Fraunhofer Institut
Experimentelles
Software Engineering

A Framework for Product Line Quality Model Development

The PuLSE-Eco Meta Quality Model

Authors:
Klaus Schmid

In part supported by the
ESAPS project
Eureka Σ! 2023 Programme,
ITEA project 99005

IESE-Report No. 047.00/E
Release 1.0
June 21, 2001

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 competetive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach
Sauerwiesen 6
D-67661 Kaiserslautern

Table of Content

1 Introduction

This report provides the background material for developing product line quality models. While the approach aims at the development of quality models to support the PuLSE-Eco V2.0 scoping approach[1], the resulting models are not restricted to this context, but support the development of arbitrary, general purpose product line quality models.

This report does not provide any quality models, but provides a common framework for the development and documentation of these models. This framework describes the underlying structure of the specialized models, as well as a basic approach for developing them. This helps to ensure that the various quality models will adher to the same general structure.

The resulting quality models are described in companion reports (effort modelling [2], reliability modelling [3]). The motivation behind this guidebook is to provide guidance to anybody attempting to develop additional models for not yet covered quality aspects.

As the quality models that are built using this approach are still generic models, the resulting models still need to be adapted to the project-specific context, turning them into customized models. This gives the three level hierarchy depicted in Figure 1.

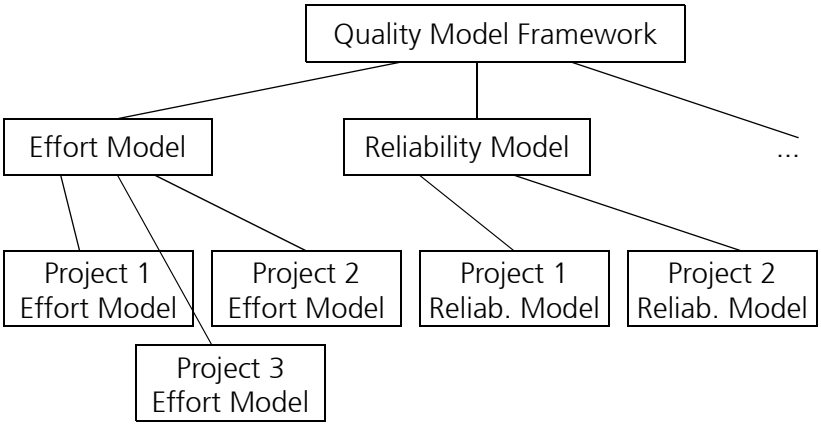

Figure 1: Hierarchy of quality models used in PuLSE-Eco

The quality model framework depicted in Figure 1 is described in this report.

1 PuLSE™ is a registered trademark of Fraunhofer IESE.

The framework is supposed to be instantiated for quite a large number of product line quality models over time. An analysis of existing standards on quality factors (cf. [5]) and of product line literature lead to the following list of qualities as potential candidates:

- Process Qualities:
  - Effort and derived qualities as TTM, pure maintenance effort, etc.
- Product Qualities:
  - Reliability (only impact on)
    - Maturity
    - Fault Tolerance
    - Compliance
  - Usability (only impact on)
    - Understandability
  - Efficiency (only impact on)
    - Time behaviour
    - Resource utilisation
  - Maintainability (only impact on)
    - Analysability
    - Changeability
    - Stability
    - Testability
  - Portability (only impact on)
    - Adaptability

The breakdown given above is based on [5] and restricted to those factors where an impact on quality from product line can be reasonably expected. All of the above-mentioned qualities are generally accepted to be impacted by product line development, thus in order to derive appropriate models for them adapted quality guides would be needed. However, we expect only to develop models for those that have a major impact on the business decision of product line development.

The remainder of this report is structured as follows: in Section 2 we discuss the general structural aspects of product line quality models, i.e., how is such a model structured, what is the structure of product line development, as far as it has an impact on the quality model, etc. In Section 10, we discuss basic inputs and approaches to the development of the generic quality models. The subsequent section then discusses how to describe the generic models for the specific qualities and how to instantiate them for specific projects. Finally, Section 6 discusses how the quality models can be used for answering quality related questions about the product line.

2 Structure of Product Line Quality Models

In this section, we concentrate on the basic structure of quality models for product lines, as oppossed to the development and usage of these models.

We discussed the relationship between the meta-model, the generic quality models, and the instantiated models for specific projects already in the previous section (cf. Figure 1). Here, we will particularly focus on the structure of the generic models. We will describe the various parts and discuss what is product line specific.

2.1 Product Line Quality Models as Extensions to Conventional Quality Models

There are three basic situations that needs to be distinguished in case one asks for a product line specific quality model:

1 There already exists a model for estimating the respective quality in the organization

2 There already exists profile data for the quality, but not an estimation model

3 There exists neither a model nor profile data

The last situation actually goes beyond the realms of what we are discussing here in the context of PL quality models. In this case, existing knowledge on developing estimation models for one at a time system development should be applied in order to develop basic models that can be extended towards product line models.[1] Thus, we will restrict ourselves in this report to the question of how to extend existing quality models towards product line quality models.

In order to address situation 1 and 2 in a homogenous fashion, we use a basic model structure, which is compositional. This approach is shown in Figure 2. As is illustrated there, the base information is taken from a standard quality model and then the deviation according to product line development is added to it. One should note that there are actually different models for the different product line development situations (cf. Section 2.2). Besides the specific situation this deviation strongly depends upon the specific product line development plan that is envisioned. Thus, it is shown as an additional input. Further, other influ-

1 While we do not focus on the specific development of base models for the various qualitites, some product line quality models also include default base models for the different models.

