Fraunhofer IESE

Selection Criteria and Process for Quality Model Tailoring

Authors:
Constanza Lampasona
Michael Kläs
Reinhold Plösch

IESE-Report No. 034.12/E
Version 2.0
June 2012

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competitive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach (Executive Director)
Prof. Dr. Peter Liggesmeyer (Director)
Fraunhofer-Platz 1
67663 Kaiserslautern

Abstract

The objective of WP 3.3 in the Quamoco project was to develop a procedure for quality model adaptation based on existing content and on the proposed selection criteria.

In this document we present the mentioned selection criteria and an adaptation process for Quamoco quality models, which are both based on the requirements for adaptation mechanisms defined in the project as well as on the results from WP 3.1 on variability and tailoring mechanisms for quality models, and on the Quamoco meta-model v2.5.

We define the operations that are allowed to be performed when tailoring a quality model. These are basic operations (delete, add, and modify) and compound operations (tag, select, slit, and combine). The scope of the adaptation process is also defined in this document, based on the three levels where we consider that quality models can exist: public, organization and project levels.

The adaptation process itself is described by means of interconnected activities with defined inputs and outputs. We tried to leave this part as independent as possible from the quality meta-model. However, specific examples are given in connection with the Quamoco meta-model v2.5.

We complement the adaptation process with guidelines for quality model adaptation and typical adaptation scenarios. These are described less formally and intend to support the stakeholder performing the adaptation as much as possible.

Keywords: Adaptation, tailoring, quality model, methodological support, Quamoco.

Table of Contents

1 Introduction

Software quality models concretize the concept of software quality. In general, this is achieved by describing it as a set of quality attributes and related measures that can be used to evaluate the quality of software with respect to those attributes. Quality models can be created from scratch using methods to guide the derivation of custom-tailored quality attributes and metrics (define-your-own-model approaches), or, existing fixed quality models specifying a prescriptive set of quality characteristics or metrics can be reused. Unfortunately, fixed models are very often focused on a very specific context or are too general, and deriving a model from scratch very often requires intensive expert effort. Balanced quality models [18] are based upon the idea of adapting a core model for specific domains and specific purposes. This adaptation should be as much repeatable as possible and should therefore be methodically performed and guided by a tailoring process. The use of a base quality model and its systematic customization may support the cost-effective handling of quality needs within an organization or a project. Furthermore, the existence of a quality meta-model, as in the case of Quamoco, which specifies the same structure for all quality models, represents another potential advantage because it supports systematical quality model adaptation.

A central challenge for the model-based evaluation of software quality is the identification of an adequate model to be customized and successfully tailoring it to the needs defined by specific quality goals. Another fundamental challenge for effectively tailoring software quality models is identifying necessary changes (what, when and how is to be modified) as well as concrete consequential changes to guarantee the consistency of the tailored model. Furthermore, tailoring can occur at different levels of abstraction, which also should be considered, in order to allow tailoring a model from different points of view.

Existing software quality models are deficient when it comes to their adaptation to the needs of a specific organization or project in a reusable, reproducible manner. Based on our literature review [19], adaptations of quality models are generally based on ISO9126 [16]. Some common modifications are to define new attributes [2] or to add quality characteristics for a very specific domain [25, 4].

The literature related to adaptation methods for software product quality models is rather meager. For adapting the ISO9126 quality model to the do-main of component-based software development, Andreou and Tziakouris [2] take into account the users: component developers, re-users, and end users. The outcome is a quality model especially for original software components. Calero et

al. [6] use a general portal quality model for the creation of a spe-cial quality model for eBanking: BPQM (eBanking Portal Quality Model). To achieve the specialization of the model, a survey was performed among domain experts. Unfortunately, these specific adaptations focus on the resulting adapted quality models and not on a reproducible customization process.

Plösch et al. present in [23] a tool-supported approach to derive an adapted quality model for source code evaluation. They focus on tailoring a set of rules provided by static code analysis tools based on a set of defined criteria (e.g., relevant quality attributes or trustworthiness of the rule results). Although the scope and structural complexity of the models is limited compared to the Quamoco models, the general idea of providing a well detailed and comprehensive model that is primarily reduced during the adaption following certain criteria seem promising to ensure an efficient adaptation.

Other authors use define-your-own-model tools to refine their specific mod-els. These customizations have a narrow focus and are difficult to transfer to other contexts. Andersson and Eriksson [1], for example, present a process for the construction of a quality model founded on a basic quality model with existing metrics (SOLE quality model). They illustrate how to customize the model to the specific needs of an organization, including how to identify quality factors and mapping them to metrics. The SOLE quality model [11] has the factor-criteria-metric [21] structure. Bianchi et al. [5] use GQM [3] to refine a specific model. They center their research on quality model reuse, i.e., which changes can be requested when a quality model is reused, how to verify that despite the changes made in the reused quality model, it remains suitable for its goals, and what the side effects caused by changing the metrics are on the quality model. Horgan and Khaddaj [15] propose an approach for model refinement based on expert knowledge. Franch and Carvallo [13] present a very general process to build an ISO9126 quality model.

The project Quamoco adheres to the concept of balanced quality models and therefore proposes the creation and maintenance of a domain-independent base quality model which may be adapted according to specific quality needs. This document describes such an adaptation process and gives guidelines for tailoring by proposing a series of selection criteria that should help the stakeholders when adapting a quality model.

In chapter 2 we address the requirements for adaptation mechanisms extracted from the quality model requirements specified in WP 1.1 [20]. We define the operations which can be performed to customize quality models in chapter 3. In chapter 4 we specify the scope of quality model adaptation. Chapter 5 gives an overview on the general adaptation process. In chapter 6 we give a detailed analysis of the typical scenarios for using the adaptation process. Chapter 7 describes criteria which support the stakeholder in taking the

right decisions when adapting a quality model. Chapter 8 explains in detail how to define a goal for an adapted quality model and how use it for pre-tailoring a model. Chapter 9 describes in detail how to iteratively perform quality model modifications. Chapter 10 explains how changes are docu-mented. Chapters 11 and 12 present topics closely-related with quality model adaptation and how they influence each other.

2 Requirements for Adaptation Mechanisms

The mechanisms to adapt the Quamoco quality models are related to the following requirements, which were specified in WP 1.1 [20]

- Requirement 96: Ensure Proper Configuration and Application
- Requirement 98: Setup Possible within a Few Days
- Requirement 123: Adaptation of Quality Models
- Requirement 124: Conformance of Adapted Quality Models
- Requirement 129: Base Model

Based on these requirements, we condensed three major requirements with respect to the adaptation process:

(R1) Correctness – An adapted quality model must be syntactically correct in that it remains compliant to the underlying meta-model and defined consistency rules. In addition, the resulting model should be sound. However, the adaptation process can only support but not enforce the soundness (e.g., by guiding the adaptation hints or providing support for making the decisions taken during the adaptation traceable).

(R2) Goal Orientation – The adaptation of a base model should be driven by organizational needs and capabilities. In particular, organization-specific and project-specific software quality objectives should be considered.

(R3) Efficiency – This is concerned with the overhead (e.g., personnel, time, and budget) needed for adapting a quality model. Acceptable overhead would differ depending on the organizational level (e.g., more overhead will be allowed for adapting a quality model at the level of the whole organization, where such adaptation has a larger scope and is performed relatively rarely).

We refer to the solutions proposed for each requirement throughout the document.

3 Adaptation Operations

Which changes can be requested when a quality model is adapted?

The process of adapting a quality model can be seen as a series of transformation steps. Each transformation would convert a quality model into anoth-er one, by performing an operation. We call basic or primitive operations those operations that can produce atomic transformations in the model (add, delete and modify). Moreover, we refer to compound operations to those operations that support a transformation in a higher level of abstraction (select, tag, combine and split). All compound operations can be constructed by combining basic operations.

3.1 Basic Adaptation Operations

From the analysis of the existing tailoring mechanisms [19], including those being used by the Quamoco partners, we deduced that any customization can be reduced or organized into a combination of atomic changes. We define, therefore, three primitives or basic operations, which are used to perform those elemental changes:

- **Delete**: Identification of elements not needed and their removal. We use DEL to represent this operation.
- **Add**: Identification of needed elements not included and their incorporation. We use ADD to represent this operation.
- **Modify**: Identification of further needed adjustments and their implementation. We use MOD to represent this operation.

Figure 1 presents a simplified adaptation process for tailoring a quality model which depicts the definition of the three basic operations. In essence, deleting something in a model results in a reduced model, adding something to a model produces an extended model, and adjustments of existing content in a model are accomplished by means of modifications. We call the model used as input for the adaptation reference quality model. The quality model obtained after tailoring is an adapted quality model. Although the figure presents a sequence of three operations, they do not have to happen in this order. Instead, a series of many operations are performed iteratively.

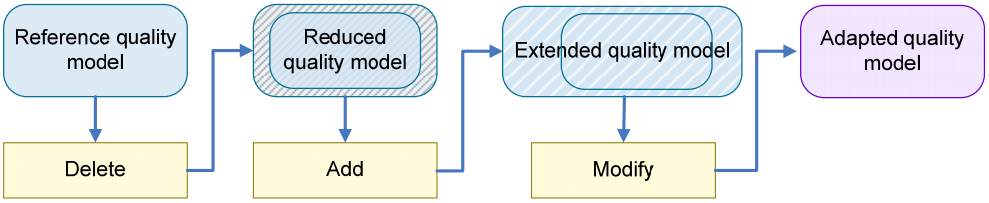


Figure 1: General adaptation steps

3.2 Compound Adaptation Operations

While the basic operations serve to implement any desired quality model customization, we list here also compound operations that refer to predefined combinations of basic operations which have a strong semantic power in the context of adaptation. These operations are:

Tag: This is the assignation of a mark, i.e., a tag name and a tag value, to an element or group of elements. This operation serves to add additional information to elements which was not defined as an element attribute in the metamodel. Normally, this additional information is something specific related to the context in which the model will be used. Therefore, we refer to a tag also as a context attribute. This operation is particularly useful when the stakeholder needs to add specific information to a model. For example, "programming paradigm = OO" or "inclusion of element = mandatory".

Select: A selection is an indirect implementation of the delete operation, that is, elements needed are identified and all other elements are removed from the model.

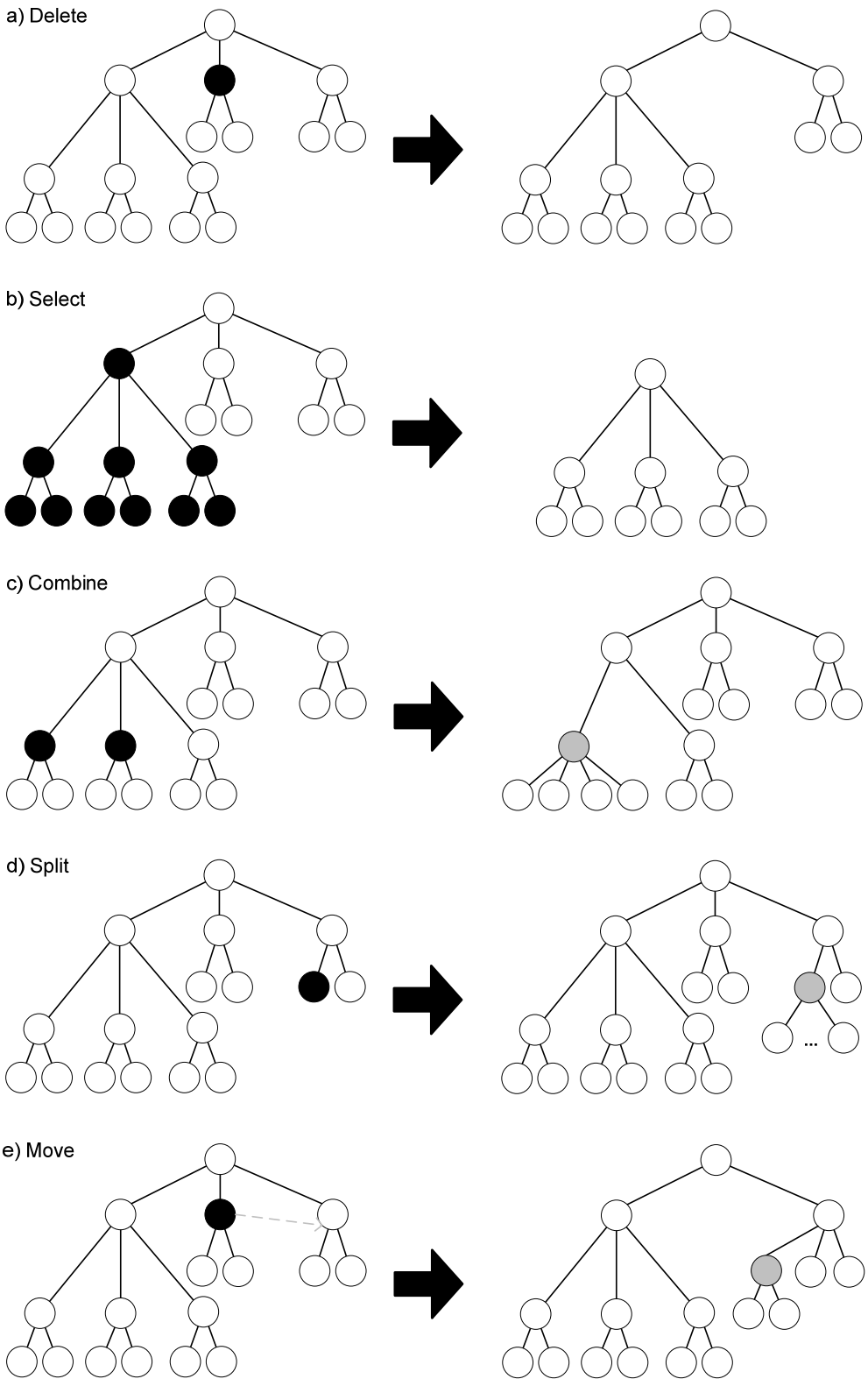
Combine: Using this operation two or more elements can be merged into one element.

Split: This operation is used to divide one element into two or more elements.

Move: This operation moves one element including its sub-elements to a new position. This equals a combination of a delete and an add operation.

Simple Import: This operation consists in adding a whole tree from another model. This assumes that the quality model and the tree added are independent. Diff and Merge functions are out of scope the work package.

In Figure 2, we schematically illustrate the adaptation operations delete (a), select (b), combine (c), split (d), move (e) and simple import (f).



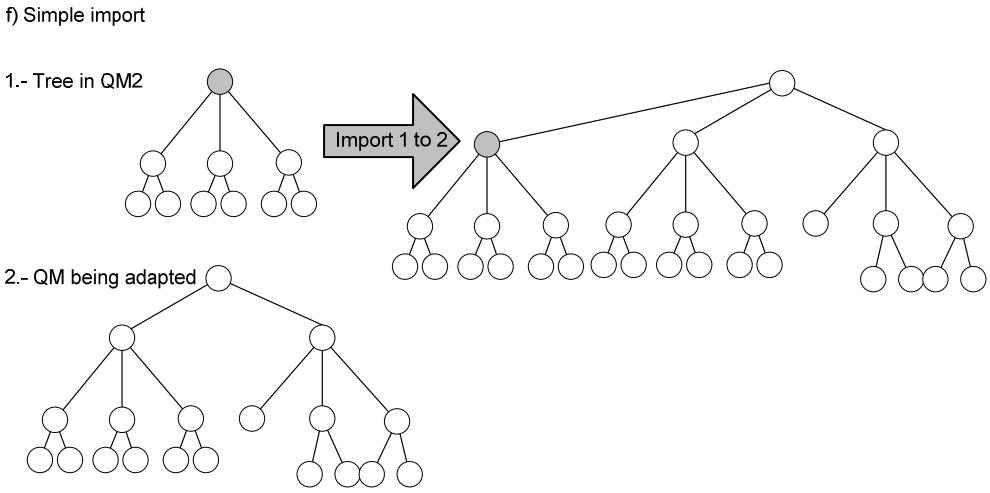


Figure 2: Adaptation of quality aspects: On the left we show the quality aspect hierarchy in the reference model and on the right the quality aspect hierarchy on the adapted model. a) Delete the quality aspect in black. b) Select the quality aspects in black. c) Combine the quality aspects in black. d) Split the quality aspect in black. e) Move the quality aspect in black to the right sub-tree. f) Import tree in QM2 to QM being imported.

4 Scope of Quality Model Adaptation

Considering that software quality models may exist and be applied at different levels, tailoring a quality model may also be needed at those different levels. Analogically to the levels for tailoring software processes, quality models may be developed and maintained at three levels (Figure 3):

Public level: The models at this level are universally available, may be intended for general use (e.g., ISO9126 [16], Quamoco Base Model), or may be intended for some specific domain (e.g., IEC 61508 [9] for safety in embedded systems, EN 60601-1-4 [10] for medical devices). This level is comparable to the broad *industry level* described by Fitzgerald [12]. Most of the models at this level are very generic and usually not operational and need to be customized. Using and tailoring these models is however useful for showing adherence to a public standard.

Organization level: At this level, the main emphasis is on creating quality models that satisfy the quality aspects of interest for an organization such as a whole organization, a business unit, or a project portfolio. The quality models here focus on issues that are relevant organization wide, i.e., they include overall organization's quality requirements. They are more specific than quality models at public level and include aspects not covered at it. They are common models intended to be tailored for particular projects. An organization can have more than one organization quality model, each adapted to a different scope. Sub-organizations or projects may adapt the models at organization level according to what is relevant for them. At the organizational level, the performance of quality models is the subject of continuous improvement, and, in order to increase efficiency and decrease costs at project level, major quality model adaptations should take place at this level. Mandatory quality model elements, which must be considered by every project in an organization, are defined here. However, optional elements that do not have to be taken into account by every project but reflect the implementation of good practices can also be included at this level.

