or an iterative way.

Notes

1. An Evaluation can be done during several phases in a software product's life cycle. Assessment—which can be used as synonym for evaluation—is about evaluating an end product.
2. Some of the ISO 14598 documents and its predecessor in the Scope project (Rae et al., 1995) distinguish a fifth activity: Conclusion of the evaluation.
3. Predecessors of this method are described by (Heemstra et al., 1994; Zwan, 1995).

References

Afotec. 1996. *Software Maintainability Evaluation Guide*. Kirtland, USA, Department of the Air Force.

Azuma, M. 2001. SQuaRE: The next generation of the ISO/IEC 9126 and 14598 international standards series on software product quality, *Proceedings of the ESCOM 2001*, April 2001, London, pp. 337–346.

Bache, R. and Bazzana, G. 1994. *Software Metrics for Product Assessment*. London, McGraw-Hill Book Company.

Basili, V. and Weiss, D. 1984. A methodology for collecting valid software engineering data, *IEEE Transactions on Software Engineering* 10: 728–738.

Boegh, J., Depanfilis, S., Kitchenham, B. and Pasquini, A. 1999. A method for software quality, planning, control and evaluation, *IEEE Software* 16: 69–77.

Briand, L., Wüst, J. and Lounis, H. 1999. Using coupling measurement for impact analysis in OO systems, *Proceedings of International Conference on Software Maintenance (ISCM)*, pp. 475–482.

Cartwright, M. and Shepperd, M. 2000. An empirical investigation of an object-oriented software system, *IEEE Transactions on Software Engineering* 26: 786–796.

Colombo, R. and Guerra, A. 2002. The evaluation method for software products, *Proceedings of ICSSEA'2002—International Conference Software & Systems Engineering and Their Applications*, Paris.

Dromey, G. 1996. Cornering the Chimera, *IEEE Software* 13: 33–43.

Fenton, N. and Pfleeger, S. 1996. *Software Metrics, a Rigorous & Practical Approach*, London, International Thomson Computer Press.

Hausen, H. and Welzel, D. 1993. Guides to software evaluation, Arbeitspapiere der GMD 746.

Heemstra, F., Kusters, R. and Trienekens, J. 1994. *From Quality Needs to Quality Requirements* (in Dutch). Leidschendam, Lansa Publishing.

IEEE 1471. 2000. *IEEE Recommended Practice for Architecture Description*, IEEE Std. 1471.

ISO 8402. 1994. *Quality Management and Quality Assurance Vocabulary*. Geneve, ISO/IEC.

ISO 9126. 1991. *Information Technology—Software Product Evaluation—Quality Characteristics and Guidelines for Their Use*. Genève, ISO/IEC.

ISO 14598. 1999. *Information Technology—Software Product Evaluation*, Parts 1–5. Genève, ISO/IEC.

ISO FDIS 14598. 1999. *Information Technology–Software Product Evaluation*, Part 6: *Documentation of Evaluation Modules*. Genève, ISO/IEC.

Kazman, R., Klein, M., Barbacci, M., Longstaff, T., Lipson, H. and Carrière, J. 1998. The architecture trade-off analysis method, *Proceedings of the 4th International Conference on Engineering of Complex Computer Systems*, pp. 68–78.

Kitchenham, B., Lawrence Pfleeger, S. and Fenton, N. 1995. Towards a framework for software measurement validation, *IEEE Transactions on Software Engineering* 21: 929–943.

Koshgoftaar, T.M. and Allen, E.B. 1998. Predicting the order of fault-prone modules in legacy software, *IEEE Transactions on Software Engineering* 24: 344–353.

Mayrand, J. and Coallier, F. 1996. System acquisition based on software product assessment, *Proceedings of International Conference on Software Engineering (ICSE)*, pp. 210–219.

Miller, K. and Dunn, D. 1997. Post implementation evaluation of IS/IT: a survey of UK practice, *Proceedings of 4th European conference on the Evaluation of Information Technology (EVIT)*, pp. 47–56.

Park, R., Goethert, W. and Florac, W. 1996. *Goal-Driven Software Measurement—A Guidebook*, SEI report CMU/SEI-96-HB-002.