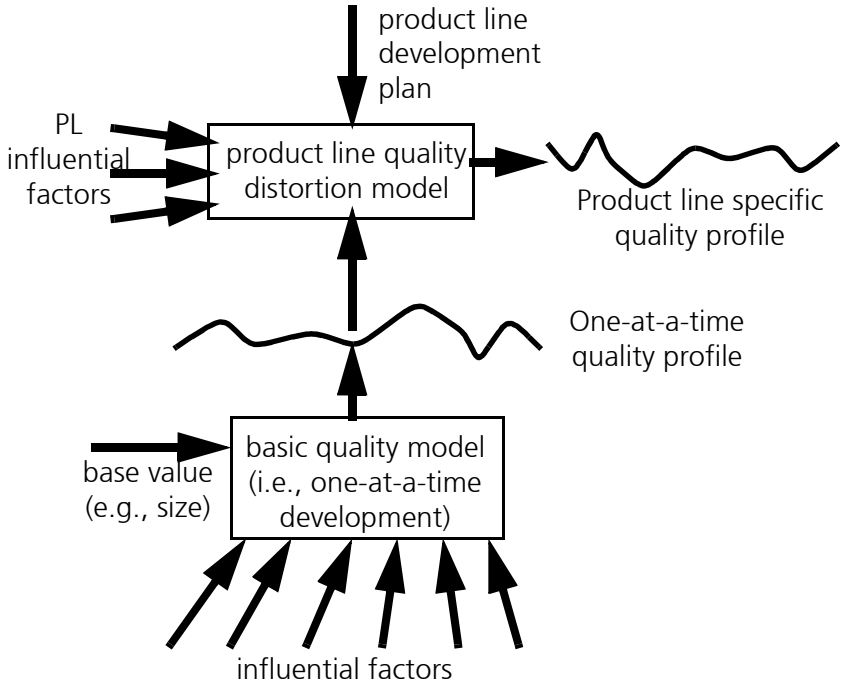

Figure 2: Overall model structure for product line quality models

ential factors (e.g., project-, personnel-, or product line specific aspects) determine the specific form of the deviation. These influential factors may overlap with the influential factors that are needed for the basic quality model, but usually a major part of the factors come in in addition to the base factors. If such a base model is not available, but a profile (i.e., a description of the quality distribution) is available, then one can start with the one-at-a-time quality profile directly. We use the following notation to describe the quality modeling shown in Figure 2:

Given a quality Q (e.g., eff: effort or rel: reliability) and a feature f for which the to-be-expected quality has to be determined, $Q_{base}(sit(f, P))$ denotes the value of the quality attribute for this feature according to the basic quality model, where $sit(f, P)$ denotes the characterization of the feature and its overall development situation in terms of influential factors. In particular, P denotes the product (P) context. In cases, where functionality is retrieved from a different product P′ this is denoted by mentioning the P′-context in addition.

On the other hand, in the case of product line development different product line development situations (i.e., modes) have to be distinguished (cf. Section 2.2). Consequently, we denote the corresponding quality attribute value for the product line situation as $Q_{PL}(mode, sit(f, P, P'))$ and denote the product line distortion itself as: $\partial Q_{PL}(mode, sit(f, P, P'))$. Using this notation we can describe Q_{PL} as:

$$Q_{PL}(mode, sit(f, P, P')) = \partial Q_{PL}(mode, sit(f, P, P')) \cdot Q_{base}(sit(f, P, P'))$$

	reusable asset Reuse	development from scratch New	implemented in different product Adapt	implemented in legacy product Reeng
system-specific development (**spec**)	Q_{PL}(spec,reuse,sit(f,P)	Q_{PL}(spec,new,sit(f,P))	Q_{PL}(spec,adapt,sit(f,P,P'))	Q_{PL}(spec,reeng,sit(f,P,P'))
development with reuse (**with**)	Q_{PL}(with,reuse,sit(f,P))	Q_{PL}(with,new,sit(f,P))	Q_{PL}(with,adapt,sit(f,P,P'))	Q_{PL}(with,reeng,sit(f,P,P'))
development for reuse (**for**)	Q_{PL}(for,reuse,sit(f,P))	Q_{PL}(for,new,sit(f,P))	Q_{PL}(for,adapt,sit(f,P,P'))	Q_{PL}(for,reeng,sit(f,P,P'))

Table 1: Product Line Development Situations

So far, we did only deal with quality attributes on a per feature basis. However, there are some general (infrastructure) activities, which will usually be difficult to relate to a single feature (cf. Section 5).

Depending on the specific quality analysed, it may be very hard to attribute the quality contribution of this activity to a certain feature. For example, gathering the high-level requirements will be very hard to relate to single features from an effort point-of-view. However, from a defect point-of-view the requirements will show up as defects in specific software features. We will discuss these infrastructure activities further in Section 2.2.

In general the quality impact of these activities can only be analyzed on a per-project level. Consequently, they depend on project characteristics. We will denote this by: $Q_{PL}(mode, sit(proj))$. Where $sit(proj)$ gives a project-classification in terms of situational factors.

In order to understand the role of the product line development plan, one needs to understand various different software development modes that are relevant to product line development. This will be discussed in Section 2.2. In the subsequent sections we will then discuss various issues related to the notion of product line development situations. Finally, in Section 5 we will wrap up this discussion and show how the product line development plan uses the situation-specific models for deriving product line quality models.

2.2 Product Line Development Situations

We can distinguish different situations in which product line development happens depending on whether we are doing development for reuse or with reuse. In addition, we can differentiate different starting situations, depending on whether some software already exists and if so, in what state it is. If we take the development of software for a specific system additionally into account, then we can classify the product line development situations along the following two dimensions:

- *development approach*
  - development for specific system (one-at-a-time development; single systems are developed)

  - development with reuse[1] (using an existing reuse base a new system is developed – this is also called *application engineering*)
  - development for reuse (reusable assets are developed – this is also called *domain engineering*)

- *starting situation*
  - nothing exists (development from scratch)
  - reusable asset exists
  - implemented in a different product (needs adaptation)
  - implemented in legacy system (needs reengineering)

These two dimensions define the matrix given in Table 1[2].Using this notation we can write:

$$Q_{base}(sit(f, P, P')) = Q_{PL}(spec, new, sit(f, P, P'))$$

The core reason for distinguishing between the different situations using such a matrix is that the actual processes enacted will vary depending on the situation and thus also the influence of the quality will change accordingly. As the specific activities will actually vary from company to company, we discuss here prototypical, high-level categories of activities.

As we discussed in Section 2.1, we focus on the quality impact on individual features. However, there are some process steps, which have an impact on the quality aspect, which cannot be easily attributed to a single feature. A prototypical example for this is the development of a reference architecture and the relationship of this activity to effort.

In the case of single system development (**spec**), we can identify the following main activities, if we assume that implementation happens via the successive integration of features:

1 Note, that in the development with reuse specific system parts may also be developed as system specifics, i.e., they are developed from scratch, but they are developed in the context of developing a system mostly from reusable pieces.

2 Note, that a mode is actually described using two parameters: $mode = (x, y)$. As a notational simplification we write $Q_{PL}(x, y, sit(f))$ instead of $Q_{PL}((x, y), sit(f))$.

- General requirements gathering
- Detailed requirements gathering *(can be attributed to individual features)*
- System architecture development
- Detailed design *(can be attributed to individual features)*
- Implementation *(can be attributed to individual features)*
- Unit test *(can be attributed to individual features)*
- Integration (test) *(can be attributed to individual features)*

Besides the basic development phases, we also want to take into account the maintenance phase. Here, we focus on pure problem correction:

- Problem correction *(can be attributed to individual features)*

The activities given above apply for the starting situation *new*. They have to be adapted for the other starting situations accordingly. However, the identified infrastructure activities stay the same in this case. When looking at the categorizations as "related to individual features" vs "general activities" given above, one should keep in mind that depending on the specific quality under study the categorization may look somewhat different.

Similarly to the categorization given above, if we look at the product-line based development of software, we can in a matching form identify general activities and activities that can be attributed to individual features.

If we look at the standard product line process given in Figure 3, we can distinguish the following activities for the two life-cycles (here, we count scoping as part of the domain engineering life-cycle.)