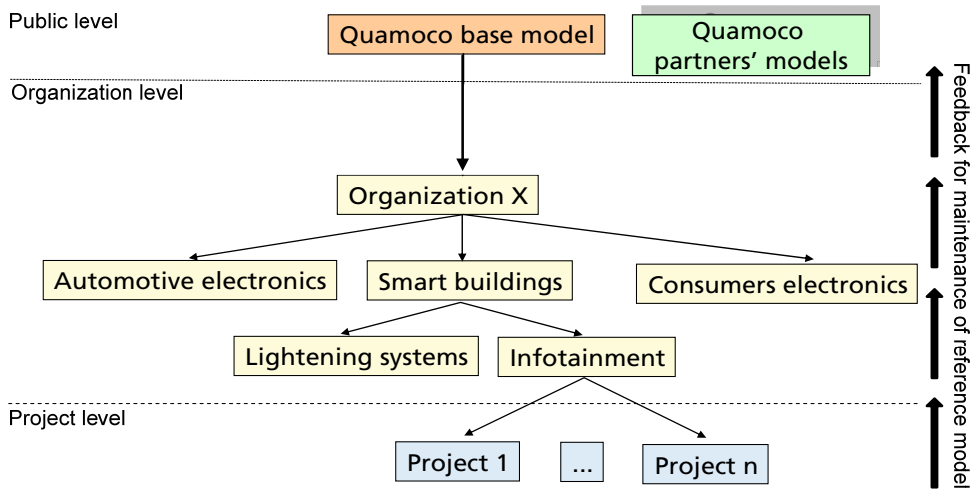


Figure 3: In this example, a public quality model (Quamoco base model) is adapted for organization x. This organization is divided into three sub-organizations, which adapt the organization quality model for specific needs: automotive electronics, smart buildings, and consumer electronics. The smart building division is structured into two departments: lightening systems and infotainment. The latter has tailored quality models for many projects. Specific adaptations are given as feedback to be considered when maintaining reference models.

Project level: At the project level, quality models are put into operation, i.e., they are used for their application purpose with respect to software product quality. At this level, quality model adaptation should be limited to minor adjustments driven by project-specific quality requirements, without drastically changing the structure of the quality model, in order to preserve conformance of quality evaluations across software products created at the project level. Organization's quality models are tailored for specific projects by performing only fine tuning, i.e., at the project level organization-wide models are further refined for a particular project. At this level, those quality model elements are chosen that are needed by a project and the quality models produced at this level should be always operational, i.e., tailoring at this levels is of pragmatic nature. Major quality model adaptations may take place also here, if a project is large enough to justify the costs.

The reuse of a quality model for adaptation is in essence more efficient than creating a new model from scratch for each project. We recommend tailoring models stepwise, for organizations and for projects. In this way, requirement 0[1] is addressed. As illustrated in the picture, we enforce that a model inherits only from one reference model explicitly. This limitation avoids the inherent problems with multiple-inheritance. However, this fact does not prohibit that certain parts of non-reference models are reused in the adapted model (which means

[1] 0 (R3) Efficiency – This is concerned with the overhead (e.g., personnel, time, and budget) needed for adapting a quality model. Acceptable overhead would differ depending on the organizational level (e.g., more overhead will be allowed for adapting a quality model at the level of the whole organization, where such adaptation has a larger scope and is performed relatively rarely).

that an independent tree from another quality model is imported to the model).

5 General Quality Model Process

The process for tailoring a quality model consists of 5 steps (Figure 4). First of all, the quality model goal has to be identified. This goal can then be mapped to the quality model selection criteria. The criteria help the user in two ways. First, they help the stakeholder to choose the model to be tailored (the reference model). Second, they help to decide which changes are needed on the reference model in order to make it satisfy the goal defined. Detailed adaptation scenarios where this process is instantiated are provided in chapter Typical Adaptation Scenarios6 “Typical Adaptation Scenarios”.

Once a reference model was chosen, the actual tailoring process can start. The adaptation typically starts by discarding elements that are not needed in the model (pre-tailoring or pre-selection). Afterwards, tailoring operations (delete, add, modify) are performed iteratively. Parallel to this, a list of adaptation tasks is generated that suggests further customization based on the relationships among the model elements and on the selection criteria. Consistency checks are triggered after each operation. The tasks needed to bring the model into a state which is consistent with the structure of the meta-model are given to the user in a list of adaptation tasks. The process is documented during adaptation. In the next paragraphs we give a detailed description of the process.

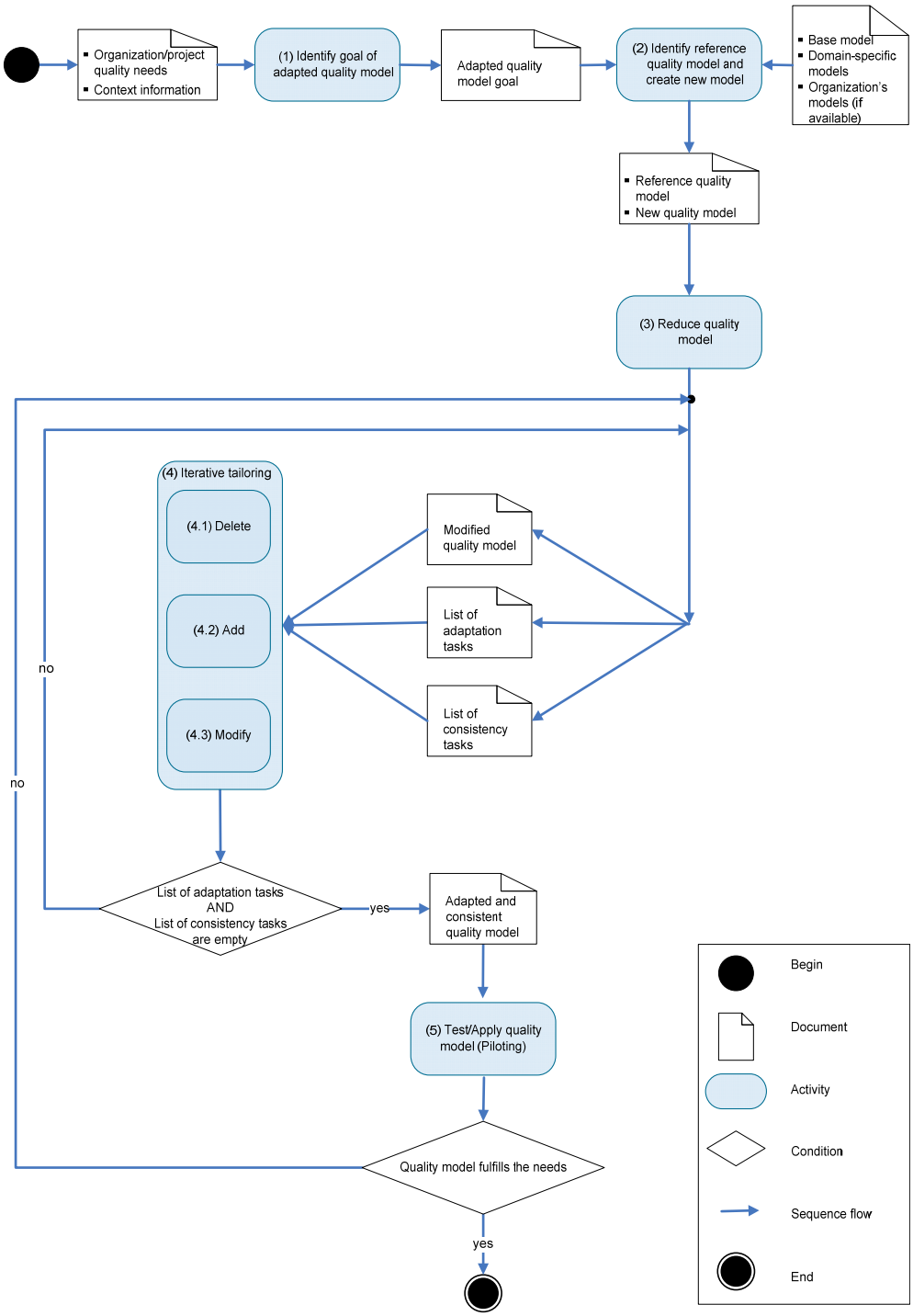


Figure 4: Adaptation process and piloting, schematic view.

Step 1 (S1): Identify goal of adapted quality model

INPUT: Organization/project quality needs, context information (e.g., domain, programming language).

OUTPUT: Adapted quality model goal.

ACTIVITIES: In this step, a quality model goal is defined. The objective is to identify the fundamental characteristics of the desired quality model (i.e., the adapted quality model). The organization/project quality needs and context information are used to define this goal.

The goal of the adapted quality model is formulated in a structured way. This can be done using the GQM goal template [3], which contains five parameters: (1) Object (the entities to be considered), (2) Viewpoint (from which perspective we consider them, e.g., quality in use or technical issue), (3) Quality Focus (the factors of interest), (4) Purpose (the intention of the adapted quality model), and (5) Context (the circumstances under which the quality model is used). Formulating the goal in this way allows easily using the adaptation support provided by the selection criteria, which are also based on these five goal parameters. The adapted quality model goal and the selection criteria provide fundamental support for identifying necessary changes (what, when and how is to be modified). The goal is documented and can be used later, if the model is inspected, to corroborate that it actually serves for the stated goal.

The goal of a model is used to characterize it. We use the term quality model attributes to refer to the different parts of a goal in a model.

A goal can be, for example:

Purpose	Object	Viewpoint	Quality Focus	Context
Evaluate	the source code	from the perspective of Product Quality	with respect to Reliability	for a product developed with Java.

RELATED ROLES[2]: Project Leader, Quality Assurance Manager (QAM).

[2] Roles as defined in [14].

Step 2 (S2): Identify and duplicate reference quality model

INPUT: Adapted quality model goal, base model, domain-specific models, organization's models (if available).

OUTPUT: Reference quality model, duplicate of reference quality model.

ACTIVITIES: In this step, a reference model that best fits the adapted quality model goal defined in STEP 1 is selected. A reference quality model is the model on which the adaptation is based. This means that an adapted quality model is derived from a reference model. It can be, for example, the Quamoco base model, a domain-specific model, or an organization's model.

In order to find a reference model, the values of the goal parameters are used. These values are searched among the quality model attributes of the available models and among the hierarchies of factors and entities to exclude quality models that are not interesting with regard to the adapted quality model goal.

If no model exactly fits the goal, then the goal is used partially to search for models satisfying the most relevant goal parameters. I.e., the goal parameters can be prioritized and used as a filter to select the model that at least satisfies the goal parameters that are considered more important. The quality model that best fits the adapted quality model goal is now the reference model.

Once the reference model was found, a copy of it can be created, on which the adapted quality model will be built, and values can be assigned to its attributes (purpose, object, viewpoint, quality focus, and context).

RELATED ROLES: QAM.

Step 3 (S3): Reduce quality model (Pre-Tailoring)

INPUT: Reference quality model, duplicate of reference quality model.

OUTPUT: Modified quality model, list of adaptation tasks, list of consistency tasks.

ACTIVITIES: Once the new model has been created, only quality model components in the reference model that are relevant for the new model are taken over. In this way, unnecessary quality model components are eliminated (this is a select operation). In this step, the elements that are included in the new quality model are taken over together with all their related elements. For example:

Top-down pre-tailoring: if we choose the factor *maintainability*, we are automatically including all maintainability sub-factors, all the factors that influence them, measures related, etc. (unless they are explicitly unselected).

Bottom-up pre-tailoring: in this case pre-selection of quality model elements is based on, for example, the availability of measurement tools. For example, the tool PMD may be chosen and only the factors that can be measured with it will be automatically selected.

The selected parts are the foundation for further adaptation. Basically, two things are identified in this step: the factors relevant for the model quality focus and the entities that are to be considered. During this step, the lists of adaptation and consistency tasks are created. These tasks guide the further adaptation steps. The list of consistency tasks includes hints that indicate which actions must be taken to bring the model into a syntactically valid state, i.e., conform to the structure prescribed by the meta-model.

RELATED ROLES: QAM, optionally: stakeholders with experience with respect to the quality foci in the goal of the adapted quality model.

Step 4 (S4): Iterative tailoring

INPUT: Modified quality model, list of adaptation tasks, list of consistency tasks.

CONDITION TO FINISH THIS STEP: List of adaptation tasks and list of consistency tasks are empty, i.e., the adaptation is done, the quality model is syntactically correct, and the quality model content satisfies the defined goal.

INTERMEDIATE OUTPUT (BETWEEN ITERATIONS): Modified quality model, list of adaptation tasks, list of consistency tasks.

FINAL OUTPUT: Adapted and consistent quality model.

The adaptation approach uses the basic operations to stepwise tailor the model. The model transformations resulting from the execution of an operation do not necessarily produce a syntactically correct, consistent quality model. Compound operations, i.e., tailoring operations that involve a combination of basic operations, e.g., splitting a factor into two quality aspects, are also performed here.

The preservation of consistency is achieved through the use of consequential changes, which in turn may generate further consequential changes. If an element is deleted from the model, and the element had related elements that are no longer needed in the model, those elements should also be deleted.

These consequential changes needed to bring the quality model back into a consistent state and produced by the adaptation of an element are based on the meta-model structure and are saved on the list of consistency tasks, as mentioned in STEP 3. This list is organized as a set of basic operations on the model elements.

A basic adaptation produces a list of tasks, i.e., needed consequential changes, to be performed in order to reach consistency. The sub-steps delete, add, and modify are performed iteratively until no more customization is needed. The extent to which these operations are used depends on the appropriateness of the reference model, i.e., it depends on how similar the goals of the reference model and of the adapted quality model are.

Step 4.1: Delete

ACTIVITIES: This sub-step focuses on performing delete operations, i.e., in identifying elements not needed and removing them. Once an element is eliminated from the model, consequential tasks are generated and added to the lists [3].

Step 4.2: Add

ACTIVITIES: Here, elements needed, which are still not included in the model, are incorporated. Add operations extend the quality model by adding quality model elements, such as factors and measures. A new element can be created by the user, or, can be searched in another model. That is, individual elements partially satisfying the goal of the adapted model can be taken from other models and reused in the new model.

If a new element is created, its relationship to the already existing elements in the model is analyzed. For example, if we create a factor, as a sub-factor of maintainability, we need to study the relationship of the new sub-factor with the quality model, the factor maintainability, the other sub-factors of maintainability, the factors related to the new factor, etc.

If an element is selected from another quality model, its relationship to the already existing elements in the model is likewise analyzed. For example, if we take a factor from another model, we have to harmonize it with the model. This means that we are not only taking an isolated element. We are adding a factor with sub-factors. All the related elements of the one being added have to be controlled, i.e., if they already exist in the model, they will not be duplicated, but their associations with other elements are updated.

[3] Consequential tasks are used to manage change throughout the adaptation process, based on the structure provided by the meta-model. A table documents the consequences of the changes in one element on other elements, whether the task can be performed automatically or if it requires user interaction.

Step 4.3: Modify

ACTIVITIES: Further needed adjustments are performed according to the needs of the lists of adaptation and consequential tasks. This can be, for example, changing the metrics used in a measure, or changing the contribution points for assessment.

RELATED ROLES: QAM, stakeholders with experience with respect to the quality foci in the goals of adapted quality model.

Step 5 (S5): Piloting/Test

INPUT: Adapted and consistent quality model, a selected product for test.

OUTPUT: Tested adapted and consistent quality model.

ACTIVITIES: In this step, the adapted quality model is tested at a small scale, i.e., pilots can be conducted to validate the adapted quality model. This can lead to a rejection of the model. Based on the results further adaptations can be performed.

RELATED ROLES: QAM.

Step 6 (S6): Documentation

ACTIVITIES: In parallel to quality model adaptation, relevant decisions and justifications are documented, and, for projects, especially those changes that exclude elements that were tagged as mandatory in the organization's model.

RELATED ROLES: QAM, organizational QAM (responsible for organizational quality model and the mandatory use of elements).

6 Typical Adaptation Scenarios

Based on discussions with the Quamoco project partners, we formulated a set of typical adaptation scenarios. The scenarios presented here complement the existing general adaptation procedure. They should help making the adaptation process more concrete and easier to apply in practice by providing scenario-specific guidance for the adaptation of quality models. We proposed 16 different scenarios:

Q1.- New data sources (e.g., add an additional static analysis tool): A model exists and is extended. This means that it is not necessary to reduce the quality model. Therefore, jump to iterative tailoring. Add new tool and associate to measures through instruments. If the tool covers new measures, they may need to be added to the model as well as the associated instruments and factors. Evaluation specifications have to be checked for correctness.

Q2.- New modeling/implementation languages (e.g., add additional programming language): A model exists and is extended. This means that it is not necessary to reduce the quality model. Therefore, jump to iterative tailoring. If the existing factors are not adequate for the new programming language, new factors have to be added to the model, as well as measures, instruments, and tools. If the factors can be reused, measures can be modified, as well as instruments and tools.

Q3.- New technologies or paradigms: In the case of extending the model for new technologies or paradigms, new factors relevant for the new technology are added. This means that also measures and measurement instruments/tools as well as evaluations for the new factor are to be defined. If existing factors can be reused, additional evaluations applying for the new technology, are to be defined, for example, where the weights are different.

Q4.- New artifacts (e.g., add additional entities): A model exists and is extended. This means that it is not necessary to reduce the quality model. Therefore, jump to *S4 Iterative tailoring.* If, for example, the current model considers only source code and it should also consider design documents, new entities describing design documents have to be added to the model, as well as factors describing their quality, measures associated to the factor, instruments and tools. If some factors can be reused, the new entities have to be related with them, new measures can be defined for them, and also the evaluations have to be recalculated.

Q5.- New domains: For adapting a model for a new domain, we recommend beginning with pre-tailoring. During that activity the elements which both domains have in common can be adopted for the new model and the other elements simply discarded. Afterwards iteratively tailoring the model is necessary to complete the model with new factors, measures, instruments and tools that are relevant for the new domain and were not included in the reference model.

Q6.- New/modified quality requirements: In the case of new quality requirements, for instance, a factor is selected that is appropriate for a goal as well as product factors influencing it. Requirements are then built from these and evaluations are defined which describe the fulfillment of the requirements. This means, that new factors and evaluations are needed, where the impact of product factors is assessed.

Q7.- New/updated perspectives: In case of adding a new perspective, this would translate in adding an additional factor tree, which contains new factors and sub factors describing quality from that new viewpoint. New impacts from an existing product factor hierarchy into the new defined hierarchy are to be set. Existing factors may be copied into the new tree. In that case, it would be necessary to adapt the factors' evaluations, e.g., weights, in order to reflect factors' importance for the new perspective.

Q8.- Using a subset of data sources: In this case, the available resources can be selected during pre-tailoring, for instance, selecting only the available tools, or measures according to the cost of data collection, trustworthiness or relevance. Afterwards, during iterative tailoring, the model can be brought into a consistent state, by executing the tasks in the To-do-List, such as re-assigning weights to measures.

Q9.- Focusing on specific modeling/implementation languages: If the model contains, for example, many implementation languages, and only a subset of them are to be addressed in the adapted model, the model can be pre-tailored, and during this operation, the relevant languages are defined in the model goal. In this way, only the model elements tagged with the relevant languages are filtered and the uninteresting ones discarded.

Q10.- Reduce technology or paradigm scope: In this case, the procedure is similar with Q9. During pre-tailoring, the relevant technology or paradigm are defined in the context of the new model and only the model elements tagged with the relevant paradigm/technology are kept. The elements that are not relevant are discarded.