Pivka, M. and Potocan, V. 1996. How can software packages certification improve software process, *Proceedings of International Conference on Software Quality*, Additional paper No. 3, 9 pages.

Punter, T. 1998. Developing an evaluation module to assess software maintainability, *Proceedings of Conference on Empirical Assessment in Software Engineering (EASE)*, Keele, 1998.

Punter, T. 2001. *Goal Oriented Software Evaluation* (in Dutch), Ph.D. Thesis, Eindhoven University of Technology.

Punter, T. and Lami, G. 1998. Factors of software quality evaluation, *Proceedings European Software Control and Metrics Conferences (ESCOM)*, pp. 257–266.

Punter, T., van Solingen, R. and Trienekens, J. 1997. Software product evaluation—current status and future needs for customers and industry, *Proceedings of 4th European Conference on Evaluation of Information Technology*, pp. 57–66.

Rae, A., Hausen, H. and Robert, P. 1995. *Software Evaluation for Certification. Principles, Practice and Legal Liability*. London, McGraw-Hill.

Space-Ufo Consortium. 1998. *The Space Ufo Methodology—User Guide*. Esprit project P22290.

Starlander, M. and Popescu-Belis, A. 2002. Corpus-based evaluation of a french spelling and grammar checker, *Proceedings of 3rd International Conference on Language Resources and Evaluation*, pp. 268–274.

van Solingen, R. and Berghout, E. 1999. *The Goal/Question/Metric Method—A Practical Guide for Quality Improvement of Software Development*. London, McGraw-Hill.

van der Zwan, M. 1995. *A Quality Profile—A Foundation to Certify Software Product Quality* (in Dutch), M.Sc. thesis, Eindhoven University of Technology.

Dr. Ir. Teade Punter is a competence manager for software product, process and subcontracting assessments at the Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE) in Kaiserslautern, Germany. He receives his M.Sc. from Twente University and holds a Ph.D. about goal-oriented software evaluation from Eindhoven University of Technology. His current research interests include subcontracting, collaborative software development, evidence-based software engineering and system dynamics.

Prof. Dr. Rob Kusters is a Professor in "Business process and ICT" at the Dutch Open University in Heerlen, the Netherlands. He is also an Associate Professor of "IT Enabled Business Process Redesign" at Eindhoven University of Technology where he is responsible of a section of the program in management engineering and he is an associate member of the research school BETA which focuses at operations management issues. Research interests include enterprise modelling, software quality and software management. Rob Kusters obtained his Master degree in econometrics at the Catholic University of Brabant and his Ph.D. in operations management at Eindhoven University of Technology.

Dr. Ir. Jos Trienekens is an Associate Professor at Eindhoven University Technology in the area of ICT systems development. He is responsible for a research program on ICT driven business performance and is an associate member of the research school BETA at TUE that focuses at operations management issues. Jos Trienekens published over the last ten years various books, and papers in international journals and conference proceedings. He joined several international conferences as PC member and as a member of organization committees. Jos Trienekens has experience as a project partner in several European projects (ESPRIT/ESSI) and is on a part-time basis with KEMA Quality, a service organization and certification body in The Netherlands.

Prof. Dr. Theo Bemelmans is a Full Professor at the Eindhoven University of Technology on the domain Information Systems. Formerly, he was for some years Director of the Research Institute on perception and cognitive sciences and he was Dean of the Department/Faculty Technology Management.

Prof. Dr. Ir. Aarnout Brombacher is a Professor in "Quality and Reliability Management" in the faculty Technology Management of Eindhoven University of Technology and professor in the same field at the department of Industrial and Systems Engineering of the National University of Singapore. Aarnout Brombacher obtained a B.Sc., M.Sc. (cum laude) in electrical engineering and a Ph.D. in engineering science at Twente University of Technology. Aarnout Brombacher has experience in industrial quality and reliability improvement projects and the development of quality and reliability analysis methods and tools. He has authored and co-authored over fifty papers on these subjects and has written a book with the title "Reliability by Design."