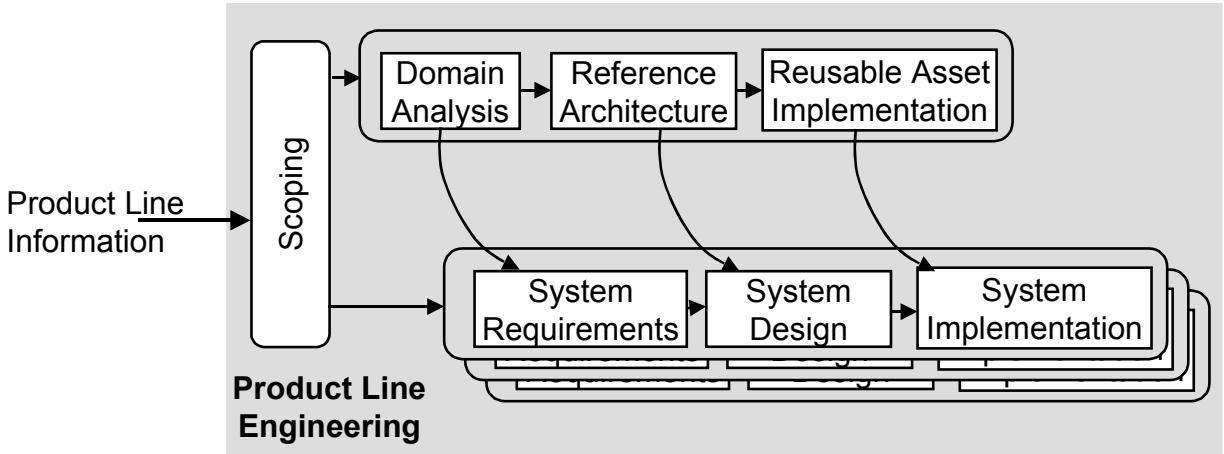

Figure 3: Product Line Process Illustration

As two different main life cycles exist (domain engineering and application engineering), we have to distinguish the identified main activities accordingly:

Domain Engineering:

- Product line mapping
- Scoping
- Detailed Domain Analysis (can be attributed to individual features)
- Development of reference architecture
- Detailed design (can be attributed to individual features)
- Implement feature reusable (can be attributed to individual features)
- Unit test (can be attributed to individual features)
- Problem correction (can be attributed to individual features)
- Product rebuilding[1] (can be attributed to individual features)

The product rebuilding step is specific to a product line development approach, as in this case if a generic asset is updated, we need to integrate the new instance of the component into all systems, so that the problem correction is integrated everywhere.

Similarly, there is the application engineering life-cycle. Here, we have again two main process variants, depending on whether a reusable asset for this feature already exists or not. In the case that no reusable asset for the system part exists, the process is rather similar to the process for single system development.[2]

	For parts that are reuseable	**For parts that are not yet reuseable**
		General requirements gathering
(per feature)	Details for system-specific adaptation	Detailed requirements gathering
	Derive product instance architecture	
(per feature)	Retrieve feature implementation	
(per feature)	Instantiate feature implementation	Detailed design
(per feature)		Component implementation
(per feature)		Unit test
(per feature)	Integration (test)	
(per feature)	*problem correction done in domain engineering*	problem correction

1 Is performed only as a consequence of problem correction.

2 Note, that in the case that non-reusable sub-systems become as small as individual features, we have a direct mapping to the quality impact also for "generic" activities.

Again, we have specific variants of these processses for the situations where already products exist which can be adapted or reengineered.

If we look at the steps that have now been identified as general infrastructure steps we find that these are the high level analysis (requirements, scoping) and development (architecture) steps. In addition, general activities like configuration management may play a role, but this is outside the scope of this study. We will defer the discussion of these general steps till Section 5 and will for now concentrate on the activities that can be attributed to individual features. Based on the discussion above, we can derive the list of activities given in Table 2:

	reusable asset Reuse	development from scratch New	implemented in different product Adapt	implemented in legacy product Reeng
system-specific development (**spec**)	*not applicable*	a) Detailed requirements b) Detailed design c) Implement F d) Unit test e) Integrate F in P	a) Detailed requirements† b) Identify F in P' c) Recover code from P' d) Adapt Code for P e) Unit† test d) Integrate† F in P	a) Detailed requirements†† b) Identify† F in P' c) Recover† code from P' d) Adapt† Code for P e) Unit†† test f) Integrate† F in P
development with reuse (**with**)	a) Detailed requir. b) Instantiate feature b) Retrieve F b) Integrate F in P††† *	a) Detailed requirements* b) Detailed design* c) Implement* F d) Unit* test e) Integrate* F in P	a) Detailed requirements b) Identify* F in P' c) Recover* code from P' d) Adapt* Code for P e) Unit†* test f) Integrate†* F in P	a) Detailed requriements b) Identify†* F in P' c) Recover†* code from P' d) Adapt†* Code for P e) Unit††* test f) Integrate††* F in P
development for reuse (**for**)	not applicable	a) Detailed domain analysis b) Detailed design** c) Implement** F d) Unit test**	a) Detailed domain analysis† b) Identify** F in P' c) Recover** code from P' d) Make code generically reuseable e) Unit test†**	a) Detailed domain analysis†† b) Identify†** F in P' c) Recover†** code from P' d) Make code generically reuseable† e) Unit test††**

Table 2: High-level Development Activities Based on Product Line Development Situations

If we want to look at the total quality for a specific feature (or product) then we will usually have to take contributions from different modes into account (e.g., feature is developed for reuse, later it is reused in different products). We will discuss this composition of quality values in more detail in Section 5.

In the above table many activities can happen in different modes leading to variations of how the activities will be performed and this may consequently have different impacts on those qualities. In order to distinguish between the same activity in different modes, we mark multiple occurences with † and *. We use † to differentiate among instances of the same activity in different starting situations and * to differentiate among instances of the same activitiy in different development approaches (e.g., integrate).

Now, we will turn to a different concern: in principle we need for each of the fields in the table (and actually for each of the high-level activities) a different quality estimation model (at least a different quality distortion model). This

would in many situations be prohibitively costly from an effort point-of-view.[1] Consequently, we need to study possibilities to reduce the number of needed sub-models.

2.3 Reduction of Product Line Situations

In the preceeding section we mapped out the space of product line development situations and discussed how hogh-level activities vary throughout these situations. However, in real world applications the amount of data that would be necessary to construct individual models for each of these situations would usually be prohibitively large. Consequently, we need to look into possibilities for reducing the overall number of required (sub-)models.

In this section, we will only discuss general reduction possibilities. More specific possibilities may turn up for specific qualities or within specific projects. These will be discussed in the respective reports. Obviously, any sort of model simplification leads to additional constraints on model-building and may negatively impact the precision of the models. Consequently, the simplifications described here should be re-evaluated in the specific project contexts.