Q11.- Assessing a subset of artifacts: For selecting a subset of artifacts, relevant entities are selected during pre-tailoring. That is, for example, if the refer-

ence model describe quality based on source code and documentation, but only source code is interesting, the documentation entities are discarded in the pre-tailoring step, and the source code ones are kept in the model.

Q12.- Focusing on actual domain: If the model is constructed for many domains, such as information systems and embedded systems, and, only embedded systems is valid for the target product, during pre-tailoring, the context of the new model is defined as embedded systems and the elements which only apply for information systems are discarded.

Q13.- New/modified project-specific quality requirements: In this case, if a requirement for a specific project changes, for example, if a lower coupling level is required than the one defined in the reference model, the evaluation defining the fulfillment condition of that requirements has to be adapted, for example, by defining a new threshold.

Q14.- Performing assessments limited to specific perspectives: In order to tailor the model to a specific perspective, this must be defined in the model goal. Then, only the relevant factor hierarchy addressing that specific perspective is kept in the model.

Q15.- Fixing observed assessment problems: Assessment problems may occur, for example if measures or evaluations were wrongly defined. In this case it is necessary to track the problem back to the faulty model elements and correct the modeling mistake, for example if an inadequate measure was used for normalization or weights were wrongly assigned.

Q16.- Calibrating the model for future assessments: In this case, evaluations rules can be modified in order to fit better into the environment. For example, by giving a higher weight to more trustworthy measures.

7 Selection Criteria

An **adaptation decision** is taken after considering a criterion, which supports the decision. A **justification** (adapted from [22]) is a property of an adaptation decision and it captures a summary of the reasons for taking that adaptation decision, especially when an adaptation decision did not arise from an existing criterion. A criterion is the base on which an adaptation decision is taken. **Criteria** support the stakeholder in taking the right decisions, i.e., the decisions that trigger the **adaptation tasks** which transform the reference model into a model which satisfies the goal of the adapted quality model. Defining such a **goal** is the first step for any adaptation of quality models. Conformance with the meta-model is achieved by performing **consistency tasks**, which bring the model into a syntactically correct state (**Figure 5**).

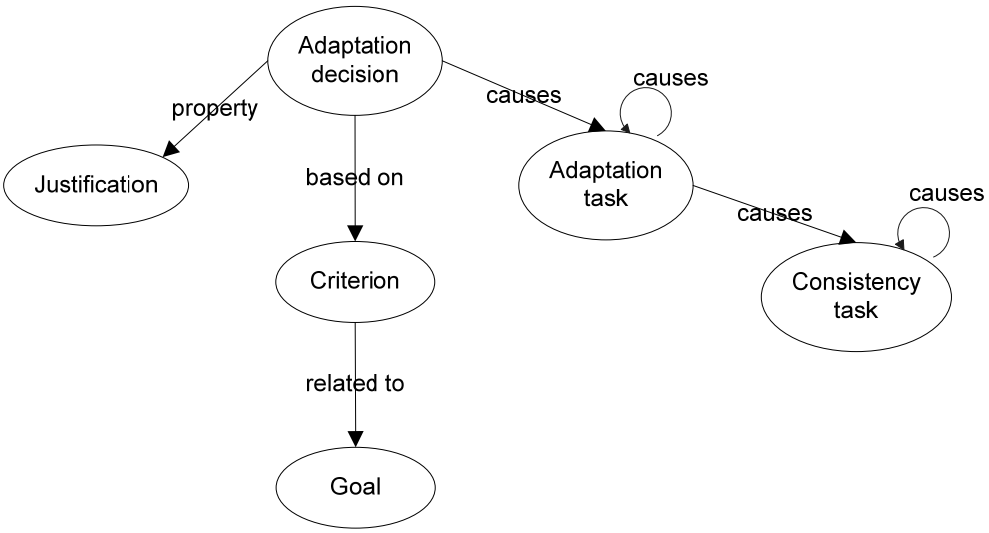


Figure 5. Structure of an adaptation decision, its causes and consequences.

What are influencing factors and motivations for adjustments?

The selection criteria supporting an adaptation decision is also structured in a goal-oriented manner. A goal-oriented approach supports a methodical adaptation process in that the criteria are categorized and can be systematically mapped to quality model elements. This approach helps also the stakeholders in focusing on the needs of the organization or project.

Each one of the goal elements can trigger adaptation decisions, that is, the goal parameters are bases for criteria. The adaptation decisions can include the

justification of a change in the quality model and serve to document specific tailoring decisions, which are based on particular criteria (e.g., a decision taken by the quality assurance manager without considering the criteria proposed here, but based on previous experience).

The criteria presented here have a rather general character. As the model and the experience using it increases, the criteria can be extended and concretized adding more details within quality model maintenance (whether this is done with the Quamoco base model or with the models in an organiza-tion).

Object-related Selection Criteria

The object of a goal refers to one or more entities in the quality model. According to the Quamoco meta-model v2, entities are described by the entities hierarchy, and they are characterized by a factor, which can be measured and, it is either a refinement of other factors or influences one or more other factors.

Examples for entities are: documentation, source code, requirements, design, build process, test suite, components, statements, classes, memory, subroutines, macros, functions, and procedures.

An abbreviated name is given to each criterion: [Cr=Criterion, Ob=Object-related | V=Viewpoint-related | QF=Quality Focus-related | P=Purpose-related | C=Context, consecutive numbering].

[CrOb1] Need to exclude from the model all entities which are not included in the object of the goal: All factors in which the entity does not corre-spond to the object specified in the quality model goal can be re-moved from the quality model by applying the rule: DEL(entities □ instance_of(Object)). For example, if we are interested in source code, this operation removes requirements- and design-related factors from the quality model.

[CrOb2] Need to include in the model all entities which are included in the object of the goal: It may also happen that some entities of interest are still not included in the model, so they need to be included in it.

Viewpoint-related Selection Criteria

A viewpoint can be, e.g., developer, management, quality assurance, project manager, customer, and user. The perspective from which quality is specified or evaluated refers to factors hierarchies in the quality model. In one quality model, there can be many parallel factors hierarchies corresponding to differ-ent

viewpoints. Moreover, it is possible that many viewpoints share a com-mon factor hierarchy for the specification of quality and they differ in how they evaluate it. So, in order to find the right elements corresponding to the point of view in the quality model goal, the following criteria are considered:

[CrV1] Need to identify the factor hierarchy that corresponds with the view-point.

[CrV2] If the purpose of the adapted quality model is to evaluate, it is needed to include the evaluation elements which reflect the perspective in the viewpoint of the goal.

Quality Focus-related Selection Criteria

The quality focus of the adapted model is directly related to the factors considered in the quality model. The factors are included either in the "viewpoints" hierarchies or in the product hierarchy, according to the Quamoco meta-model v2. Examples of quality foci are maintainability, functionality, portability, efficiency, usability, and reliability.

The first thing to do is to match the adapted model quality focus with the quality focus on the reference model. If a factor already exists in the reference model, the factors and sub-factors related to it have to be checked.

For all factors influencing a factor of interest, check if they are relevant. If they are not relevant:

[CrQF1] Need to exclude the connection of all influencing factors that are not relevant for a considered "quality focus" factor.

For all sub-factors that refine a factor of interest, check if they are relevant. If they are not relevant:

[CrQF2] Need to exclude the sub-factors that are not relevant for a considered "quality focus" factor.

If a factor of interest still does not exist in the reference model:

[CrQF3] Need to add the factor of interest to the new model.

Purpose-related Selection Criteria

The practicable application purposes of a quality model are determined by those supported by the structure given by the meta-model. For example, if the quality meta-model does not support the estimation of final product qual-ity as

an application purpose, the concepts relevant for final product quality estimation cannot be realized by any quality model based on such meta-model.

The Quamoco meta-model v2 supports the application purposes of specification and evaluation. Specification means that a concept is described by refining it into sub-concepts. Evaluation means that a concept is quantified, measured, and compared to defined evaluation criteria to check the fulfillment of the criteria. The Quamoco meta-model has a specification part which qualitatively describes software product quality. It is represented by factors, which characterize entities. The evaluation part of the Quamoco meta-model extends the specification part by defining software product quality in a quantitative way. It includes measures (quantifications of factors) and evaluations of factors, in addition to the elements in the specification part. More tailoring criteria arise out of these purposes supported by the meta-model v2.

If the purpose of the adapted quality model is to specify quality and the purpose of the reference model is to evaluate quality:

[CrP1] Need to delete elements corresponding to the evaluation part of the model.

If the purpose of the adapted quality model is to evaluate quality and the purpose of the reference model is to specify quality:

[CrP2] Need to add elements corresponding to the evaluation part of the model.

Context-related Selection Criteria

Although the context of a quality model is not defined in the Quamoco meta-model v2, it is a very important aspect that should be considered, especially for adapting quality models. The quality model context refers to environmental variables that affect the model. It is an inherent characteristic of every quality model and its elements, especially essential for making a quality model operational. That is, if a model has to be more than a conceptual abstraction of software product quality and if it is meant to adequately serve to be applied for actual software, it is unconceivable to exclude the model context information. Otherwise, the model would be too general.

When we studied the existing adaptation mechanisms, we identified the most common context attributes. They are: domain, development methodology and practices, development paradigm, and programming language. Other more project-/organization-specific context attributes can be also considered, such as scope of application and the available resources.

Domain: The domain is the sphere of application of the product. Examples for domains are railway systems, medical devices, embedded systems, and information systems.

Development Methodologies: This context attribute refers to the set of methods and practices used, and rules followed for the software develop-ment. Examples for development methodologies are agile, open source, and customer development.

Development Paradigm: The development paradigm describes how the re-al-world is conceptualized as software. Examples for development paradigms are object-oriented, structured programming, event driven, and data oriented development.

Programming Language: This is the programming language or languages used for writing the software. Examples are C, C++, and Java.

Scope of Application: Under scope of application we want to include criteria which are not covered by the other context attributes and which are specifically related to software quality. Examples are: the mandatory inclusion in the quality model of requirements for all projects within an organization, the classification of models' elements into levels to allow using the model for one or more levels including more details, the use of measures based on their level of importance (or trustworthiness) [23] for the organization according to its experience.

Elements in which the context does not correspond to the context specified in the quality model goal can be removed from the quality model, or their context must be extended to include the context specified in the quality model goal.

[CrC1] Need to delete elements which are for contexts other than the context in the goal of the model.

[CrC2] Need to modify elements which are for contexts other than the con-text in the goal of the model, including in their context tag the con-text specified in the quality model goal.

Elements which are relevant for the quality model context that are not considered in the model have to be included in it.

[CrC3] Need to add elements for the context defined in the goal of the model and still are not included in it.

Concrete examples for the influence of the context on a quality model are:

- For open source development, factors can be added reflecting the influ-ence the development community on the product quality. Soto et al. [26] describe three community-related quality characteristics: (1) maintenance capacity is defined as the ability of a community to provide the resources necessary for maintaining its product(s) over a certain period of time; (2) sustainability is the likelihood that a free and open source software com-munity remains able to maintain the product(s) it develops over an extended period of time; and (3) process maturity which is defined as the ability of a developer community to consistently achieve development-related goals by following established processes.
- For component-based development, Andreou and Tziakouris [2] redefine the factor usability (their reference model is ISO9126). This is because the users of components are not the end-users of software, but developers. One change that they introduce is the addition of the sub-factor identifia-bility-reachability which refers to the facility with which a component can be found and identified (discovered) and retrieve for usage. This is justified by the need of efficiently tracing and retrieving the desired components.

  Chidamber and Kemerer [7] defined metrics which especially apply for object-oriented development: weighted methods per class (WMC), depth of inheritance tree (DIT), number of children (NOC), coupling between object classes (CBO), response for a class (RFC), and lack of cohesion in methods (LCOM). All these metrics serves only to measure factors, whose entities are object-oriented concepts.

  For development in the programming language Java, measures for other programming languages are excluded.

8 Goal Definition and Pre-Tailoring

8.1 Goal Definition

Where do I start with quality model adaptation?

The first thing to do, as we described in STEP1 of the adaptation process is to define the goal of the adapted quality model. In order to define the goal, we need first to know the organization/project quality needs and context information (scope). That is, it is necessary to identify the different parameters of the adapted quality model goal that are consonant and aligned with the quality needs of the organization or the project, and with the context information (the circumstances under which the quality model is used).
The very first time it is necessary to decide on the need of an organization model. One will probably need an organization model, if the answer to the question of how many products the organization develops is "more than 1".
Defining a goal means identifying:

Object: The question to answer here is, what the elements in the software are, for which quality is to be defined, measured or assessed. Example for objects are documentation, source code, requirements, design, build process, test suite, components, statements, classes, memory consumption, sub-routines, macros, functions, and procedures.

Purpose: This will depend on the model capabilities defined by the meta-model. The Quamoco meta-model v2 considers two different purposes: specification and evaluation of quality. Specification means that quality is described, but not quantified. For example, functionality is specified in ISO9126 as the aggregation of suitability, accuracy, interoperability, security, and functionality compliance. For the purpose of evaluation, quality is quantified, measured, and compared to defined assessment criteria to check the fulfillment of those criteria. For example, the response time of the application is less than 2 seconds for each query in a defined set of typical user queries.

Viewpoint: The perspective from which quality is described or evaluated is very important. According to the roles involved in the development process, the viewpoints to consider can be identified. Do we have specific requirements from the management? Which agreements do we have with the customer? Are there established good practices on the organization that we want to consider with the quality model?
Quality Focus: Which quality aspects do we want to cover with this model? The quality focus can include general high-level aspects, such as reliability, usa-

bility or maintainability, but also lower-level attributes or specific aspects can be considered, such as globalization, learnability, or training.

Context: When defining the context of the model, many things can be considered. The context may include the scope of application, which is described, for example, by answering the question: are there things which are mandatory within the organization? Context can also refer to the domain for which the quality model is intended, e.g., railway, medical devices, embedded systems, or information systems. The development methodology and practices can also be included in the context of the model, such as component-based software development, agile, open source, or custom development. Other things that are part of the context are the development paradigm (e.g., object-oriented, structured programming, event driven), or the programming language (C++, Java).

By defining a goal and using it as the foundation for quality model tailoring, we assure a goal-oriented adaptation approach, which is required by 0[4] Moreover, this increases efficiency[5] by focusing on the key elements of the adapted quality model goal.

8.2 Pre-Tailoring

As described in chapter Selection Criteria, all models will be created or can be characterized after their creation having in mind the quality goal parameters, which values constitute the premises of selection criteria.

The goal is used to look for a model and adapt it to the needs of the project or organization. This model, on which the model adaptation is based, is called reference model. Finding the right reference model consists in finding the model whose attributes best fit to the defined goal.

Once a reference model is chosen, elements that are not needed in the final model are discarded according to the logical consequences of the selection criteria. Only quality model components in the reference model that are relevant for the new model are taken. Unnecessary components are eliminated at the beginning. Sometimes, specific elements in the model can be partially reused but need some adjustments. Such elements should stay in the model and be

[4] 0 (R2) Goal Orientation – The adaptation of a base model should be driven by organizational needs and capabilities. In particular, organization-specific and project-specific software quality objectives should be considered.

[5] 0 (R3) Efficiency – This is concerned with the overhead (e.g., personnel, time, and budget) needed for adapting a quality model. Acceptable overhead would differ depending on the organizational level (e.g., more overhead will be allowed for adapting a quality model at the level of the whole organization, where such adaptation has a larger scope and is performed relatively rarely).

marked for detailed inspection and modification in the next step. The parts selected to remain in the model are the basis for further adjustments.

In **Table 1** we show an example. The left column show the goal parts and the right columns the consequential tasks performed during pre-tailoring.

Table 1: Example of using selection criteria for pre-tailoring a quality model

Quality Model Goal	Consequential Task on Reference Model
Object: source code	Delete all entities not belonging to source code (e.g., documentation, user manuals, test suite).
Purpose: evaluation	Include measures and evaluations.
Perspective: quality in use	Delete all other views (e.g., stakeholder satisfaction, technical issue). This means: keep the quality in use factor hierarchy and the elements in product factor hierarchy influencing it.
Quality focus: maintainability	Delete all other factor in the view. This means: keep maintainability in the quality in use factor hierarchy and delete factors that neither influence it nor refine it.
Context: java, automatic measurement tools	Delete all non-java elements and keep only measures which can be collected with automatic tools.

9 Iterative Model Changes

Adaptation Matrix (Basic operations)

After pre-tailoring the reference quality model, the model obtained might not be consistent or operational anymore. The removal of model components triggers further adaptation tasks. These tasks help to bring the model back to a consistent, operational state. Some adaptation tasks can be automated. Other tasks will require user interaction, as they are based on user decisions.

Accomplishing all adaptation tasks will lead to a consistent model customized to the user's needs. Elements are incrementally deleted (DEL), added (ADD), or modified (MOD) in the model until no further adaptation tasks are requested. The extent to which these operations are used depends on the suitability of the reference model.

New elements can be defined and added to the model or elements from models can be added to the adapted model. That is, individual elements from other models can be taken and reused in the new model.

To manage change throughout the adaptation process, we analyze the im-pact of changes on the quality model elements. Based on the structure pro-vided by the meta-model, a table is created that documents the consequences of the changes in one element on other elements, whether the task can be performed automatically or if it requires user interaction.

Based on the structure of the Quamoco meta-model v2, we created a matrix which summarizes the possible changes triggered by performing the basic operations add/delete/modify on the different quality model elements. The structure of the matrix for adaptation is shown in Table 2. The first column contains the elements that can be changed, associated to the different operations that can be effected on each of them.

In the second column, consequential tasks, the changes that must be performed to bring the model into a syntactically correct state are shown. In the third column, we specify if the changes from column 2 can be performed automatically, or if they must be performed manually. In the fourth, we describe the conditions that should be fulfilled after the tasks in column 2 were performed. In the last column, type of justification for documenting the changes is identified.

Table 2. Table of consequential changes.

Action	Consequential tasks (ToDos)	Can be performed Automatically (A) Manually (M)	Task was performed condition	Justification type: Consequential change (CC) Goal-triggered (G) User-based (U)
QMM ELEMENT task	**Task:** Affected QMM Element / attribute	M or A	Condition	CC or G or U

In **Table 3**, we show an example of how the adaptation matrix for the element *measure* and each of the possible operations: *add, delete* and *modify.* For each operation differing from the basic operations, a definition is additionally given, as well as the text that is shown to the user.