From Table 2 we can easily infer that a large number of activities are rather similar in the various modes. While they are never identical, they are similar enough to warrant some simplifications. In this section, we will discuss two major simplifications that directly lead to mode reductions and will point out an analogy which can also be used for reducing the amount of model-building.

The first simplification, we want to point out relates to the distinction between system-specific development (*spec*) and development for reuse (*for*). In the modes, where we are doing (basically) development with reuse, but do not have a reusable asset, we are actually developing the software for a specific system, despite the fact that we are doing this in the context of product line engineering.

While the product line development environment will usually have some impact (e.g., reference architecture) we can assume that this will usually be rather small. In particular, as we do here explicitly exlude the general activities like reference architecture development, we can make the following three simplifications:

- $Q_{PL}(\text{with}, \text{new}, \text{sit}(f, P)) = Q_{PL}(\text{spec}, \text{new}, \text{sit}(f, P))$
- $Q_{PL}(\text{with}, \text{adapt}, \text{sit}(f, P)) = Q_{PL}(\text{spec}, \text{adapt}, \text{sit}(f, P))$

1 The precise number of (sub-)models (e.g., whether a per-activity decomposition is needed) obviously depends on the exact granularity on which the different models are needed.

- $Q_{PL}(\text{with, reeng, sit(f, P)}) = Q_{PL}(\text{spec, reeng, sit(f, P)})$

This reduces the overall number of sub-models we need to look at from ten to seven.

On the other hand, if we look at the reuse step (with, reuse), we see that the impact of this step on the final product quality will usually be slim, in case we truely do product line engineering. This is due to the fact, that in true product line engineering components will be reused verbatim and the integration will strictly adhere to the constraints provided by the reference architecture. Thus, if the reference architecture has been appropriately developed, so that components can be integrated in a straightforward manner into the architecture, then the integration will only require a neglectable effort, will not contribute to the defects, etc. Thus we will usually assume that:

- $Q_{PL}(\text{with, reuse, sit(f, P)}) = 0$

One should keep in mind that this approximation strongly relies on the existence of a well-defined reference architecture for the product line and the presence of verbatim reuse. If these preconditions are not met, this simplification cannot be made.

If we now look at the remaining six modes and compare them with each other, we can notice that there is an analogy between the "with-" and the "for-" category. The common difference is that in the "for-"category development is done with the goal of developing generically reusable assets based on a domain model, while on the other hand in the "with-"category we are aiming at developing assets for specific system requirements. Thus, we save the additional development activities for making the software more generic, but have to invest additionally in integrating the assets with the overall product. However, as these differences only depend on the development approach , but not on the starting situation, we can assume (as an approximation) that the impact on the quality-attribute only depends on the difference with respect to the starting situation. Actually, in some development processes the transition between the two rows may actually be explicitly performed as an individual activity "make more general", i.e., the feature implementation may be first aimed for a specific system, but within the bounds of the reference architecture and is later on turned into a generic implementation. Using $Q_{PL}(\text{gen, sit(f, P)})$ as notation for the quality impact of this activity, we can make the following assumptions:

- $Q_{PL}(\text{for, new, sit(f, P)}) = Q_{PL}(\text{spec, new, sit(f, P)}) + Q_{PL}(\text{gen, sit(f, P)})$
- $Q_{PL}(\text{for, adapt, sit(f, P)}) = Q_{PL}(\text{spec, adapt, sit(f, P)}) + Q_{PL}(\text{gen, sit(f, P)})$
- $Q_{PL}(\text{for, reeng, sit(f, P)}) = Q_{PL}(\text{spec, reeng, sit(f, P)}) + Q_{PL}(\text{gen, sit(f, P)})$

Using this simplification we can again reduce the number of quality sub-models from six to four. Note, that in order to make use of this simplification it is suffi-

cient to have either $Q_{PL}(gen, sit(f, P))$ or both $Q(for, x, sit(f, P[P']))$ and $Q(with, x, sit(f, P[P']))$ for at least one x.

Using the above simplifications we were able to reduce the required number of quality sub-models from ten to four. Additional reductions may be possible for specific qualities.

3 An Approach for Developing Product Line Distortion Models

In the preceeding sections we discussed at quite some length the various modes for which we will need (sub-)models and the relationships among them. In this section we will discuss in more depth, a so-to-speak default approach for building these models. This approach will hardly be the most optimal approach for all kinds of product line aspects. However, we think it allows a reasonable fall-back strategy in cases where no other good approach is known. In particular, the issue of product-line specific models has not yet achieved a lot of attention in literature. Consequently, no other default approaches exist.

In cases, where the existing approach for quality estimation approaches for the base quality can be extended towards product line engineering, this should be done. In the other cases (i.e., no such extension seems reasonable), modeling the base quality using an existing approach and modeling the product line distortion using the approach described here, should be regarded as a standard approach.

The approach described here is actually derived from the Cobra-method for cost- and risk-modelling [3]. Consequently, integration with the approach described here for developing the product line distortion model is rather straightforward.

The basic approach for developing product line distortion models as it is described here is based on the Corba [3] model-building approach.This approach integrates both qualitative (expert knowledge) and quantitative data. This reduces the overall requirements of this approach for data, which in turn is a major requirement for a model-building approach which is suppossed to be used in conjunction with scoping, where usually only rather little data is available.

The model-building phase of the Cobra-method consists of the following main steps:

1 Identifying the quality drivers

2 (opt.) Develop qualitative causal model

3 Develop Data Questionnaire

4 Quantify Relationship

5 Operationalize Model

Below we will now discuss each of these steps in turn with a particular emphasis on how the steps differ from the corresponding steps in the original method and

what is particular about their application in the context of product line distortion models.

3.1 Identification of Quality Drivers

There are actually two levels of quality driver identification. First of all the various quality drivers potentially relevant to a certain quality should be identified. These should be described in the corresponding quality report (cf. Section 1). As a basis for this both literature and experts can serve. If we use this approach to develop base models for a quality, we can freely use literature as a basis for this. However, specifcally for quality distortion models there is hardly any applicable literature available. In this case we have to rely even more on expert knowledge.

Given such a description of potential quality drivers, those quality drivers particularly relevant to the specific project need to be selected. This should be usually be performed by the project experts. These are on one hand the experts of the organization coming from the environment, and on the other hand experts on the product line approach that is introduced in the organization. The latter group of experts may also be part of the organization or may come from the consulting organization, if there is not yet a sufficient understanding in the organization of the new approach.

When selecting the experts for performing the rating one should keep in mind that we are here addressing the product line distortion model, but product line development may still be a rather new idea to them at this point, thus the group responsible for the introduction of the new approach needs to be appropriately included. This is a particular important issue if the model building for the base quality and for the product line adaption happens simultaneously.

A typical approach is to express the importance of the various quality drivers using importance scales[1]. As these scales are constructed in such a way as to provide numerical equivalences, we can than compute an average importance values for the different influential quality drivers. Note, that this ranking approach is substantially different from the approach used in [3].

This initial rating can be performed upfront by the experts via questionnaires. After these ratings have been gathered, agreeement of the experts needs to be analyzed. Typical agreement measures are the standard deviation among ratings or Kendall's coefficient of concordance. When disagreements have been found they should be discussed with the experts and resolved.

From the final list the most important factors should be selected as a basis for the modeling effort.

1 Importance scales (both in german and in english) were developed and are described in Appendix A.1.

3.2 Develop Qualitative Causal Model

This step aims at identifying interactions among the individual quality drivers. The reason for this is that interactions among the different quality drivers may change the impact for the quality drivers on the final result and thus on the overall precision of the final estimates.

While capturing the interrelations among quality drivers will improve the overall precision of the final models, it also has the disadvantage that it puts a higher workload on the experts, thus we regard this step as optional. The decision has to be made as a trade-off between the increase in precision that can be expected and the additional load on experts.

Like the first step, this step can happen on two levels: during development of the general quality model and during development of the project-specific quality model. In the first case only qualitative causal relations that hold generally (i.e., for every kind of project) among the different quality drivers are captured. During development of the project-specific model this model will be restricted to the specific quality drivers relevant in the specific project and will be augmented with additional causal relations that hold only for the specific project.

The standard way for developing a causal model is to perform expert interviews where the experts are asked in what way the various quality drivers are expected to impact the quality and whether are other cost drivers that have to be considered simultaneously in order to determine the impact of the cost driver.

Usually, different experts will identify different relationships. Consequently, the various proposal need to be gathered, integrated, and the final model needs to be validated with the experts.

This basic approach is identical on both levels, the general quality model level and the project-specific quality model level.

3.3 Develop Project Data Questionnaires

For each of the quality drivers identified above appropriate questionnaires need to be developed, which basically provide a characterization of the drivers in terms of independent variables. For the rest of the process it is helpful to use equally spaced scales as values for the variables.

Typically used scales are frequency, evaluation or agreement

A standard variable decomposition should be provided along with the list of quality drivers in the general quality guide.

This basically reduces the development of project specific questionnaires to the selection of the relevant subset of variable definitions. However, if additional qualitative causal relationships are identified for the project they need to be

taken into account for the questionnaires. Further, the final questionnaires should be validated with the people who will have to fill in the questionnaires, finally, whether they are useable from their point of view.

Experience shows that depending on the people involved scales sometimes will need to be adapted appropriately [3].

3.4 Quantifying Relationships

The quantitative relationships are highly project- and situation-dependent. Consequently, we will not be able to provide general quantitative information on the quality model level. Therefore, the quantification has to happen completely on the project level. An approach for performing such a quantification relying on triangular distributions has been described in [3]. The approach described here is strongly based on this approach.

First, we differentiate between the nominal case and an extreme quality-overhead[1] situation. The goal is to determine how high the quality-overhead will be if the extreme situation happens (e.g., if there is no domain knowledge available, then quality will rise by 30 per cent). This information is gathered from the experts, where they are asked for minimal, maximum, and most-likely values for the quality. We call these points $\lambda_{q,\,min}$, $\lambda_{q,\,max}$, $\lambda_{q,\,ml}$ respectively, where q represents the quality in question. From these three points a triangular distribution is constructed which is used as a basis for deriving the final estimates.

Obviously, the situation is much more complex if interactions among relationships is taken into account. Then there will be different distributions depending on the exact value of the interacting factor. Briand et al. [3] provide a description of this approach, which we will not repeat here.

3.5 Operationalizing the Model

In the previous steps we determined the individual relationships the model consists of. In this section, we will briefly discuss how these relationships are used to compose a final model.

This final model will then be used to derive the specific values for individual assets that shall be scoped for product line development. We use the notation $v_\lambda(a)$ to denote the value derived in characterizing the asset a with respect to quality driver λ. Using the quantities derived in Section 3.4 above, we can

1 We use here the concept of overhead in general. In cases, where there is a quality for which more is better (e.g., reliability) it would be actually more precise to use the term *quality reduction*.

derive final values for the various quality overhead values specific for the current situation (i.e., the specific assets that shall be scoped at a certain point in time).

The basic approach is to characterize the assets in terms of the quality drivers that are seen to be relevant. The specific values that are given are transformed into numbers from 0..n, where 0 corresponds to the nominal case and n to the most extreme. This value is then used to scale the distribution derived in Section 3.4. This distribution can then be employed in different ways, e.g., using Monte-Carlo-Simulation (cf. [3]) or using the most likely value of the distribution as a predictor. No matter what way the quality overhead estimate is derived, we denote it as $o(a) = o(v(a))$ where v(a) stands for the values characterizing the asset a.

As we assume the individual quality overhead factors are orthogonal, we can arrive at the final quality overhead by adding up the individual quality overhead numbers.

The operationalization of the model as discussed above only relates to the quality overhead part. In addition we need a base estimate to derive the final estimate. If we have both a base estimate and measurement data, we can in turn use this data to improve the relationships we found.

4 Structure of Situation-specific Models

In Section 3 we discussed in general, how quantitative models for product line development can be constructed. However, as we discussed in Section 2.2 there are many different modes of software development in the case of product line development, each of which basically needs a product line distortion model. In Section 2.3 we discussed ways of reducing the overall number of product line distortion models we finally need to develop using the approach described in the preceeding sub-section.

Besides the various distortion models it will usually be relevant to develop a product line base model $Q(spec, new, sit(f, P))$. (Obviously, if such values are already given (e.g., through a different kind of model, or for another reason, then only distortion models are needed).

Using the notation presented in Section 3.5 we can describe the distortion purely as quality overhead times a basic constant α_{mode} typical of the mode:

$$\delta Q(mode, sit(f, P)) = \alpha_{mode} \cdot \sum o_{mode}(sit(f, P))$$

There are two aspects particularly noteworthy about this description:

1 We replaced a (for asset) by $sit(f, P)$, i.e., a feature characterization relative to a project context, as this is what we mean by an asset in the context of asset scoping.

2 We anotated the overhead notation 'o' with the mode, as different characteristics λ may be relevant in different modes and if the same characteristics are relevant still different values may apply in the various situations.

Using this notation we can give the following formulation for mode-specific quality estimates: (using $base=spec,new$)

$$Q(mode, sit(f, P)) = \delta Q(mode, sit(f, P)) \cdot Q(base, sit(f, P))$$

That is we describe a product line mode basically as a distortion of the base mode.

5 Product Line Quality Models

In the preceding section we discussed how product line models for the individual product line development modes may be developed. However, the ultimate goal of the quality models we are looking at in this report is to analyze the potential for product line development certain assets exhibit. In order to do so, it is insufficient to merely analyze a certain development step for a single product, but we have to look at all the different development modes an asset goes through in product line development in order to determine the impact of the different development traces.