Table 3: Exemplary matrix for the element measure

Action	Consequential tasks (ToDos)*	Can be performed Automatically (A) Manually (M)	Task was performed condition	Justification type: Consequential change (CC) Goal-triggered User-based
Measure MOD	IF MOD(Name): For all associated factors, update the evaluations specifications using the measure. *"Check that the modified measure is correctly used in evaluation specification"*	M	-	CC
	IF MOD(Normalization): If change to 0 Decide: *"This measure is used for normalization, are you sure that you want to change it?" Yes/No* IF Decide == No THEN normalization = 1 IF Decide == Yes THEN normalization = 0 AND MOD(All Evaluations using the measure) *"Please choose another measure for normalization"*	M	No evaluation uses the measure for normalization anymore	CC
	IF MOD(MaxPoints) *"Recalculate evaluation specification of measure"*	M	-	CC
	IF MOD(Type) *"Update instrument / aggregation / evaluation related to measure"*	M	-	CC
	IF MOD(Measures) *"Update evaluation related to measure"*	M	-	CC
	IF MOD(Refined by): IF ADD(Refined by): MOD(Aggregation) OR ADD(Aggregation) to the refined measure. *"The refining measure needs to be used in an aggregation "* IF DEL(Refined by): MOD(Aggregation). *"Check use of measure in aggregation"*	M	-	CC
	IF MOD(Refines): MOD(Aggregation) OR ADD(Aggregation) to the refined measure. *"For aggregation in refined measure, check use of available measures"*	M	-	CC

Table 3 cont.: Exemplary matrix for the element measure

Action	Consequential tasks (ToDos)*	Can be performed Automatically (A) Manually (M)	Task was performed condition	Justification type: Consequential change (CC) Goal-triggered User-based
Measure ADD	MOD(Name)[6] *"Provide measurement name"*	M	Name ≠ 0	CC
	MOD(Description) *"Provide measurement description"*	M	Description ≠ 0	CC
	MOD(Type) *"Provide measurement type"*	M	Type ≠ 0	CC
	[optional] MOD(Normalization) *"If the measure is intended to be used for normalization, mark it as normalization measure."*	M	Normalization ≠ 0	CC
	IF Normalization==0, MOD(Measures) *"Define which factor is associated with this measure"*	M	(Measures ≠ 0) OR (Normalization =1)	CC
	MOD(Refines) *"If measure is part of an aggregation, specify which measure is refined"*	M	Refines ≠ 0	CC
	Decide: [ADD(Instrument)* XOR ADD(Aggregation)] *"This measure needs to be associated to an instrument or to an aggregation"* IF Decide == ADD(Instrument) IF Type == number OR percentage THEN ADD(metric-based or manual instrument) *"Provide a metric-based or manual instrument"* IF Type == points THEN ADD(manual instrument) *"Provide a manual instrument"* IF Type == findings THEN ADD(rule-based instrument) *"Provide a rule-based instrument"*	M	IF Decide == ADD(Instrument) THEN Check that an instrument is referenced by the measure IF Decide == ADD(Aggregation) THEN Check that an aggregation is referenced by the measure	CC

[6] This is a low level description. In this case an "empty" measure is added with blank attributes (name, description, etc.) that need to be modified.

Table 3 cont.: Exemplary matrix for the element measure

Action	Consequential tasks (ToDos)*	Can be performed Automatically (A) Manually (M)	Task was performed condition	Justification type: Consequential change (CC) Goal-triggered User-based
Measure DEL	AND was normalization measure THEN Decide: *"This measure is used for normalization, are you sure that you want to delete it?" Yes/No* IF Decide == No THEN measure is not deleted IF Decide == Yes THEN IF Evaluations using the measure == 0 complete measure deletion ELSE MOD(All Evaluations using the measure) *"Please re-work the evaluations using this measure before deleting it"*	M	No evaluation uses the measure anymore	CC
	AND was used by factor AND factor has no refinements THEN ADD(Measure) *"Provide a measure for the factor"*	M	Check that the factor has a reference to a measure	CC
	AND was used by factor AND factor has refinements THEN MOD(Evaluations that were using the measure) *"Check use of deleted measure in evaluation"*	M	No evaluation uses the measure anymore	CC

* "text to be shown to the user"

Set: a value has to be assigned to field/variable

Check references: references must be inspected

Decide: one of an n-tuple of options must be chosen

10 Reporting of Changes

What should be documented?

Goal of adapted quality model: The goal is a compact manageable description of the quality model. If during tailoring the appropriateness of the model with respect to the goal is put into question, the need to complete the model, so that the goal can be achieved, must be documented, as well as the fact that the model is not complete. This may happen if something cannot be modeled before studying or finding out more specific information.

Deviations from reference model: Operations on elements which were mandatory for the organization and are no longer being considered in the sub-organization or project and the reasons of non-inclusion. Eventually, an agreement should be signed that those changes are approved. The QAM responsible for the organizational model can take these changes and their justifications as a source for changes on the organization quality model.

Reporting changes is an important procedure to make decisions traceable. After performing an adaptation task, the changes can be documented as a history, showing a list of the performed actions, the affected elements, a justification, and a timestamp.

We group justifications according to the different tasks triggers:

Consequential changes, triggered by an adaptation task. For example, if a measure is deleted, it should be checked if it was used by a factor, and tailor the corresponding evaluation. The justification, as a history entry would look like this:

Performed action	Affected elements	Justification	Timestamp
Deleted measure from evaluation	Measure deleted: *m_xyz* from evaluation: *e_abc*	Measure (*m_xyz*) deleted from model, cannot be used	2011/11/22 12:23:34

Goal-related changes, triggered by the model goal parameters: For exam-ple, if the goal specifies that the model should be for used with C++, all in-struments which are tagged for other languages are deleted. The justifica-tion, as a history entry would look like this:

Performed action	Affected elements	Justification	Timestamp
Deleted instrument	Instrument deleted: *i_xyz*	Goal context = C++ Instrument *i_xyz* tagged as *java*	2011/11/22 12:23:34

User-based changes, triggered by user decisions: For example, the user deletes a tool because it is not available, and therefore cannot be used. The justification, as a history entry would look like this:

Performed action	Affected elements	Justification	Timestamp
Deleted tool	Tool deleted: *t_xyz*	Tool unavailable	2011/11/22 12:23:34

In order to make it easier for the user, we propose using some standards justifications which are regularly used:

- Elements relevant / not relevant for context
- Adapted to context
- Correction of error in model
- Minor change
- Tool not available
- Limited assessment
- Improvement of assessment

11 Quality Model Maintenance

During the application of the adaptation process, information is collected based on add, modify, and delete operations performed to adapt the reference model, i.e., the model that is the basis for the new adapted quality model.

This section shows how this context information, which could be (partially) automatically collected, can be used to improve/extend the reference model.

Available Information Resources

There are three potential information sources that can be used to provide feedback for the reference model: (A) the quality model goal definition, (B) the set of context tags of each model element, and (C) the justification provided when a model element is added, modified, or removed during adaptation.

(A) Quality model goal definition

Each Quamoco quality model is described by the goal parameters: object (e.g., *source code*), viewpoint (e.g., *management*), quality focus (e.g., *maintainability*), purpose (e.g., *evaluation*), and context (e.g., *embedded, avionic, programming language "c"*). Context information can be described as a set of attribute-value tuples (e.g., *"domain = avionics", "language = c"*).

(B) Set of context tags of each model element

Besides the context information provided as part of the quality model goal, each element of the quality model has the possibility to independently provide specific context information. This information is represented as a set of attribute-value tuples associated with the element.

(C) Justification for added/modify/delete

Justification: Operations performed with an element due to the changed context are enriched with a justification that motivates the addition, modification, or deletion of the element. Providing a justification has two advantages. First, it helps to decide whether an element should be included in the reference model. Second, it supports the Quamoco certification scenario H3 [14] because the rationale behind the adaptation can be evaluated.

Dealing with Context Tags of Elements during Model Adaptation

In order to reference to a specific set of attribute-value tuples, we will use in this section the following abbreviations:

C_ref_g: Context information of the reference quality model provided in the quality model goal

C_ref_e: Context information of a specific element e in the reference quality model

C_new_g: Context information of the new quality model provided in the quality model goal

C_new_e: Context information of a specific element e in the new, adapted quality model

C_diff_e (= C_new - C_ref): Context information that differentiate the context of the new quality model and the existing context information of the element.

Example 11.1: Assume that we have a reference model for embedded systems from the avionics domain developed for the programming language c.

C_ref_g = {"type = embedded", "language = c", "domain = avionics"}.

An element will then be tagged e.g., with

C_ref_e = {"type = embedded", "language = c", "domain = avionics"}.

The new model will also be used for embedded systems but in the automotive domain and it has to consider also c++ code and should be focused on a multi core environment.

C_new_g = {"type = embedded", "language = c, c++", "domain = automotive", "hardware = multi_core"}

This results in a difference between the two contexts represented by

C_diff_e = { "language = c++", "domain = automotive", "hardware = multi_core" }

STEP1: Keep existing context information from the reference model

The existing context tags of all elements in the reference model are kept in the new model.

Remark: Typically, the tags of a reference model element are equal to the context information of the quality model goal (C_ref_g) as in our example: C_ref_e = {"type = embedded", "language = c", "domain = avionic"}. But the reference model may also contain elements independent of specific context characteristics and, therefore, tagged differently (e.g., the factor as "Modularization" is independent of the domain).

STEP2: Extension of context information based on changes in the context

Adaptations of elements based on changes in the context are performed on the reference model motivated by the context of the new quality model. This means the operations are not a result in a changed object, viewpoint, quality focus, or (application) purpose.

STEP2.1a: Automatic tagging approach (no effort during quality model adaptation)

The automatic approach requires no user interaction to extend the context information during the adaptation. All modifications in the context information are done automatically in the background. This provides the advantage that model adaptation can be performed more efficiently. We consider it as the default case.

ADD: Element added due to changes in the context are tagged additionally with ADD_C_diff.

For instance, based on example 11.1, C_new_e = {"language = ADD_c++", "domain = ADD_automotive", "hardware = ADD_multi_core"}.

MOD: Elements modified due to changes in the context are tagged additionally with MOD_C_diff.

For instance, based on example.11.1, C_new_e = {"language = MOD_c++", "domain = MOD_automotive", "hardware = MOD_multi_core"}.

DEL: Elements removed due to changes in the context are tagged additionally with NOT_C_diff.

For instance, based on example 11.1, C_new_e = {"language = DEL_c++", "domain = DEL_automotive", "hardware = DEL_multi_core"}.

STEP2.1b: Semi-automatic tagging approach (effort during quality model adaptation)

The semi-automatic approach requires explicit user interaction during model adaptation to extend the context information of the elements. However, it supports the person performing the adaptation with an automatically generated list with possible context factors to select from. This approach provides advantages during the reference model maintenance but increases the model adaptation effort.

ADD: If an element of the reference model is added based on changes in the context, the person conducting the adaptation has to decide which part of the changed context (C_diff) is responsible for the inclusion of the new element (C_diff_part). The element added is then tagged additionally with ADD_C_diff_part.

Example: If a factor relevant in the new model is not part of the reference model due to the additional development concepts that have to be considered in the new model (e.g., classes in c++), the relevant C_diff_part would be {"language = c++"} and would be added to the context information of the element as {"language = ADD_c++"}.

MOD: If an element of the reference model is modified based on changes in the context, the person conducting the adaptation has to decide which part of the changed context (C_diff) is responsible for the modification of the element (C_diff_part). The modified element is then tagged additionally with MOD_C_diff_part.

Example: If a factor in the reference model (from the aerospace domain) is justified by an aerospace-specific standard, the standard have to be replaced in the new model with the automotive-relevant standard. This means the relevant C_diff_part is {"domain = automotive"} and have to be added to the context information of the element as {"domain = MOD_automotive"}.

DEL: If an element of the reference model is removed based on changes in the context, the person conducting the adaptation has to decide which part of the changed context (C_diff) is responsible for the deletion of the element (C_diff_part). The element removed is then tagged additionally with NOT_C_diff_part.

Example: If a factor as "Redundancy of critical components" in the reference model from the aerospace domain is not relevant in the new quality model due

the changed domain (automotive), the relevant C_diff_part would be {"domain = automotive"} and would be added to the context information of the element as {"domain = DEL_automotive"}.

STEP2.2: Provide justification for adaptation (e.g., for certification)

The person performing the adaptation operation can optionally provide a justification, i.e., why the operation was reasonable.

Example: If the factor "appropriate use of inheritance concept" is added, a justification may be "reference model do not consider object-oriented concepts".

Using the Information during Reference Model Maintenance

During the maintenance of the reference model, one can make usage of information collected during its adaptation for other applications. In Quamoco, we distinguish between four different maintenance scenarios for quality models: *preventive*, *corrective*, *perfective*, and *adaptive* maintenance [14]. The feedback from the adaptation can be used in first line for perfective and adaptive maintenance:

- *Perfective maintenance* is the proactive modification of the quality model performed in the same application context to meet new, improved requirements.
- *Adaptive maintenance* is the proactive modification of the quality model performed to keep it usable in a changed context where it should be applied.

In general, we assume that each adaptation operation (i.e., addition, modification, or deletion) can be explained by at least one change in the quality goal parameters. These goal parameters can be separated in two groups:

- The *first group* describing the application purpose, object, (general) viewpoint, and the quality focus. They influence typically the need and existence of elements in the model such as specific factors, measures and evaluation rules.
- The *second group* collects all context characteristics relevant for the quality model, they primarily influence the importance of specific factors, their relevance, how they can be measured and how they influence other factors.

Therefore, the results of the adaptation operations can be used in the maintenance of reference quality models in different ways depending on its motivation by the first or second group of goal parameters:

- The results of the adaption operations in the first case (*object, purpose, quality focus, and viewpoint*) are candidates to be included in the existing reference model to broaden the scope of applicability without invalidating existing knowledge contained in the reference model. This is part of perfective maintenance activities for the reference model.

Example: If the reference model was defined only for the object "source code" and was extended during the adaptation for the object "design" the added elements are candidates to be included in the reference model to make it applicable for both "design" and "source code".

- The results of the adaption operations in the second case (*context*) have to be analyzed in more detail because they are motivated by a change in the context. The adaptation operation may be caused by a *refinement* of the context (i.e., added attribute-value tuple), and *extension* of the context (i.e., added attribute-value tuple for an existing attribute), or an *adjustment* in the context (i.e., changed attribute-value tuple). Depending on the type of context change, the information can support different types of adaptive maintenance.

Example: Assume, we have a model for embedded systems from the avionics domain developed for the programming language c: C_ref = {"type = embedded", "language = c", "domain = avionics"}. The new model will also be used for embedded systems but in the automotive domain and it have to consider also c++ code and should be focused on a multi core environment: C_new = {"type = embedded", "language = c, c++", "domain = automotive", "hardware = multi_core"}. In this case, "language = c++" is an extension, "hardware = multi_core" is a refinement, and "domain = automotive" describes an adjusted context.

- The results of adaptation operations motivated by **context extensions** can be used to extend the context of the reference model (see **Figure 6**(1)).

    - **ADD:** An element added due to a context extension is a candidate to be included when the context of the reference model should be extended.

  Example: C++-specific measures included in an adapted model to deal with the programming languages c and c++ can be included in the reference

model, if the reference model should be extended from the context "language = c" to "language = c, c++"

- **MOD:** The modification of an element due to a context extension should be evaluated to be included in the reference model. Usually, such modifications are generalizations to make the element fit an extended context.

- **DEL:** The deletion of an element due to a context extension is very unusual. There is no reasonable example known to the authors.

- The results of adaptation operations motivated by **context refinements** can be used to provide optional parts for the reference model that can be used if required (*e.g., an optional part for multi core systems in an embedded quality model,* see **Figure 6** (2)). Alternatively, an addition, refined reference quality model can be defined, keeping the original reference model unchanged. However, this would increase the number of reference quality models to maintain. Elements that are added, deleted, or modified as a result of a context refinement are candidates to be included in the reference model.

    - **ADD:** (A) If the element seems generally applicable (i.e., it is estimated to be relevant for more than 80% of the reference model applications) it can be included directly in the reference model with the standard context tags of the reference model relevant for the element. This can be considered as an exception since it is perfective maintenance. (B) If the element seems to be not generally applicable (i.e., it is estimated to be relevant for less than 80% of the reference model applications), it should be included as an *optional* element. This should be marked by keeping the tagging used in the adapted quality model.

Note: If automatic tagging is used, unnecessary context tags may have to be removed (doing the step usually performed during semi-automatic tagging).

Example: A factor as "assure load balancing between CPU cores" that was added in the new, adapted model due to the context extension "multi_core" is not relevant for more than 80% of the embedded systems, which are addressed by the reference model. Therefore, it would be added with the attribute value pairs {"type = embedded", "hardware = ADD_multi_core"}, where "hardware = multi_core" is not part of the context definition of the quality goal of the reference model (marking this element as optional).

- **MOD:** The approach is the same as described for added elements. The only difference is that in case (B) two instances of the element exist in the reference model. The original and the modified with the additional context tag.

Example: An evaluation that was modified in the new, adapted model due to the context extension "multi_core" is not relevant for more than 80% of the embedded systems, which are addressed by the reference model. Therefore, it would be added as an optional replacement for the original element applicable in the case that multi core systems are considered. The attribute value pairs of the added element would be {"type = embedded", "hardware = MOD_multi_core"}, where "hardware = multi_core" is not part of the context definition of the quality goal of the reference model (marking this element as optional replacement for the original element).

- **DEL:** The approach is the same as described for added or modified elements. The only difference is that in case (B) the existing element in the reference model is marked as optional in the case of the refined context (i.e., it is replaced by the element from the adapted quality model).

Example: A measure that was removed in the new, adapted model because it is not reasonable in the context "multi_core" is relevant for more than 80% of the embedded systems, which are addressed by the reference model. Therefore, it would be kept in the reference model with the note that should be deleted in the case that multi core systems are considered. The attribute value pairs of the element would be {"type = embedded", "hardware = DEL_multi_core"}, where "hardware = multi_core" is not part of the context definition of the quality goal of the reference model (marking this element as removable for a specific context).

- The results of the adaptation operation motivated by **context adjustments** can support the building of a more abstract reference model capturing the communities and a sister reference model with a context different from the current reference model (*e.g., building a general embedded model and an automotive embedded model based on an existing reference model for avionics embedded systems and the results of its adaption for the automotive domain,* see **Figure 6**(3)).

Remark: This kind of adaptation based on context adjustment implicitly builds sister quality models under a virtual reference model.

- **ADD:** Elements that are added as a result of a context adjustment are indicators for differences caused by the differences in the contexts. They will usually not be part of the abstract reference model but included in the new reference model for the adjusted context. The elements should be marked by keeping the tagging used in the adapted quality model removing the "ADD" prefix.

Example: A measure was added in the adapted model due to the context adjustment "automotive" (e.g., since it is requested by a domain-specific standard). If the measure is not relevant for more than 80% of the embedded systems, which are addressed by the new abstract reference model, it would not be included in the reference model. Instead, it would be added as an element in the automotive reference model with the attribute-value pairs {"type = embedded", "domain = automotive", "language=c"}. The avionics reference model will stay unchanged.

- **MOD:** Elements that are modified as a result of a context adjustment are also indicators for differences caused by the differences in the context. They will usually not be part of the abstract reference model but included in both the new reference model for the adjusted context and the old existing reference model. The elements should be marked in the new reference model by using the tagging in the adapted quality model removing the "MOD" prefix. In the old reference model, the element should be marked with the context tag with the old value that was changed in the adapted model with the new value including the "MOD" tag.

Example: A factor in the reference model (from the aerospace domain) is motivated by an aerospace-specific standard and the standard was replaced in the new model with the automotive-relevant standard. This means that the factor itself can be included in the abstract model but the part that are specific for the domain (the standard motivating the factor) have to be provided by the domain-specific refined reference models, independently. The element would be tagged with the respective context in each reference model (e.g., {"type=embedded", "domain = avionic"} in the old reference model and {"type=embedded", "domain = automotive"} in the new automotive reference model). In the abstract reference model an abstraction can be included with the tag {"type=embedded"}.

- **DEL:** Elements that are removed as a result of a context adjustment are indicators for differences caused by the differences in the contexts. They will usually neither be part of the abstract reference model nor of the new reference model for the adjusted context. In the old reference model, the element should be marked with the

context tag with the old value that was changed in the adapted model with the new value including the "DEL" tag.

Example: The factor "Redundancy of critical components" in the reference model from the aerospace domain was considered as not relevant in the new quality model due the changed domain (automotive). If the factor is not relevant for more than 80% of the embedded systems, which are addressed by the new abstract reference model, it would not be included in the reference model. Instead, it would be kept as an element in the avionics reference model with the attribute-value pairs {"type = embedded", "domain = avionic"}.

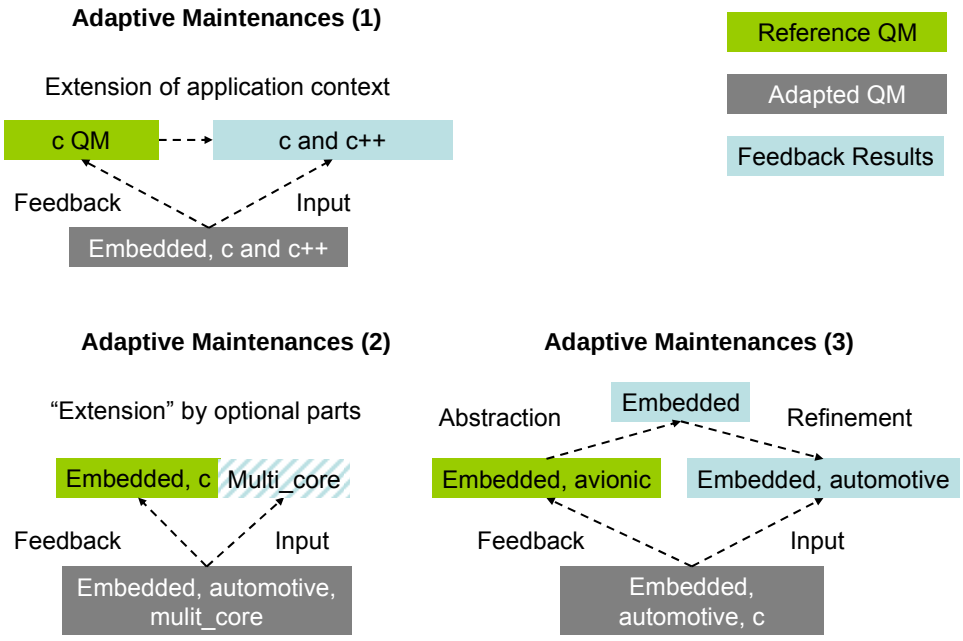


Figure 6: Adaptation process feedback can be used for different maintenance activities in the reference model.

Dealing with changes in the Reference Model

Another important part of model maintenance is dealing with changes in the reference model. Namely, when the adapted quality model has to be based on or complaint to the reference model, the adapted model needs to be re-adjusted.

There are two possible information sources to be used for re-working or updating the adapted model:

– If the provider of the reference model documents the changes, it is possible to directly introduce them into the adapted quality model.

– If that information is not provided, it the adaptation could be repeated using the adaptation history.

12 Interface to Further Quamoco Deliverables

12.1 Meta-Model

The quality meta-model defines the underlying structure to which all Quamoco quality models have to be conformant [17]. This means that all adapted quality models must have this structure. The quality meta-model assumed by the adaptation process presented here is shown in **Figure 8**. It evolved from the quality meta-model v2 during the second project iteration.

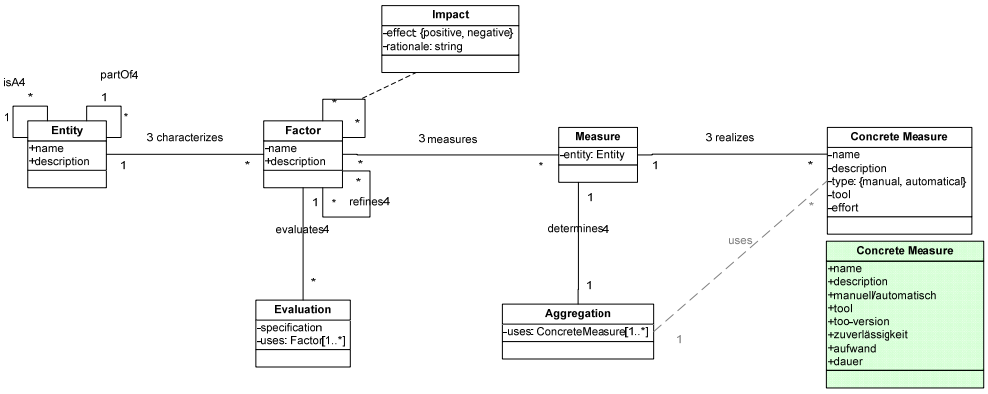


Figure 7: Quality Meta-Model Version 2 as described in [24].

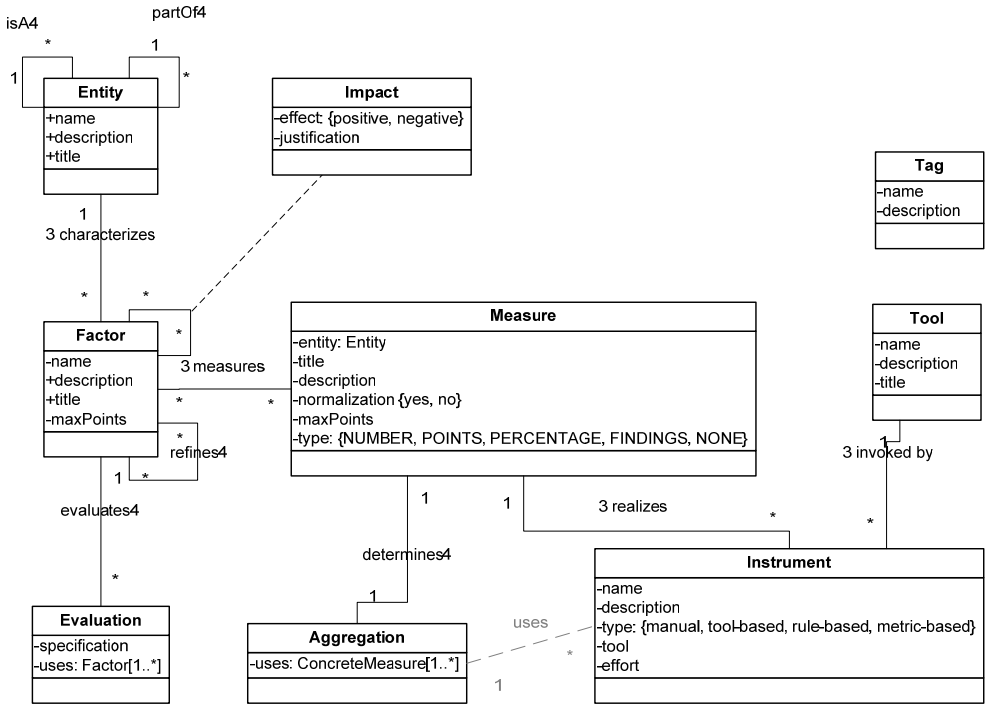


Figure 8: Quality Meta-Model Version 2.5, in which the adaptation process v2 is based.

Organization quality models may include multi-language implementations of measures and therefore many evaluations.

Project models intended to actually be applied (executables) must have only one evaluation associated to each factor.

12.2 Base Model

"The Quamoco base model is:

– Intended to specify basic quality properties and how to collect and evaluate data for all kinds of software systems;

– Structured into technology-specific modules;

– Intended to provide a comprehensive off-the-shelf quality model that can be applied instantly to assess the quality of software products." [27, pg. 5]

Assumed Modeling Guidelines

In order to be able to use the adaptation process on the base model, we assume a set of modeling rules on how the modeling should be done.

G1 For modeling entities we assume a structure showing ***product-parts, use case*** and ***stakeholders' entities*** as described in the help of Quamoco Quality Model Editor v2.3 [24, Entity.html]:

> "***Product-Parts:*** *They describe a part of a product. If they are characterized by a factor, a Product-Factor results.*
>
> ***Use Cases:*** *A use case describes an activity conducted with the system by a stakeholder. If they are characterized by a factor, a Use Case-Factor results.*
>
> ***Stakeholders:*** *A stakeholder describes a role that is interested in the product. If they are characterized by a factor, a Satisfaction-Factor results."*

Examples for ***product-parts*** can be syntactical elements of source code such as *source code comment*, *HTTP statement*, and *version information*. Exemplary ***use cases*** can be *execute*, *system test*, and *interaction.* ***Stakeholders*** can be, for instance, *architect*, *developer*, and *operator*.

G2 For modeling factors we assume parallel hierarchies: a hierarchy describing product intrinsic factors (essential) and hierarchies describing perspectives. The current version of the base model contains five factors hierarchies:

1. Product Factor *[Property @Product Part]*:

> *"Product factors are properties of specific parts of the software product. These parts are specified as artifacts or product entities, such as, Conciseness @Identifier or Appropriateness @Comment"* [24, FactorBase.html]

2. Product Quality Attribute (ISO 25010) *[Quality @Product]*:

> *"Product quality attributes define one way to decompose the abstract concept software quality. These quality attributes, as given in ISO 25010, relate to the quality of the product without explicitly considering its use. These quality attributes are colloquially called -ilities, because they contain, for example, reliability or maintainability. In the standard, the top level attributes are refined by so-called quality characteristics."* [24, FactorBase.html]

3. Quality in Use Attribute (ISO 25010) *[Quality @Use Case]*:

> *"Quality-in-use attributes define a way to decompose the abstract concept software quality. These quality attributes are defined in the ISO 25010 as description of the quality in its various forms of usage. Examples are efficiency or effectiveness. To be more precise, we add the activity that is characterized by the attribute as entity to the factor. Activities like maintenance, program comprehension, modification, or testing, which can be decomposed in their respective sub-activities, provide a means to model software development cost structures."* [24, FactorBase.html]

4. Stakeholder Satisfaction *[Satisfaction @Stakeholder]*:

> *"On a high level of abstraction, quality is the satisfaction of stakeholders. This can also be modeled in the base model. For this, we use the stakeholder as an entity."* [24, FactorBase.html]

5. Technical Issue *[TIC]*:

> *"Technical issues are also a way to decompose the abstract concept of quality. This is a technical view, which assigns problems to areas such as memory or declaration.* ***TIC factors do not characterize specific entities.****"* [24, FactorBase.html]

G3 For the adaptation process, the ***viewpoints*** in quality models goals are given by the root-node in the hierarchies parallel to the product hierarchy.

G4 *Impacts* **as described in the quality meta-model are** only **from product hierarchy into the other hierarchies (Figure 9)**

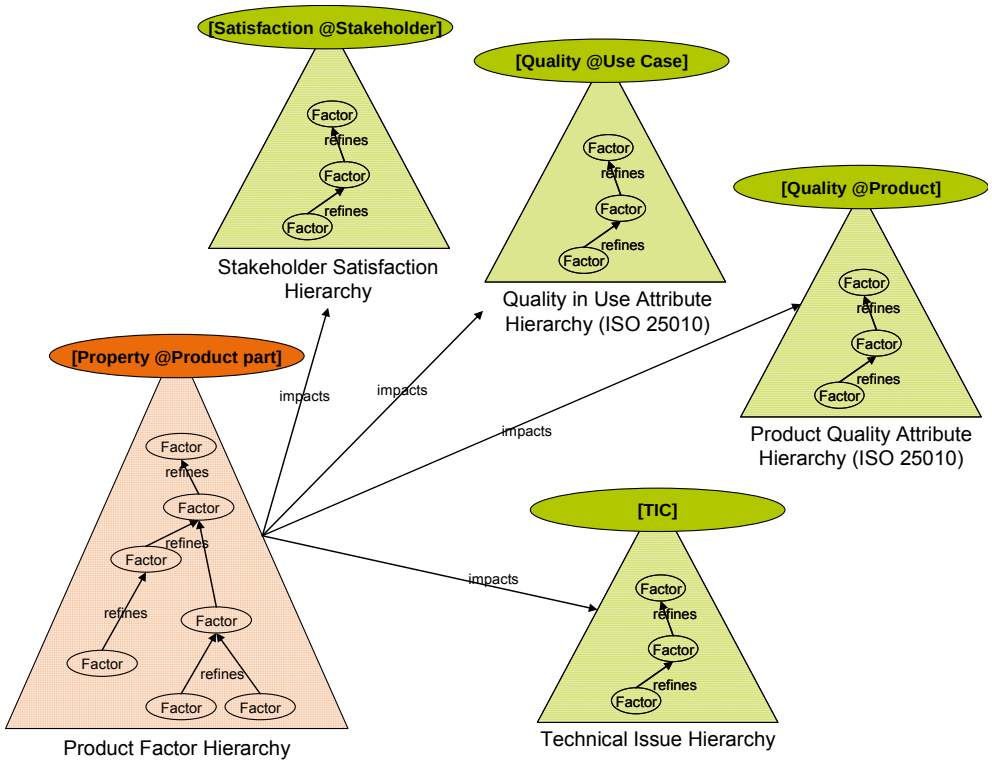


Figure 9: Assumed factor hierarchies in base model.

Modularization Concepts and Tagging

Need and Advantages of Modularization for the Base Model

The Base Model aims for including *basic quality properties,* supporting *all kinds of software systems,* and being *a comprehensive off-the-shelf quality model that can be applied instantly.*

This means that it should include different implementations of factors of different *technologies, methodologies,* and *paradigms.* This results in the need of specializing factors, measures, and evaluations for these concepts.

For the sake of an easier modeling and maintenance of the base model and in order to be able to *instantly apply* it (without adapting it), a modularization concept was specially developed.

The module hierarchy describes in a *root module* the set of all *basic quality properties*, not considering how the specialize on concrete technologies. The other modules in the model implement then their concrete instance of the abstract concepts on the root module, as in the example shown in Figure 10.

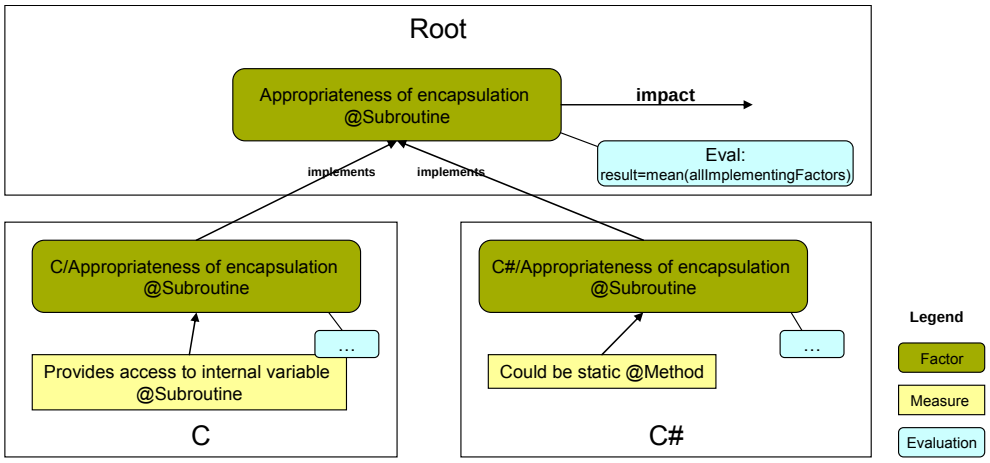


Figure 10: Example of implementations of factors for specific programming languages.

Mapping of Base Model into Meta-Model Structure

The implementation of the base model as modules means that the model is spread in many resources. For adapting a quality model, however, it should be implemented in one resource. We agreed to create a mapping from the base model as modules into one resource in order to be able to apply the adaptation process on it. For this, we need to assemble all elements which are scattered in many modules into a unique module.