We call these possible development traces *scenarios*. The typical scenario for product line development would be to develop an asset in domain engineering for reuse and then to reuse it in all products. On the other hand, the typical scenario for stovepipe-development would be to develop an asset of the respective functionality individually for all the different systems.

The mode-specific models we discussed in Section 2 are the basic building blocks for determining the quality impact of the different development scenarios. In order to describe how these different (sub-)models interact in order to derive a quality model for the whole product line, we will now introduce a shorthand notation for scenarios. As a basis we use the mode notation introduced in Table 1 and augment it with product specifications.

As we simply want to specify development actions for individual features, we keep this feature implicit and only give the mode the feature development goes through and the product it is developed for. We write the development approach as name of the situation, indexed by the starting mode, and write the product the development relates to in brackets. Some examples of this notation are given in the following table:

Development situation	Shorthand
System-specific development, develop artifact from scratch for product P	$spec_{new}(P)$
System-specific development, adapt artifact from different product P'	$spec_{adapt}(P, P')$
System-specific development, use asset reengineered from product P as basis	$spec_{reeng}(P)$
Develop system with reuse, based on reusable asset, for product P	$with_{reuse}(P)$
Develop system with reuse, but develop feature from scratch for product P	$with_{new}(P)$
Develop asset for reuse from scratch	$for_{new}()$
Develop for reuse, based on asset developed for specific product P (Generalize asset)	$for_{adapt}(P)$

Table 3: Scenario Situation Notation *(Examples)*

Using this notation we can now formally describe complete development scenarios. For example, the standard development scenario for product line development would be described as (in a situation with four products):

$$S_1 = for_{new}(), with_{reuse}(P_1), with_{reuse}(P_2), with_{reuse}(P_3), with_{reuse}(P_4) \quad (1)$$

In comparison a development scenario where the functionality would first be developed for a single system, then extended for generic reuse, and then reused over several products would be written as:

$$S_2 = spec_{new}(P_1), for_{adapt}(P_1), with_{reuse}(P_2), with_{reuse}(P_3), with_{reuse}(P_4) \quad (2)$$

Further, the standard stove-pipe approach would then be written as:

$$S_3 = spec_{new}(P_1), spec_{new}(P_2), spec_{new}(P_3), spec_{new}(P_4) \quad (3)$$

Using this notation, we can easily describe various development scenarios. One should also note that these scenarios can also be interpreted as operators. Thus, each of these operators provides the value of the respective quality that results (i.e., is estimated) if the operator is applied to the feature (i.e., the feature is developed in the way described by the development scenario). In cases, where we use scenarios in such a way we will add the specific quality as an index.

$$[S_3]_q(f) = [spec_{new}(P_1), spec_{new}(P_2), spec_{new}(P_3), spec_{new}(P_4)]_q(f) = [spec_{new}(P_1)]_q(f) \circ [spec_{new}(P_2)]_q(f) \circ [spec_{new}(P_3)]_q(f) \circ [spec_{new}(P_4)]_q(f) \quad (4)$$

In Equation 4 we use *f* to denote the feature on which the models are applied, *q* denotes the specific quality in question and *o* denotes the composition operator applying for the quality *q*.

The specific composition operator may vary based on the quality. For example, if the quality under study is *cost*, then the composition operator is + as costs are additive. Thus Equation 4 can be expressed (for *q=cost*) as:

$$[S_3]_{cost}(f) = [spec_{new}(P_1)]_{cost}(f) + [spec_{new}(P_2)]_{cost}(f) + [spec_{new}(P_3)]_{cost}(f) + [spec_{new}(P_4)]_{cost}(f) \quad (5)$$

Or simply as:

$$[S_3]_{cost}(f) = \sum_{i=1}^{4} [spec_{new}(P_i)]_{cost}(f) \quad (6)$$

Using this approach we can now simply denote the quality advantage of performing product line development over traditional development as the difference between the quality attribute value for product line development and the standard approach (stove-pipe-development) as the numeric difference between the two:

$$\Delta Q(f) = [S_3]_Q(f) - [S_1]_Q(f) \quad (7)$$

Note that the way the equation is written we define ΔQ to be positive in case there is a benefit of product line development over stove-pipe development. While Equation 7 only holds for four products the generalization is straightforward and is thus not further discussed here.

Equation 7 is actually a simplification in the sense that we do only compare stove-pipe development with product line development. This gives a relative benefit. However, in the more general case we will be actually interested in the question whether product line development is better than any other form of product development. This absolute advantage can also be defined using the notation introduced above:

$$\Theta Q(f) = \min\{[S]_Q(f) \mid \text{Scenarios } S, S \neq S_1\} - [S_1]_Q(f) \quad (8)$$

However, in all practical cases we will use Equation 7 for deriving the benefit of the product line approach.

6 Usage of Quality Models in Scoping

The quality models developed based on this guidebook are supposed to be used for supporting planning and management of product line development, in particular in the context of PuLSE-Eco. In particular, they are supposed to be the basis for deriving the asset scope, i.e., determine the assets that should be developed in a reuseable manner. Thus, the models are used within the product-map-based part for detailed scoping. At this point in the process a coarse-grained scope is already available.

In this section we will discuss some general issues of model structure, which are mostly orthogonal to the issues discussed in the preceeding sections, and are strongly related to the usage of the models. In the following subsections we will in turn discuss categorizations of quality drivers and how these categorizations relate to the product map.

6.1 Categorizations of Quality Drivers

Depending on the specific purpose of the categorization, a categorization of quality drivers can be performed along many different dimensions. Some examples of such categorizations are given in literature. In particular, for the purpose of specific (single-system) quality models different categorizations have been described. However, also in the context of product line development some categorizations have been described, in particular, by Bandinelli et al. The categorization used in [6] partitions quality aspects along the following dimensions:

- Organisation
- Personnel
- Process
- Products

While such a categorization of quality drivers is very helpful in order to organize data gathering and understanding of influential factors, a different categorization is important to assist in using the quality models for scoping. This differentiation is based on the *scope* of the quality drivers. We distinguish the following scopes:

- Product Line — the aspect or property holds for the complete product line (e.g., certain organization specific aspects fall into this category).

- Individual Product(s) — The aspect holds only for certain products, e.g., if only certain products need to have a certification.
- Individual Feature(s) — The aspect holds only for certain features, but with respect to these features for all products exhibiting the feature (e.g., if certain web-based features exist, they will usually be stronger impact by change, than other features, but this holds for all products in the same way).
- Individual Product/Feature combinations — The aspect holds only for certain Product/Feature combinations, e.g., the complexity of a certain functionality in a specific product (supposed the complexity depends on the other features interacting with this one).