We need to perform the following kind of transformations:

Factors

Factor implementation with evaluation

All implementing factors disappear

Only one factor is kept with a unique name: property @entity

The factor is connected to the evaluations of the of the disappearing implementing factors

The evaluations receive a tag according to the module in which they were included

No tags are given for root elements

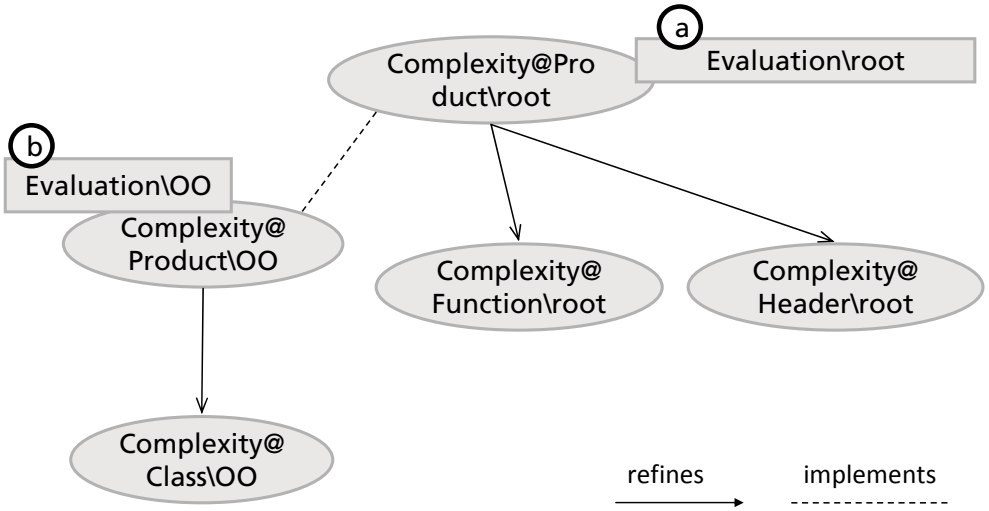


Figure 11: Factor implementation with evaluation according to the module-based structure

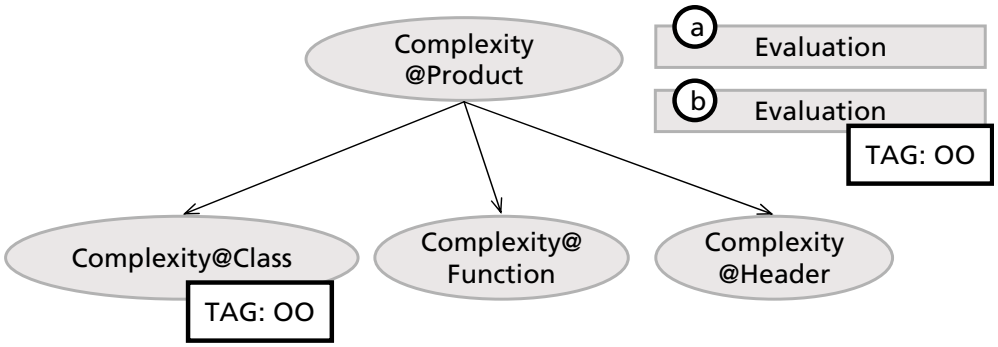


Figure 12: Factor implementation with evaluation according to the concept using one module and tags

Factor implementation without evaluation

All implementing factors disappear

Only one factor is kept with a unique name: property @entity

The factor is connected to the evaluation in root

The evaluation receive a tag “All”, meaning that it is a universal evaluation, and tags according to the modules in which the implementing factors were included

No tags are given for other root elements

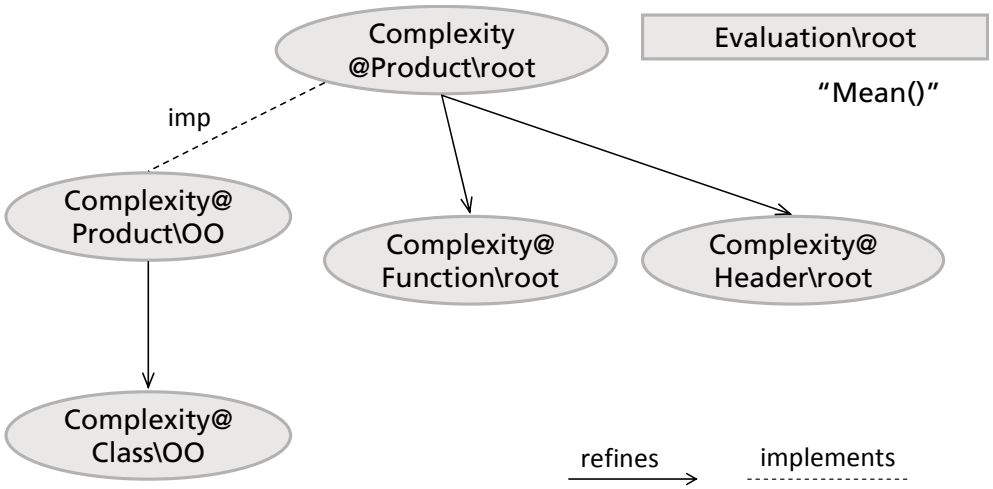


Figure 13: Factor implementation without evaluation according to the module-based structure.

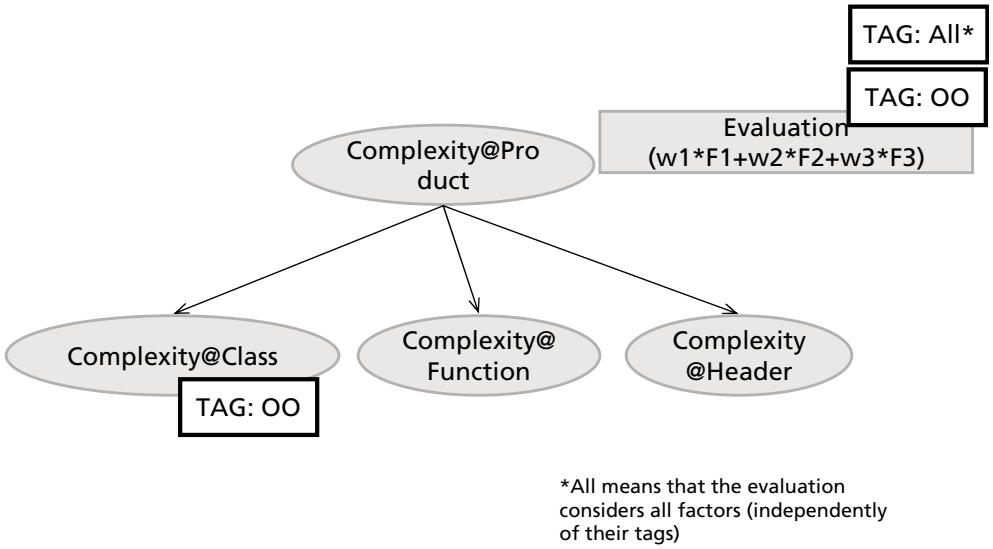


Figure 14: Factor implementation without evaluation according to the concept using one module and tags.

Measures (A) with multiple evaluations

Implemented measures disappear

For each implementing measure, a measure is created and tagged with the module's name to which it belonged. They are connected to the factor that was measured by the disappearing implemented measure.

If there different target values for the different technologies, separated evaluations are created and tagged according to the respective measures.

Measures (B) with single evaluation

Implemented measure is kept

A technology-independent aggregation is created.

For all the measures used by the aggregation, different technology specific instruments are defined and tagged according to the respective technology.

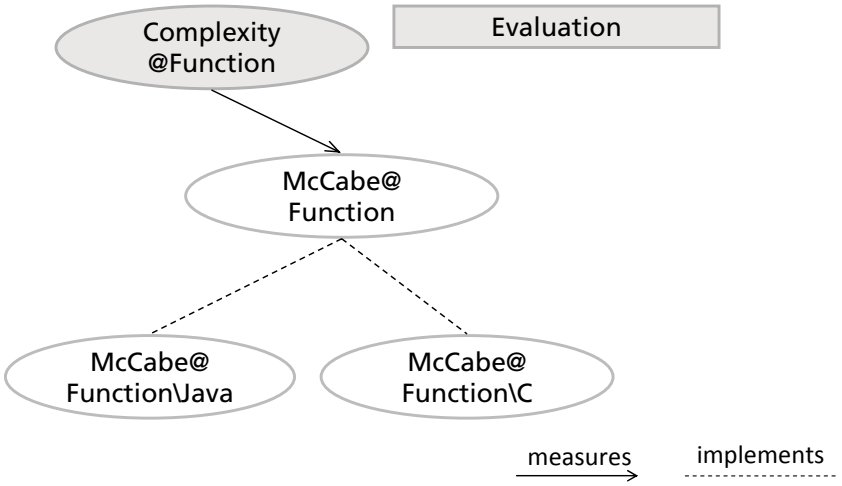


Figure 15: Measure implementation according to the module-based structure.

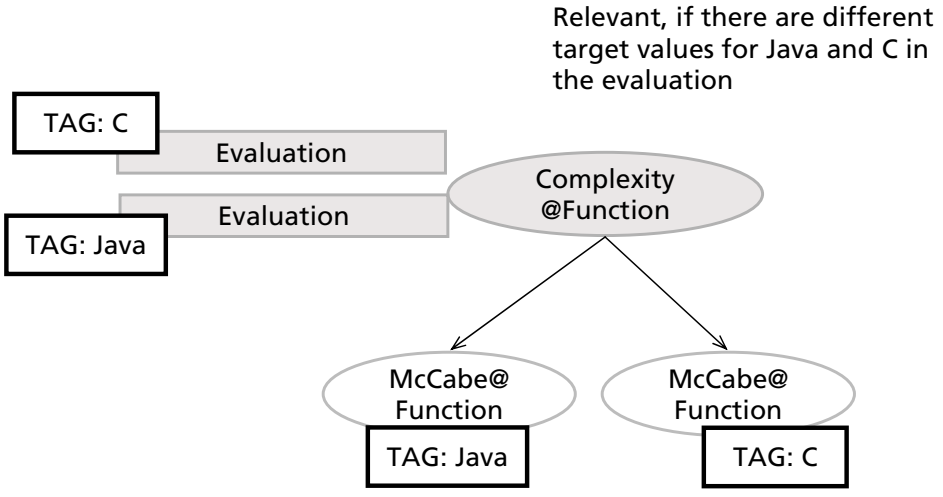


Figure 16: Measure implementation according to the concept using one module and tags, with multiple evaluations (A).

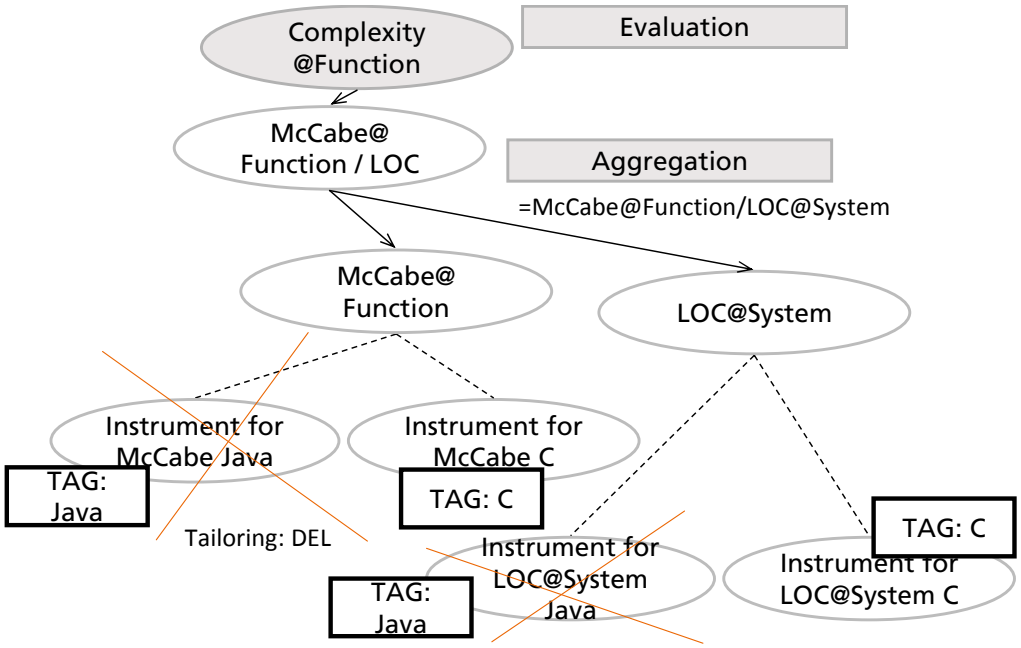
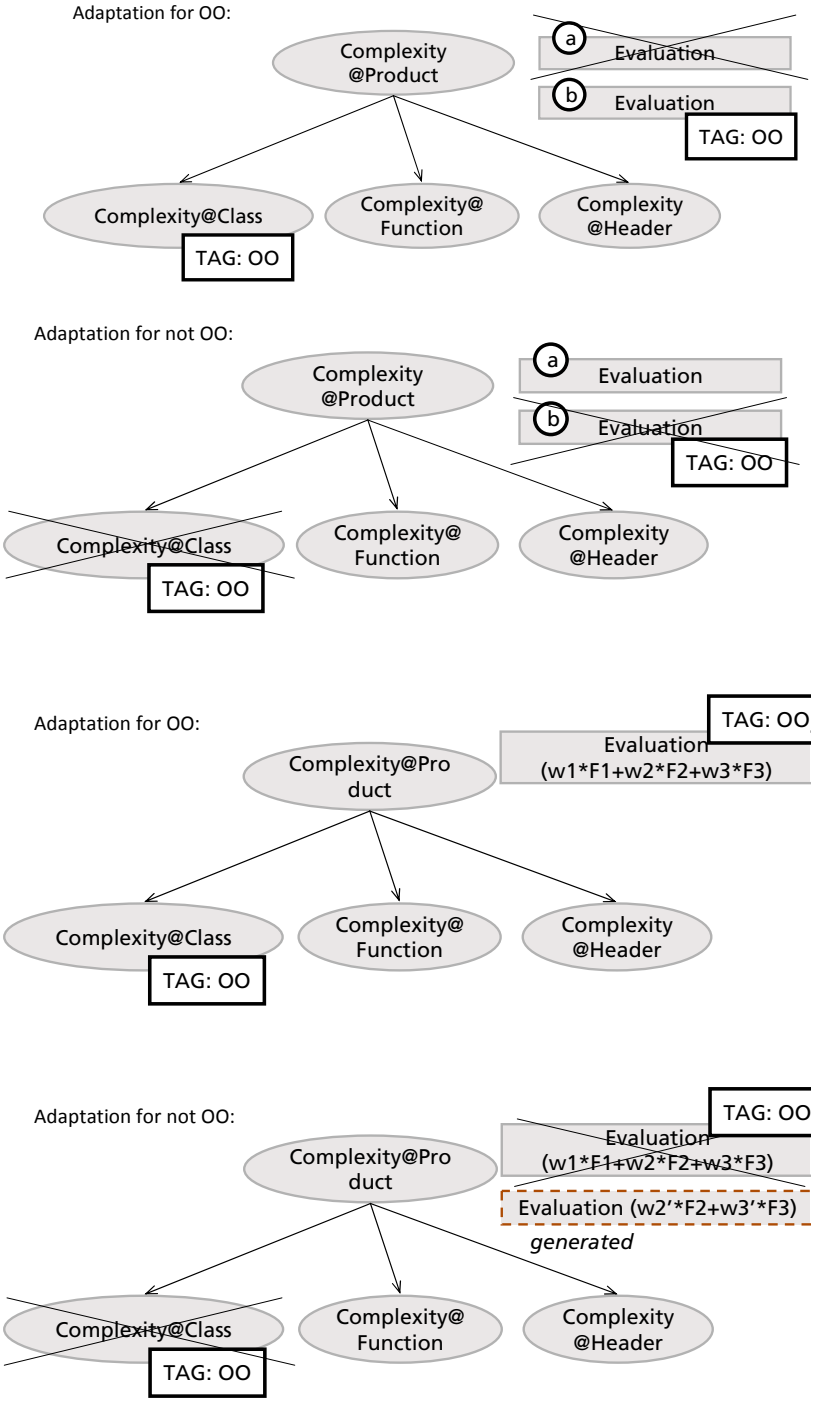


Figure 17: Measure implementation according to the concept using one module and tags, using a single evaluation.

Adaptations using tags



12.3 Assessment Method

When adapting a quality model, it may happen that an evaluation specification becomes invalid, due to the changes performed.

Changes invalidating "Evaluation Functions": Evaluation functions are used to map a measure's value into points for the factor being measured. In this case, if the measures used by the evaluation function change, or are deleted, the evaluation specification must be recalculated.

Changes invalidating "Factors Aggregations": Factor aggregations are used to aggregate points of refining or influencing factors. This kind of evaluation specification becomes invalid has to be recalculated by the following changes:

Adding a sub-factor/impact

Removing a sub-factor/impact

Changing the MaxPoints of the factor

Changing the ranking of sub-factors/impacts

Changing the Contribution Points of sub-factors/impacts

Changing the effect of impacts

Depending on the affected element the recalculation consists in:

- For a new factor, the Contribution Points or the Ranking must be changed
- For changed Contribution Points, the factor's MaxPoints are recalculated
- For changed Ranking, the sub-factors/impacts' Contribution Points are re-calculated
- For changed MaxPoints, the sub-factors/impacts' Contribution Points are recalculated

12.4 Requirements Method

When requirement factors are iteratively constructed from the product factors in an existing quality model, it is equivalent to a special pre-tailoring of an existing quality model.

This consists in selecting the relevant parts of a quality model from which the requirements are derived: suitable factors, sub-factors, or measures; and after-

wards in defining the evaluations that describe how requirement fulfillment is calculated from product factor fulfillment.

This interface with the adaptation process can be seen when using the tool implementing the process. A detailed example will be given with the tool documentation.

References

1. Andersson, T.; Eriksson, I. V. (1996): Modeling the quality needs of an organization's software. In: HICSS '96: Proceedings of the 29th Hawaii International Conference on System Sciences Volume 4: Organizational Systems and Technology. Washington, DC, USA: IEEE Computer Society, p. 139.

2. Andreou, A. S.; Tziakouris, M. (2007): A quality framework for developing and evaluating original software components. In: Inf. Softw. Technol., vol. 49, no. 2, pp. 122-141.

3. Basili, V.; Weiss, D. (1984): A methodology for collecting valid software engineering data. In: IEEE Transactions on Software Engineering, vol. 10(3), pp. 728-738.

4. Behkamal, B.; Kahani, M.; Akbari, M. K. (2009): Customizing ISO 9126 quality model for evaluation of B2B applications. In: Inf. Softw. Technol., vol. 51, no. 3, pp. 599-609.

5. Bianchi, A.; Caivano, D.; Visaggio, G. (2002): Quality models reuse: experimentation on field. In: COMPSAC '02: Proceedings of the 26th International Computer Software and Applications Conference on Prolonging Software Life: Development and Redevelopment. Washington, DC, USA: IEEE Com-puter Society, pp. 535-540.

6. Calero, C.; Cachero, C.; Córdoba, J.; Moraga, M. (2007): PQM vs. BPQM: Studying the tailoring of a general quality model to a specific domain. In: Advances in Conceptual Modeling - Foundations and Applications, pp. 192-201.

7. Chidamber, S. R.; Kemerer, C. F. (1994): A metrics suite for object oriented design. In: Software Engineering, IEEE Transactions on, vol. 20, no. 6, pp. 476-493.

8. Deissenböck, F.; Herrmannsdörfer, M.; Wagner, S. (2009): Quality metamodel (QuaMoCo WP1.3, Deliverable #6).

9. E DIN IEC 61508-3:2006-07: Functional safety of electrical/electronic/programmable electronic safety-related systems.

10. EN 60601-1-4:1999: Medical electrical equipment - Part 1-4: General requirements for safety - Collateral standard: Programmable electrical medi-

cal systems.

11. Eriksson, I.; Törn, A. (1991): A model for IS quality. In: Softw. Eng. J., vol. 6, no. 4, pp. 152-158.

12. Fitzgerald, B.; Russo, N. L.; O'Kane, T. (2003): Software development method tailoring at Motorola. In: Commun. ACM, vol. 46, no. 4, pp. 64-70.

13. Franch, X.; Carvallo, J. P. (2003): Using quality models in software package selection. In: IEEE Softw., vol. 20, no. 1, pp. 34-41.

14. Gruber, H.; Dautovic, A.; Lochmann, K. (2009): Application scenarios for QMs (QuaMoCo WP2.1, Deliverable #11).

15. Horgan, G.; Khaddaj, S. (2009): Use of an adaptable quality model approach in a production support environment. In: Journal of Systems and Software, vol. 82, no. 4, pp. 730-738.

16. ISO/IEC 9126-1:2001: Software Engineering - Product Quality - Part 1: Quality Model.

17. Kläs, M.; Lampasona, C.; Nunnenmacher, S.; Wagner, S.; Herrmannsdörfer, M.; Lochmann, K. (2010): How to Evaluate Meta-Models for Software Quality? In: Abran, Alain (Ed.) ; Büren, Günter (Ed.) ; Dumke, Reiner (Ed.) ; Cuadrado-Gallego, Juan J. (Ed.) ; Münch, Jürgen (Ed.): Applied Software Measurement. Proceedings of the joined International Con-ferences on Software Measurement. IWSM/MetriKon/Mensura 2010, pp. 443-462.

18. Kläs, M.; Münch, J. (2008): Balancing upfront definition and customization of quality models. In: Softwarequalitätsmodellierung und -bewertung SQMB'08, pp. 26-30.

19. Lampasona, C. (2010): Quamoco WP 3.1 Variability and Tailoring Mechanisms for Quality Models. Deliverable #17.

20. Mayr, A.; Plösch, R.; Körner, C.; Dautovic, A.; Wagner, S.; Trendowicz, A. (2009): Requirements for quality models (QuaMoCo WP1.1, Deliverable #4).

21. McCall, J. A.; Richards, P. K.; Walters, G. F. (1977): Factors in software quality. Concept and definitions of software quality: Final technical report. Springfield: National Technical Information Service (NTIS), Reportnr. RADC-TR-77-369 (I, II and III).

22. Ocampo, A. (2009): The REMIS approach to rationale-based support for process model evolution. Univ., Diss.--Kaiserslautern, 2008. Stuttgart:

Fraunhofer IRB-Verl. (PhD Theses in Experimental Software Engineering, 25).

23. Plösch, R.; Gruber, H.; Körner, C.; Pomberger, G.; Schiffer, S.: Adapting quality models for assessments - Concepts and tool support. In: Proceedings of SQMB 2010 Workshop, held in conjunction with SE 2010 conference, February 22nd 2010, Paderborn, Germany, published as Technical Report TUM-I1001 of the TUM.

24. Quamoco Quality Model Editor v2.3 (11/2010).

25. Sharma, A.; Kumar, R.; Grover, P. S. (2008): Estimation of quality for software components: an empirical approach. In: SIGSOFT Softw. Eng. Notes, vol. 33, no. 6, pp. 1-10.

26. Soto, M.; Izquierdo-Cortazar, D.; Ciolkowski, M. (2009): Measuring the per-formance of open source development communities: The QualOSS approach. In: Büren, Günter; Dumke, Reiner R. (Eds.): MetriKon 2009 - Praxis der Software-Messung. Tagungsband des DASMA Software Metrik Kongresses MetriKon 2009, 19. - 20. November 2009, Kaiserslautern. Aachen: ShakerMagdeburger Schriften zum empirischen Software Engineering, pp. 219-233.

27. Winter, S.; Göb, A., Trendowicz, A.; Wagner, S.; Kläs, M.; Körner, Ch. (2010): Quamoco WP 1.5. Evaluation of the Base Model. Deliverable #44.

Appendix A Adaptation Decisions and Adaptation Tasks

In chapter 9 Iterative Model Changes, we provided an example for an adaptation task. In this appendix, we provided the complete adaptation matrix for meta-model v2.5.

Action	Consequential tasks (ToDos)	Can be performed Automatically (A) Manually (M)	Task was performed condition	Justification type: Consequential change (CC) Goal-triggered User-based
Measure MOD	IF MOD(Name): For all associated factors, update the evaluations specifications using the measure. *"Check that the modified measure is correctly used in evaluation specification"*	M	-	CC
	IF MOD(Normalization): If change to 0 Decide: *"This measure is used for normalization, are you sure that you want to change it?" Yes/No* IF Decide == No THEN normalization = 1 IF Decide == Yes THEN normalization = 0 AND MOD(All Evaluations using the measure) *"Please choose another measure for normalization"*	M	No evaluation uses the measure for normalization anymore	CC
	IF MOD(MaxPoints) *"Recalculate evaluation specification of measure"*	M	-	CC
	IF MOD(Type) *"Update instrument / aggregation / evaluation related to measure"*	M	-	CC
	IF MOD(Measures) *"Update evaluation related to measure"*	M	-	CC
	IF MOD(Refined by): IF ADD(Refined by): MOD(Aggregation) OR ADD(Aggregation) to the refined measure. *"The refining measure needs to be used in an aggregation "* IF DEL(Refined by): MOD(Aggregation). *"Check use of measure in aggregation"*	M	-	CC
	IF MOD(Refines): MOD(Aggregation) OR ADD(Aggregation) to the refined measure. *"For aggregation in refined measure, check use of available measures"*	M	-	CC
Measure ADD	MOD(Name) *"Provide measurement name"*	M	Name ≠ 0	CC
	MOD(Description) *"Provide measurement description"*	M	Description ≠ 0	CC
	MOD(Type) *"Provide measurement type"*	M	Type ≠ 0	CC
	[optional] MOD(Normalization) *"If the measure is intended to be used for normalization, mark it as normalization measure."*	M	Normalization ≠ 0	CC
	IF Normalization==0, MOD(Measures) *"Define which factor is associated with this measure"*	M	(Measures ≠ 0) OR (Normalization =1)	CC
	MOD(Refines) *"If measure is part of an aggregation, specify which measure is refined"*	M	Refines ≠ 0	CC
	Decide: [ADD(Instrument)* XOR ADD(Aggregation)] *"This measure needs to be associated to an instrument or to an aggregation"* IF Decide == ADD(Instrument) IF Type == number OR percentage THEN ADD(metric-based or manual instrument) *"Provide a metric-based or manual instrument"* IF Type == points THEN ADD(manual instrument) *"Provide a manual instrument"* IF Type == findings THEN ADD(rule-based instrument) *"Provide a rule-based instrument"*	M	IF Decide == ADD(Instrument) THEN Check that an instrument is referenced by the measure IF Decide == ADD(Aggregation) THEN Check that an aggregation is referenced by the measure	CC

Measure DEL	AND was normalization measure THEN Decide: *"This measure is used for normalization, are you sure that you want to delete it?" Yes/No* IF Decide == No THEN measure is not deleted IF Decide == Yes THEN IF Evaluations using the measure == 0 complete measure deletion ELSE MOD(All Evaluations using the measure) *"Please re-work the evaluations using this measure before deleting it"*	M	No evaluation uses the measure anymore	CC
	AND was used by factor AND factor has no refinements THEN ADD(Measure) *"Provide a measure for the factor"*	M	Check that the factor has a reference to a measure	CC
	AND was used by factor AND factor has refinements THEN MOD(Evaluations that were using the measure) *"Check use of deleted measure in evaluation"*	M	No evaluation uses the measure anymore	CC
Factor MOD	IF Purpose==Evaluation IF MOD(name): For all refined/impacted factors: check evaluations for correct use of changed factor. *"Check correct use of modified factor in evaluation"*		No evaluation on refined or impacted factors uses the old name of the factor being modified.	CC
	IF Purpose==Evaluation IF MOD(maxPoints): Check consistency of rule in evaluation *"Check rule consistency in evaluation / regenerate QIESL"*	M	-	CC
	IF Purpose==Evaluation IF MOD(refinedBy): If a sub factor is deleted, check consistency of rule in evaluation. *"Check rule consistency in evaluation / regenerate QIESL"*	M	Deleted sub factor is not used in the factor's evaluation.	CC
	IF Purpose==Evaluation IF MOD(refinedBy): If a sub factor is added, check consistency of rule in evaluation. *"Check rule consistency in evaluation / regenerate QIESL"*	M	All sub factors are used in the factor's evaluation.	CC
	IF Purpose==Evaluation IF MOD(influencedBy): If an influencing factor is deleted, check consistency of rule in evaluation. *"Check rule consistency in evaluation / regenerate QIESL"*	M	Deleted influencing factor is not used in the factor's evaluation.	CC
	IF Purpose==Evaluation IF MOD(influencedBy): If an influencing factor is added, check consistency of rule in evaluation. *"Check rule consistency in evaluation / regenerate QIESL"*	M	All influencing factors are used in the factor's evaluation.	CC
	IF MOD(measured by): update rule in evaluation. *"Check rule consistency in evaluation / regenerate QIESL"*	M	-	CC
	IF MOD(evaluated by): define rule in evaluation. *"Define / generate new QIESL"*	M	-	CC
Factor ADD	MOD(name) *"Provide factor name"*	M	Name ≠ 0	CC
	MOD(description) *"Provide factor description"*	M	Description ≠ 0	CC
	MOD(maxPoints) *"Provide maxPoints"*	M	MaxPoints ≠ 0	CC
	MOD(characterizes) *"Define which entity is characterized by this factor"*	M	Characterizes ≠ 0	CC
	IF Purpose==Evaluation AND IF ≠Product Factor MOD(measuredBy) OR MOD(refinedBy) OR MOD(influencedBy) *"Associate the factor with a measure or refine the factor or define an impact on this factor"*	M	Measured by ≠ 0 OR Refined by ≠ 0 OR Influenced by ≠ 0	CC
	IF Purpose==Evaluation AND IF Product Factor MOD(measuredBy) OR MOD(refinedBy) *"Associate the factor with a measure or refine the factor"*	M	Measured by ≠ 0 OR Refined by ≠ 0	CC
	IF Purpose==Evaluation MOD(evaluatedBy) *"Provide an evaluation for the factor"*	M	Evaluated by ≠ 0	CC
	[optional] IF Purpose==Evaluation AND IF Product Factor MOD(influences) *"(optional) Define which factors are influenced the new one"*	M	Influences ≠ 0	CC
Factor DEL	IF Product Factor 1) DEL(influences) 2) Check consistency of rule in evaluations of influenced factors. *"Check rule consistency in evaluation / regenerate QIESL in influenced factors"*	1) A 2) M	-	CC
	IF characterized entity is a leaf in its hierarchy AND its sub entities and it are not used by other factors: DEL(entity)	A	-	CC
	IF measures used are no longer used by other factors or aggregations: DEL(measure)	A	-	CC
	DEL(evaluation)	A	-	CC
	DEL(subordinated factors)	A	-	CC

Impact MOD	MOD(justification) IF justification == null: "Provide a justification"	M	Justification ≠ 0	U
	MOD(effect) (from positive to negative or vice versa) *"Check rule consistency in evaluation of target factor / regenerate QIESL"*	M	-	U
	MOD(source) OR MOD(target) A) IF source."type" ≠ "product factor" AND target."type" ≠ "viewpoint factor" *"Impacts can only be used from product factors into viewpoint factors."* *"Check rule consistency in evaluation of target factor / regenerate QIESL"*	M	-	U
Impact ADD	*"Provide a justification"* *"Provide a target"* *"Provide an effect"*	M	Justification ≠ 0 Target ≠ 0 Effect ≠ 0	U
Impact DEL	*"Check rule consistency in evaluation of target factor / regenerate QIESL"*	M	-	U
Evaluation MOD	MOD(specification) IF "test" ≠ "all ok": *"Check rule consistency in evaluation / regenerate QIESL"*	M	-	U
Evaluation ADD	*"Provide a name"* *"Provide a description"* *"Provide a specification"*	M	Name ≠ 0 Description ≠ 0 Specification ≠ 0	U
	Verify that refinements / impacts have also an evaluation.	A	Refinements and impacts have an evaluation.	CC
	If a factor has more than one evaluation and the model is intended to be operational, the user needs to choose which one to use (if not resolved during pre-tailoring). *"Choose an evaluation for factor to be used"*	M	Factor.number of evaluations = 1	CC
Evaluation DEL	IF Purpose == Evaluation AND Factor.number of evaluations == 0: *"Define an evaluation for the factor"*	M	Factor.number of evaluations ≠ 0	CC
Entity MOD	MOD(name) THEN (factors characterizing it change their name also) *"Check rule consistency in evaluations using factors characterizing the entity / regenerate QIESL"*	M	-	CC
Entity ADD	*"Provide a name"* *"Provide a description"* *"Add factor(s) characterizing entity"*	M	Characterized by ≠ 0	CC
Entity DEL	For all associated factors: DEL(factor) For all subordinated entities: DEL(entity)	A	-	CC

Appendix B Practitioners' Pocket Guide

What's Quality Model Adaptation About?

Software quality models are used for describing and assessing the quality of software products. They can be created from scratch, which very often requires intensive expert effort. Alternatively, quality models specifying a prescriptive set of quality characteristics or metrics can be reused. Unfortunately, those models are very often focused on a very specific context or are too general, making it difficult to reuse them.

Quamoco models offer the possibility of adapting quality models according to your specific purposes.

Why not create a huge all-purpose model?

Using a "monster" quality model influences the assessment results. You won't always be able to collect many (probably a lot) of the measures and consequently you will need to somehow handle that missing information. In fact, specially adapted quality models will make it easier to compare software products and will help you make decisions regarding those products.

This document guides you through the steps of quality model adaptation such as identifying an adequate model to be customized and identifying necessary changes (what, when and how is to be modified).

Furthermore, tailoring[7] can occur at different levels of abstraction, which also should be considered, in order to allow customizing a model from different perspectives.

Scoping Quality Model Adaptation

Software quality models may exist and be applied at different levels:

Public level: The models at this level are universally available, they may be intended for general use (e.g., Quamoco base model), or for some specific domain (e.g., embedded systems). Using and tailoring these models could be useful for showing adherence to some standard.

[7] We use the terms adaptation, tailoring, and customization interchangeably.

Organization level: At this level, quality models focus on satisfying the interests of an organization such as a whole organization, a business unit, or a project portfolio. The quality models here focus on issues that are organization wide relevant. They are more specific than quality models at public level. They are common models intended to be tailored for particular projects. At the organization level, public models are refined for a particular organization.

Project level: At the project level, quality models are put into operation; they are applied to describe and assess quality. At this level, quality model adaptation should be limited to minor adjustments driven by project-specific quality requirements, without drastically changing the structure of the organization's quality model. This helps to preserve the conformance of quality evaluations across software products created at the project level. At the project level, organization-wide models are further refined for a particular project.

Decide the scope of the quality model you will obtain using adaptation: is it for your organization or for a specific project?

The reuse of a quality model for adaptation is in essence more efficient than creating a new model from scratch for each project. We recommend tailoring models stepwise: for organizations and for projects.

Adapting a Quality Model

Identifying a Reference Quality Model

Where do I start?

The first thing you need to do is to define the goal of the quality model which will result from the adaptation. Think: which is the intention of the model I need?

In order to define the goal, you need first to know the organization/project quality needs and context information. That is, it is necessary to identify the circumstances under which the quality model will be used.

You need to think about:

1. ***What are the elements in the software, for which quality is to be defined, measured or assessed?*** For example documentation, source code, requirements, design, build process, test suite, components, statements, classes, memory, sub-routines, macros, functions, procedures, etc. This information is the object of in your goal.

2. ***For which purpose do I need the quality model?*** The Quamoco metamodel considers two different purposes: specification and evaluation of quality. Specification means that quality is described, but

not quantified. For the purpose of evaluation, quality is quantified, measured, and compared to defined assessment criteria to check the fulfillment of those criteria.

3. ***From which perspective is quality described or evaluated?*** Do we have specific requirements from the management? Which agreements do we have with the customer? Are there established practices on the organization that we want to consider with the quality model?

4. ***Which properties of the software product do we want to cover with this model?*** Quality can focus on general attributes, such as reliability, usability or maintainability, but also lower-level attributes or specific aspects can be considered, such as globalization, learnability, or training.

5. ***Which is the context of the software products to be judged by the model?*** The context may include many different things. Are there things which are mandatory within the organization? Which domain should be covered by the quality model? E.g., railway, medical devices, embedded systems, information systems, etc. Which methodologies, practices, or technologies should be supported? E.g., component-based software development, agile development, open source software, custom development, C++, Java, automatic measurement tools, etc.

All your answers to these questions are now part of your goal, which will help focusing on the key elements of the adapted quality model.

Don't forget to document the goal of your adaptation!

To get the best of your work, we recommend documenting the adaptation goal in a structured way by listing the five categories, or goal parameters you have thought about using the GQM goal template[8]: (1) *Object*, (2) *Purpose*, (3) *Viewpoint*, (4) *Quality Focus* and (5) *Context*.

The documented goal can be used later, for example, if the model is inspected, to corroborate that it actually serves for the stated goal.

How do I find the right model for adaptation?

Now you can search a model and adapt it to the needs of your project or organization. We call this model, the model on which for the adaptation is based,

[8] Basili, V.; Weiss, D. (1984): A methodology for collecting valid software engineering data. In: IEEE Transactions on Software Engineering, vol. 10(3), pp. 728-738.

a reference model. Finding the right reference model consists in finding the model whose attributes best fit your defined goal.

If no model exactly fits the goal, then you need to use it partially and search for models satisfying the most relevant parameters in your goal.

Sorting Out Irrelevant Information

Once you chose a reference model the actual tailoring can start. The adaptation typically starts by discarding elements that are not needed in the model. Only quality model components in the reference model that are relevant for your new model are taken over. In this way, unnecessary quality model components are eliminated. The parts selected to stay in the model are the foundation for further adaptation.

Fine Tuning

After sorting out irrelevant information, the model you obtain may not be consistent or operational anymore. For this reason, the removal of model components triggers further adaptation tasks. These adaptation tasks help you to bring the model back into a consistent, operational state.

Adaptation tasks can be, for example:

Add an *Evaluation* to a *Factor*. This is based on the rule:

If a *Factor* influences another *Factor*(s), it must have an *Evaluation*.

Choose an *Evaluation* for a *Factor*. This is based on the rule:

If a *Factor* has many *Evaluations*, the user must tell the model which is the "active" one, the *Evaluation* which will be used for assessment.

Many adaptation tasks can be automated. Other tasks will require your interaction as they are based on your decisions.