This is similar to the categorization given by Bandinelli and Sagarduy , who analyzed the scope of reuse investment models in [7]. They came up with the following differentiation[1]:

- Domain Scope — the investment model analyzes the return over all products in the domain. This can be taken to correspond to our product line level.
- Project Scope — the investment model analyzes the return over a single product. While this basically corresponds to our product level, the results are fundamentally different, as here individual products are analyzed from the perspective that they simultaneously develop for and with reuse.
- Component Scope — the investment model analyzes the return for a single component. Here, a further distinction is made between producer and consumer view. This is further refined in our categorization, as we differentiate between complete feature and product/feature combinations.

The differences among the two classifications comes from the difference in their underlying goals. In particular, the classification given by Bandinelli and Sagarduy does not mention product/feature combinations, as they are only looking at the level of complete investment models. However, all their model scopes can be simulated in a straightforward manner, if we represent information using a product map based approach. Actually, Equation 7 gives the combined benefit from producer and consumer view over the whole organization.

Project and domain scope evaluation will be discussed in more detail in Section 6.3. Additional information on making investment decisions using this approach is given in the PuLSE-Eco method guide [4].

1 Note, that this is not — like ours — a categorization meant to differentiate among different types of quality drivers, but which is meant to differentiate among different kinds of investment models.

6.2 Process Aspects

When using the models developed according to this guide, usually the realm of the process under consideration will vary among applications, e.g., whether maintenance benefits should be considered or not, whether only implementation benefits should be regarded, etc.[1]

Thus, in order to tailor the models appropriately to the individual projects, the various models have to make their relationship to the process explicit. While it will usually not be possible to develop per-process-step sub-models, this enables to select those quality drivers up-front, that could be potentially relevant to the process steps under consideration. Clarity of the process scope is also relevant in order to baseline the models appropriately (how many person-months per Line of Code). We will use the task decomposition given in Section 2.2 as our reference process model.

As we discussed in Section 2.2, some features can be directly linked to individual features, while others cannot be linked to individual features, as they discuss the relationship among features like reference architecting does. However, the quality impact (e.g., effort) will still depend on the features over which it stretches. Consequently, we will simply take those activities into account by attributing a corresponding sub-part of the quality aspect value according to the proportional size of the feature (e.g., in the case of effort we will attribute x/s-parts of the effort of domain analysis to a feature, if *x* is the feature size (corrected by the quality drivers) and *s* is the overall project size.

6.3 Evaluating Assets Using the Product Map Approach

In this section, we briefly describe how the individual assets are evaluated using a *product map*. The product map is a tabular notation for capturing the individual information items relevant to evaluating the possible assets. This table gives an overview of the product line in the sense, that it shows the various to be developed products along the horizontal axis and lists the various features along the vertical. Further, it lists the quality driver (values) relevant to evaluating the various potential assets. At this point the categorization of quality drivers based on their scope as described above comes in:

- Product Line
- Individual Product(s)
- Individual Feature(s)

1 In one real application we had the situation that the organization was only interested in design, implementation and testing, but not in analysis or maintenance.

- Individual Product/Feature combinations

Depending on the specific category of the quality driver, it will show up in a different place in the product map. This is shown in Table 4.

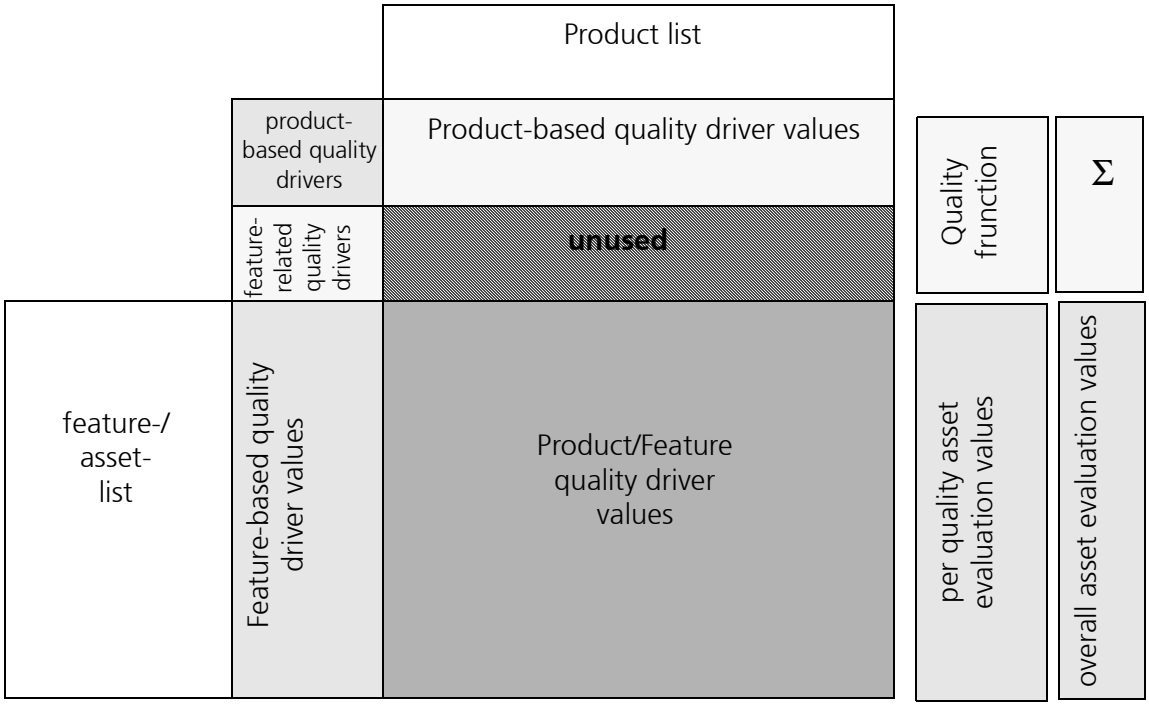

Table 4: Segments of a Product Map

Note, that the quality drivers on the product line level can be treated as constants in the model and thus do not show up explicitly in the product map. Using this approach, we can in an easy manner capture and evaluate the information relevant to the possible assets. The underlying evaluation approach corresponds to Equation 7. This evaluation is then shown on a per-quality basis to the right of the product map.

The integration of the per-quality evaluations is then based on a weighting approach. The details of this are beyond the scope of this report and are discussed in [4].

The mapping of quality drivers to the product map as discussed above works in a straightforward manner in cases where the various activities can be linked to individual features (cf. Section 2.2). If process steps are taken into account that cannot be mapped in a straightforward manner onto a feature-specific level, we will distribute the quality impact in a proper manner over the individual features as discussed in Section 6.2.

As discussed above, the product map does not show the distribution over time. However, in some cases it is needed to be able to determine the distribution of the quality aspect over time (cf. investment models, [4]). This is strongly supported, if we have per-activity sub-models. If those are not available, it is useful to have at least a standard quality decomposition available. If possible such a decomposition should be given by the individual quality guides. At least expected deviations of the product line case from the single-system development should be identified.

7 Model-Guides: Instantiating the Meta-Quality Guide

The overall purpose of this guidebook is to provide a framework for the individual quality models. These quality models are in turn described via individual model guidebooks (cf. Section 1). In this section we provide a generic outline for the individual model guides in order to promote standardization and effective working with these model descriptions. This section can be regarded as a requirements document for model guidebooks.

The overall structure of the individual model guides is as follows:

1 Background and Theory

2 Adaptation of the Framework

3 Identification of Quality Drivers

4 Experiences and Issues

5 Appendices

In the rest of this chapter we will describe the specific contents of these sections in more detail, devoting a subsection to each of these chapters.

7.1 Background and Theory

In this section a general discussion of the quality under analysis should be given. This should give some background material and the quality-specific terminology. Especially, theoretical issues particular to the treatment that is given to the subject in this guide should be presented and justified (e.g., preference of a certain analysis approach over others).

In cases where a multi-staged approach is necessary, e.g., if cost is defined through effort or reliability based on defects, the relationship of the two qualities should be discussed in this section.

This section should also provide comments on the major information used as input to the model guide.

7.2 Adaptation of the Framework

The framework describes a standard approach, a top-level process for product line development, and standard situations. While we expect them to hold more or less for all individual quality guides, some deviations might be necessary for specific qualities. These should be described in this section.

In particular, the report should explicitly discuss the following issues:

- *Product Line Development Situations* — What is the relationship of the quality to the various situations, e.g., will reuse have an impact on the quality
- *Process Steps* — What is the relationship of the quality to the various situations, e.g., do we need to consider only a subset of the process steps, do we need to alter or augment the list of steps
- *Situation Reduction* — Which situation reductions apply as given in the framework report and which reductions apply in addition to these.
  The report should provide at this point an overview of the specific product line quality (distortion) models that need to be developed.

7.3 Identification of Quality Drivers

This will usually be the most comprehensive part of the guidebook. It should first consist of an overall account of how analysis of quality drivers were performed (e.g., who was interviewed on which topic, when) and then followed by a detailed description of the various factors that could be identified.

The factor list is the core part of the whole quality model documentation. In order to provide an overview it should start with an overview diagram structuring the various factors into groups and identifying any general interactions (i.e., interactions that are supposed to be independent of the specific model instantiation). Then a detailed documentation of the model should follow, structured according to the submodels (base-model, various distortion models). This detailed documentation identifies the relevant factors, and provides measurement instruments for them.

Thus the overall sub-structure of this part should be something like this:

- Description of the Analysis
- Overview of the Model
- (per sub-model one section) Description of quality drivers

7.4 Experiences and Issues

In this section general information resulting from application or relevant to the application of the model guide should be described. As it should also describe experiences and lessons learned from applying this guide, it is supposed to continuously evolve over time.

7.5 Appendices

The appendices should contain supporting material for instantiating the model application. At least the appendices should contain:

- Questionnaire(s) for identifying the major project-specific factors (these should be using the relevancy sclale given in Appendix A).

Further appendices may exist depending on the specific quality guide.

8 Model-Guides: Project Specific Instantiation

In the previous section, we discussed the general requirements for quality guides in general. In this section, we will in turn discuss how the documentation of the application of model guides in a certain project context should happen.

Usually a single project documentation will describe all qualities relevant to the project scoping in an integrated manner. The major aspect of the project documentation is that it should provide a basis to understand:

- Why certain qualities were chosen
- How was the project conducted (what was special about the application)
- What were the results (e.g., of applying the questionnaires)
- Which factors were chosen
- How were they quantified
- Which were the final models used in scoping

The following outline follows these requirements:

1 Identified business goals and derivation of the involved qualities

   Using the business goal description for PuLSE-Eco (cf. [4]), the relevant business goals are identified, weighted, and traced to the relevant quality models.

   In this context also the relationship between the abstract process given in the quality guide and the concrete process in the organization should be discussed.

2 Identification of quality models

   An overall description of the how the relevant quality drivers were identified is given (if there are substantial differences, this can happen on a per-quality basis).

3 Resulting quality models

   Per quality model it is documented, what were the questionnaire-results, which drivers were chosen as most relevant, and what were the results of the quantification of the drivers. Derived from that the specific mathematical evaluation models are given, which will be used for the final scoping.

The sections described above will usually be only part of a larger project-specific scoping report.

9 Summary

In this report, we described the basis for developing quality models that can be used by the PuLSE-Eco scoping approach to provide a quantitative basis for the scoping of the asset base.

We described the general structure of these models as well as the necessary notation and terminology to discuss the specific models and perhaps needed deviations from the standard model.

In particular, we discussed the process aspects of product line development, we discussed the general structure of product line quality models, and discussed how specific quality models can be developed.

Further, we discussed in detail the relationship between the meta-quality model and the specific quality model guides and how these can be instantiated for the specfic projects.

10 References

[1] Klaus Schmid. *Product Line Mapping Report*. IESE-Report No. 028.00/E, Fraunhofer IESE, Sauerwiesen 6, D-67661 Kaiserslautern, 2000.

[2] Jürgen Wüst. *An Effort-Model for Product Lines*. Technical Report, Fraunhofer IESE, Sauerwiesen 6, D-67661 Kaiserslautern, 2000. Forthcoming.

[3] Lionel C. Briand, Khaled El Emam, and Frank Bomarius. *COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment*. 20th International Conference on Software Engineering, 1998, Kyoto, Japan, pp. 390–399.

[4] The PuLSE-Eco Scoping Method. Technical Report, Fraunhofer IESE, Sauerwiesen 6, D-67661 Kaiserslautern, 2001. Forthcoming.

[5] Draft ISO/IEC FCD 9126-1.2

[6] Sergio Bandinelli and Goiuria Sagardui Mendieta. *Domain Potential Analysis: Getting Serious About Product-Lines.* Third International Workshop on Software Architectures for Product Families, IW-SAPF-3, Las Palmas de Gran Canaria, Spain, March 15-17, 2000.

[7] Sergio Bandinelli and Goiuria Sagarduy. A Unifying Framework for Reuse Economic Models. Technical Report, European Software Institute, ESI-1996-Reuse03, November, 2000.

A Scales

A.1 Importance Scales

For ranking the importance of the various quality drivers (influential factors on the quality) scales have been developed both in english and in german.

(Further validation of these scales might be appropriate

German:

Value	0	1	2	3	4	5	6	7	8	9
Text	*nicht wichtig*		*etwas wichtig*			*weitestgehend wichtig*		*ziemlich wichtig*		*extrem wichtig*

Table 5: German Version of Importance Scale

English:

Value	0	1	2	3	4	5	6	7	8	9
Text	*not relevant*			*somewhat relevant*			*mostly relevant*			*extremely relevant*

Table 6: English Version of Importance Scale

Document Information

Title:	A Framework for Product Line Quality Model Development The PuLSE-Eco Meta Quality Model
Date:	June 21, 2001
Report:	IESE-047.00/E
Status:	Final
Distribution:	public