Accomplishing all adaptation tasks will lead you to obtain a correct model customized to your needs. You need to stepwise delete, add or modify elements in the model until no more adaptation tasks are requested. The extent to which these operations are used depends on the appropriateness of the reference model. At this point, you have successfully adapted the quality model to satisfy the goal you defined.

Reusing parts of other models

You can add new elements to the model, or, you search them in other models. That is, you can take individual elements from other models and reuse in your new model.

What should be documented?

Relevant decisions: In parallel to quality model adaptation, you need to document relevant decisions, especially those changes that exclude elements that were marked as mandatory in the organization's model.

Goal of adapted quality model: The goal is a compact manageable description of your quality model. If during tailoring the appropriateness of the model with respect to the goal is put into question, the need to complete the model, so that the goal can be achieved, must be documented, as well as the fact that the model is not complete. This may happens if something cannot be modeled before studying or finding out more specific information.

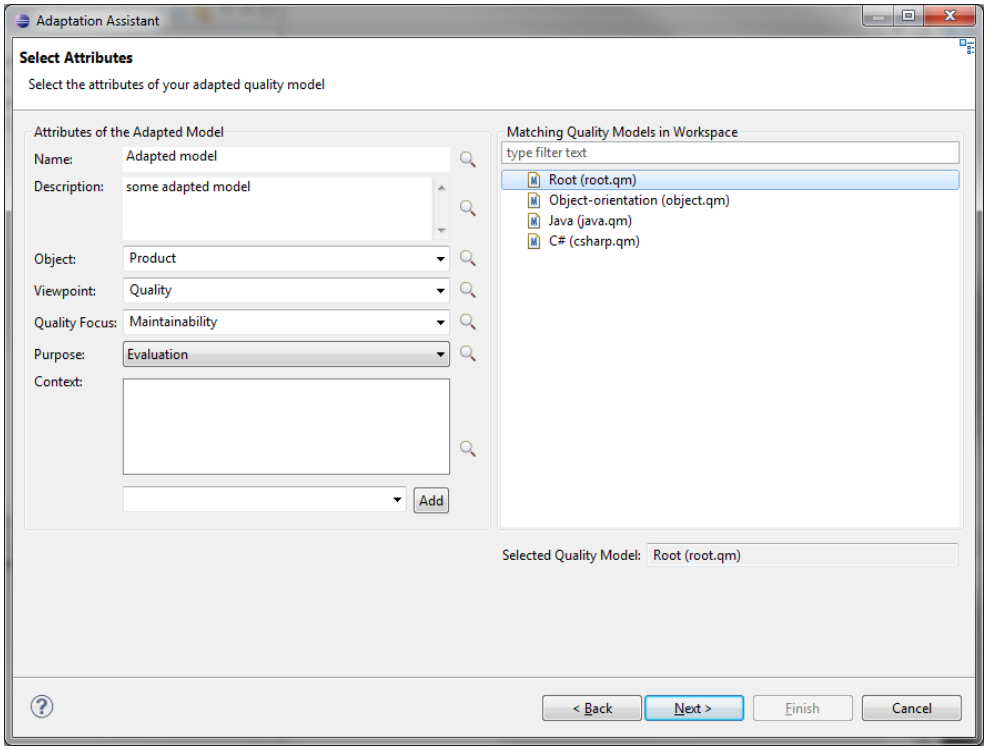
Deviations from reference model: Operations on elements which were mandatory for the organization and are no longer being considered in the sub-organization or project and the reasons of non-inclusion. Eventually, an agreement should be signed that those changes are approved. The QAM responsible for the organizational model can take these changes and their justifications as a source for changes on the organization quality model.

Testing your Adapted Quality Model

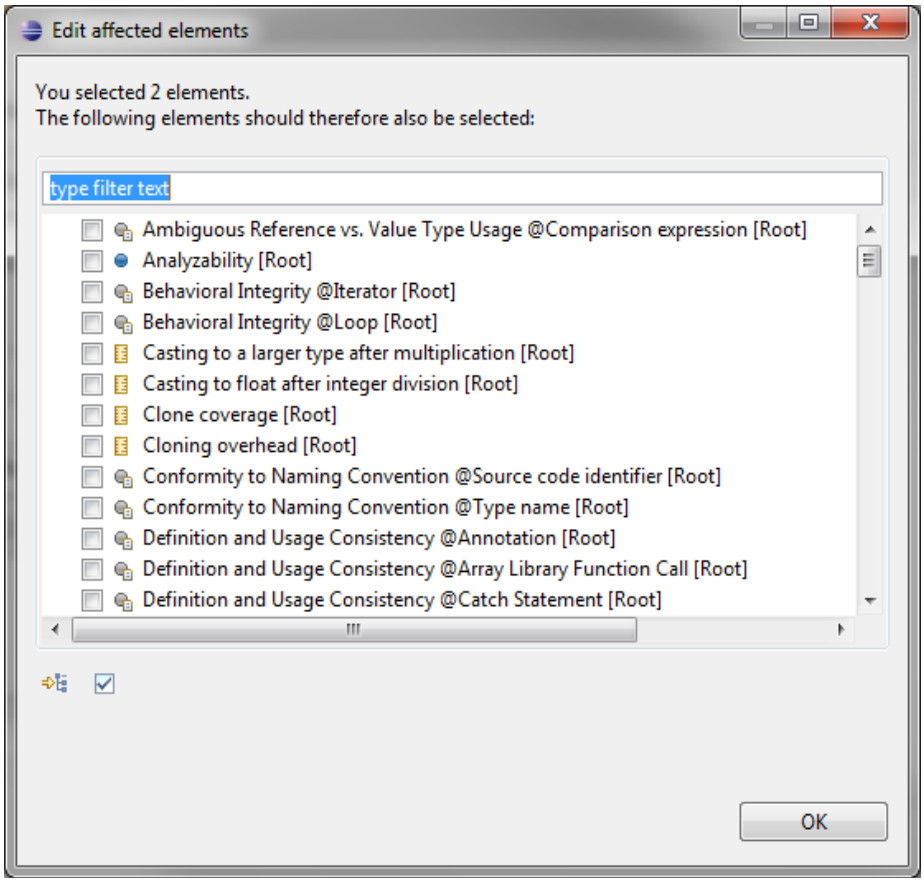
In this step, you need to test your adapted quality model at a small scale, i.e., you can conduct pilots to validate the adapted quality model. This will lead you to a final acceptance or rejection of the model. Based on the results, you can perform further adjustments.

Appendix C Tool Guide – Adaptation

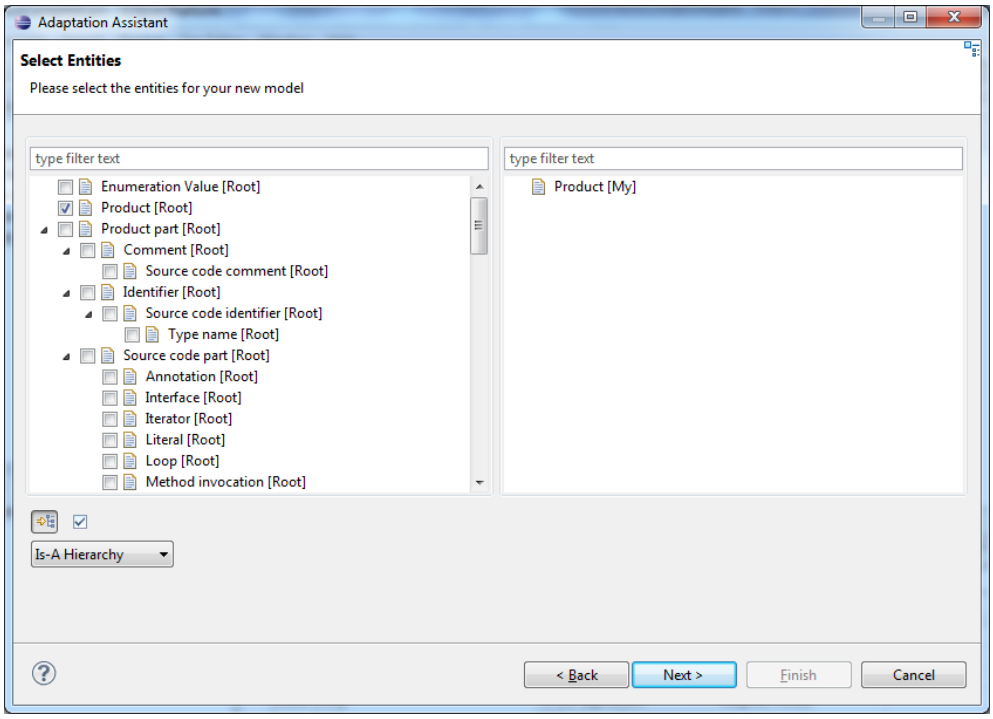
Using the Adaptation Wizard



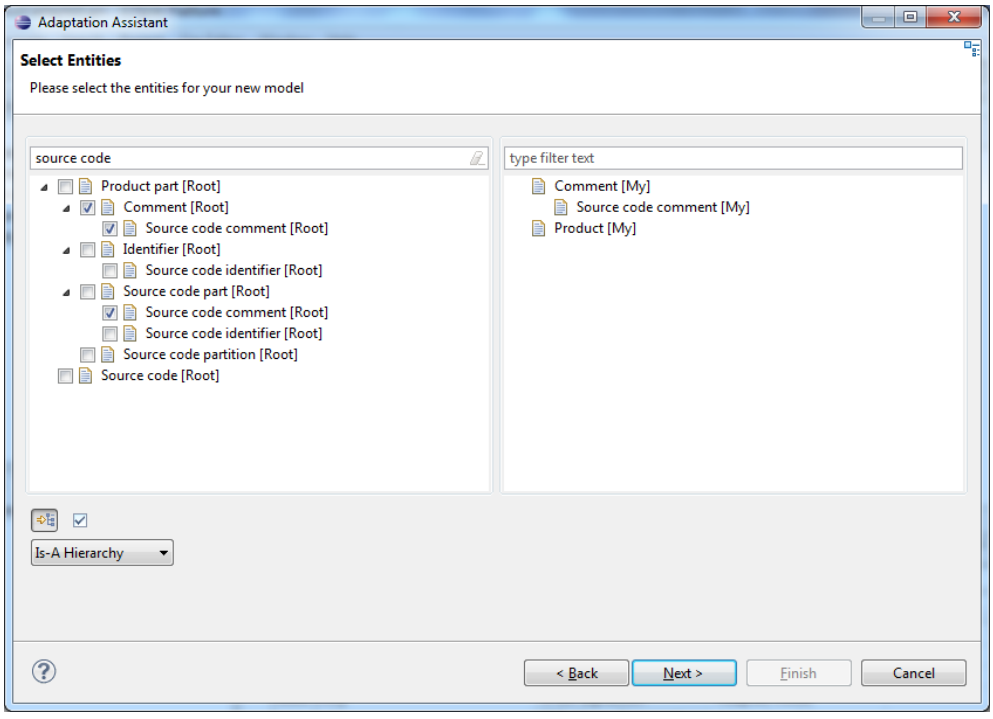
- Right-click on the model you want to adapt, and choose „Adapted Quality Model"(merge the reference models to a single model first, not done here if you wonder about the screenshots).
- Click next and choose the file name of the new model.
- On the next page choose a name, a description, an object, a viewpoint and a quality focus.
- Furthermore, select a purpose (this influences which pages of the wizard are shown) and a context. You can also filter by all these criteria by press-ing the magnifying glass on the right to search for a matching model in the work-space.
- Click next, the dialog that will open presents the transitive reflective closure around all defined pre-selections (see next section). In the bottom left corner there are two buttons, one for displaying a hierarchy (all that are defined in the adaptation model) can be used here, and another one for selecting all. By default, all elements are selected, click **OK** to continue.



- The next page is a typical „Selection page“
  - On the left page the reference model is presented, on the right the resulting adapted model.
  - Above the tree viewers there is a filter box so search for entries
    - In the left bottom corner there are two buttons and a checkbox
    - Triggers the hierarchy, the dropdown below lets you choose a hierarchy (if there is more than one defined)
    - Selects all elements that are currently visible in the tree (if a filter text is applied the select all only applies for the visible ones)



The next figure shows the filtering. It should be mentioned that elements can be shown twice in a hierarchy (when it's not a strict hierarchy).



That's basically it. **Click through the rest** of the wizard pages and select the elements that you want in your adapted model and **click finish**.

- Finally, the wizard creates the adapted model and opens the quality model editor with it.

The Adaptation History View

- The history view allows you to track all actions performed for the adaptation model. It also lists the source where the actions took place (adaptation wizard or editor).
- You can enter justifications for the performed actions in the justification column
- Click in the upper right corner of the view to open the menu which allows you to delete the complete history

The Adaptation Task View

- The adaptation task view shows you all open tasks for the currently opened quality model as defined in the adaptation model
- The button in the upper right corner lets you filter the completed (or ignored) tasks

- The view is linked to the editor, so every task you double click on will select the corresponding element in the editor. If you select an element in the editor, the corresponding tasks in the view will be selected
- Right-click on tasks, you can:
    - Mark them as completed (only for manual tasks, not for those requiring to set an attribute or reference)
    - Choose to ignore them
    - Delete them (note if tasks for missing attributes / references will reoccur when the quality model is loaded again → better choose to ignore them)
    - Select all (visible – you can filter in the top box) tasks
    - Delete completed tasks from the model

4. The Adaptation Model

- The adaptation model describes the behavior of the wizard and the adaptation tasks that are created.
- Choose New → Other → Adaptation Model and make sure "Use default adaptation model as draft" is checked, this will copy the default model for you so you don't have to start from scratch.
- In the following all elements will be explained

Adaptation Model The root element of the model.	
Name	The name of the adaptation model (as will show in the selection dialog if there are multiple models available)
Description	Describes the model
Override Default Adaptation Model	If set to true, this model will be used as default if it is the only one in the workspace – the built-in default model will not be used. If set to false, the user is asked which one to use (e.g. when the wizard starts)
Quality Model Version	This is used to check if the adaptation model corresponds to the current quality model version. If this attribute does not match to the currently plugged-in QM version, a warning will occur when the model is loaded
Possible Sub-elements: *Wizard (1)*	

Wizard The root element describing the wizard.	
Show Final Comparison Page	This feature is currently out of use, it was supposed to trigger a comparison page at the end of the page where the old and the adapted model could be compared (using the standard Eclipse mechanism)
Show Preview Buttons	This feature is currently out of use, the individual wizard pages had preview buttons that would open a comparison dialog for the currently edited elements
Possible Sub-elements: *Adaptation Element (n), Purpose (n), Attributes Page (1)*	

Adaptation Element An adaptation element is a type (like Entity) which may be modified during wizard execution. If "show on purposes" is set, the wizard will show a selection page for this element.	
Title	The title of the wizard page for this element

Description	The description for the wizard page of this element
Reference Name	The reference of the quality model which holds the elements that shall be specified here (e.g. entities)
Show Hierarchy By Default	Whether to toggle the first "Element Hierarchy" for the wizard page or not (if false the view will be flat, if true the first hierarchy will be used to show a tree)
Show On Purposes	Defines on which purposes a selection page will be shown for this element (the user chooses a purpose in the beginning of the wizard)
Possible Sub-elements: *Unselected Element Actions Remove Element Action (n), Pre-selection Descriptor (n), Element Hierarchy (n), Feature Required Action* – ignore the rest!	

Unselected Element Actions Remove Element Action The default action that must be there for every element that has a selection dialog. The element is deleted if the element is unselected in the wizard.	
Name	A name for this action, will be shown in the history view
Description	Describes the action (actually pretty useless)
Possible Sub-elements: -	

Pre-selection Descriptor If the owning adaptation element is selected a pre-selection descriptor describes which elements shall also be selected.
Possible Sub-elements: *Feature Descriptor (1)*

Feature Descriptor Describes a feature which is defined in the QM meta-model.	
Reference Type	Incoming or outgoing reference; if it is an outgoing reference the owning adaptation element has a reference to the other element type; if incoming the other element type has a reference to the owning adaptation element
Other Element Type Name	The type of the other end of the reference

Feature Name	The name of the feature
Possible Sub-elements: -	

Element Hierarchy Defines a hierarchy that can be offered to the user to show a tree instead of a flat view.	
Label	Will be shown to the user to identify the hierarchy
Feature Type	Parent feature or children feature; parent feature means that the children have a reference to their parents; children feature means that the parent has a reference to their children.
Reference Name	The name of the reference as defined in the meta-model
Possible Sub-elements: -	

Feature Required Action This action is performed at the end of the wizard or during adaptation in the editor. During adaptation in the editor, only actions with "delete if not set" = false will be considered – since no elements shall be deleted during editing. If the feature is not set then, a to-do task will be generated with the message specified in the according attribute. If "delete if not set" is true, the wizard will at the end delete all elements whose specified features are not set (i.e. null or empty list) – in this case to-do message is obviously not necessary.	
Name	A name for this action
Description	Describes the action (actually pretty useless)
Consider Subclasses	True if subclasses of the owning adaptation element shall be considered, false if not
Delete If Not Set	Deleted the instance of the adaptation element if the feature is not set at the end of the wizard execution
To-do Message	If the element is not deleted, an adaptation task will be generated with this to-do message
Possible Sub-elements: *Feature Descriptor (1)*	

Purpose Defines a purpose for the execution of the adaptation wizard. Depending on the purpose, selection pages for adaptation elements will be shown or not, as specified.	
Name	The name as the purpose, is shown to the user
Description	Describes the purpose
Possible Sub-elements: *Remove elements of type*	

Remove elements of type This action removes all elements in the adapted quality model of a certain kind, when the owning purpose is selected.	
Name	The action name
Description	Describes the action (actually pretty useless)
Type Class Name	The class name specifying the type that shall be removed
Possible Sub-elements: -	

Attributes Page Sets some values for the "Select Attributes" right in the beginning of the adaptation wizard.	
Title	The title of the page, will be shown to the user
Description	The description of the page, will be shown to the user
Object Hierarchies	Defines which hierarchy shall be used for filling the Object dropdown and how deep. Level = 1 means that only the top-level elements in the selected hierarchy will be used to populate the dropdown
Object Hierarchy Level	
Quality Focus Hierarchies	As above, same for quality focus
Quality Focus Hierarchy Level	
Viewpoint Hierarchies	Defines which hierarchy shall be used for the Viewpoint dropdown, there will always be top-level elements
Viewpoint Exceptions	Defines which factors will be included in the pre-selection anyway
Possible Sub-elements: -	

Document Information

Title:	Selection Criteria and Process for Quality Model Tailoring
Date:	June 2012
Report:	IESE-034.12/E
Status:	Final
Distribution:	Public Unlimited