Reference: Michael Kläs, Jens Heidrich, Jürgen Münch, Adam Trendowicz. CQML Scheme: A Classification Scheme for Comprehensive Quality Model Landscapes. In Proceedings of the 35th EUROMICRO Conference Software Engineering and Advanced Applications (SEAA 2009), pages 243-250, Patras, Greece, 2009. IEEE Computer Society.
DOI: 10.1109/SEAA.2009.88, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5349845

CQML Scheme: A Classification Scheme for Comprehensive Quality Model Landscapes

Michael Kläs, Jens Heidrich, Jürgen Münch, Adam Trendowicz
Fraunhofer Institute for Experimental Software Engineering
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
{michael.klaes, juergen.muench, jens.heidrich, adam.trendowicz}@iese.fraunhofer.de

Abstract—**Managing quality during the development, operation, and maintenance of software(-intensive) systems and services is a challenging task. Although many organizations need to define, control, measure, and improve various quality aspects of their development artifacts and processes, nearly no guidance is available on how to select, adapt, define, combine, use, and evolve quality models. Catalogs of models as well as selection and tailoring processes are widely missing. A first step towards better support for selecting and adapting quality models can be seen in a classification of existing quality models, especially with respect to their suitability for different purposes and contexts. This article presents so-called comprehensive quality model landscapes (CQMLs), which provide such a classification scheme and help to get an overview of existing models and their relationships. The article focuses on the description and justification of essential concepts needed to define quality models and landscapes. In addition, the article describes typical usage scenarios, illustrates the concept with examples, and sketches open questions and future work.**

Software quality management; quality assurance; quality improvement; classification sheme

I. Introduction

The multitude of software-related quality models (QMs) available and the lack of guidance for identifying, evaluating, selecting, and adapting a set of appropriate models for a specific organization or project implies (1) a need for getting a structured classification and overview of available QMs, (2) a need for linking quality aspects to higher-level goals of a project or an organization and the respective context, and (3) a need for appropriate selection and customization processes. This article focuses on the need for getting a structured classification and overview of available QMs. Currently, a variety of QMs exists, originating from the literature, company standards, official standards, or other sources (e.g., they might be implicitly defined in measurement systems, key performance indicators, or quality gates). Typically, QMs focus on product quality (e.g., ISO9126 [24]), process quality (e.g., maturity, process adherence, or performance), and resource quality (e.g., server availability or qualification). Each of these models usually supports only a limited set of application purposes (such as characterization [17], improvement [8], or prediction [31]). In many cases, it is not obvious for which usage purposes the models are suitable, in which contexts they can be applied (e.g., in which application domains), and how to customize them. In addition, it is not clear to what extent the models have already been evaluated. In case of the availability of empirical evidence, evaluation and dissemination are typically limited to a specific context and difficult to find in the literature. In consequence, quality assurance managers, quality managers, and project planners have significant problems in identifying the appropriate set of QMs that is relevant for them. Furthermore, the lack of a uniform classification of QMs aggravates communication regarding quality aspects.

The concept sketched in this article consists of a classification scheme of QMs and the use of this scheme for creating so-called comprehensive QM landscapes (CQMLs). The idea of cartographing QMs into landscapes can be compared with Enterprise IT landscapes that aim at improving communication and alignment with business goals regarding networked IT systems in a company through graphical visualizations [33].

The article is structured as follows: Section 2 defines a set of usage scenarios for CQMLs. Section 3 discusses related work in the field. Section 4 presents basic QM constructs and typical application purposes of QMs. Section 5 sketches a classification scheme for QMs and classifies existing QMs. Section 6 presents the challenges of classifying QMs and creating CQMLs. Section 7 gives an overview of comprehensive landscapes and illustrates how to construct them. Finally, Section 8 concludes with a brief summary, discusses open questions and gives directions for future work. The article extends and refines previously published work by the authors [27], especially with respect to the summary of related work, the identification of conceptual QM constructs, their relationships to application purposes, the description of classification criteria, and the characterization of example QMs.

II. Usage Scenarios for QM Landscapes

In the context of software engineering, a number of potential decision-making processes may be effectively supported by QM landscapes. In our research, we aim at the following CQMLs usage scenarios in particular:

S1 Support orientation: QM landscapes provide a comprehensive overview of QMs that are potentially relevant for decision-making processes in a particular context. A decision maker gets a quick overview of the variety of existing models.

S2 Support communication: QM landscapes facilitate the common understanding of quality between various stakeholders involved in software development (e.g., quality assurance personnel, managers, project planners, developers, and contractors). A unified terminology and classification scheme provides a basis for aligning quality objectives across different organizational levels and different parties involved.

S3 Support gap analysis: QM landscapes support the identification of gaps with respect to quality modeling in a particular context. A researcher or decision maker identifies critical dimensions with respect to QMs. Based on these dimensions, specific software development areas are identified in which QMs would be needed to support decision-making.

S4 Support evaluation and selection: QM landscapes support effective evaluation of existing QMs with respect to their contribution to organizational objectives. Driven by these objectives, a decision maker identifies relevant dimensions in the QM landscape and identifies/selects relevant QMs.

S5 Support adaptation: QM landscapes support adapting selected QMs to specific organizational goals and contexts. After selecting and evaluating a QM, a decision maker may use the dimensions of the classification scheme for adapting the QM to his/her specific needs (e.g., addressing model dimensions that do not fit his/her specific goals and context).

S6 Support comparison: QM landscapes facilitate identifying relationships between QMs. Information on commonalities and variabilities between relevant QMs supports identifying potentially conflicting and supporting quality objectives.

S7 Support integration: QM landscapes support QM integration. Based on the evaluation and comparison of QMs, a software decision maker is able to combine selected (compatible) models.

III. RELATED WORK

A variety of QMs have been proposed over the years by software professionals. This variety reflects, in principle, various perceptions of the terms “quality” and “model” in the software engineering domain. The typical perception of software quality in the software engineering domain ([30], ch. 1) focuses on customer satisfaction and is defined either as conformance to a specification (i.e., specified user expectations) or as conformance to specific needs (i.e., specified and unspecified expectations). Yet, it has been observed that there are usually different types of “customers” in the software development context. The ISO 12207 [19], for example, specifies 5 types of views and associated stakeholders involved in the software life cycle. Georgiadou [16] grouped stakeholders proposed by the ISO standard into three groups. A comparison is presented in Table 1. However, according to Garvin [15], the stakeholder perspective on quality is only one aspect. He proposes a broader view, which goes beyond conformance to specification and needs, differentiating between a transcendental view, a user view, a manufacturing view, a product view, and a value-based view.

TABLE 1. VIEWS ON QUALITY AND RELATED STAKEHOLDERS

View on Quality [19]	**Involved Stakeholders**	**Stakeholder Group [16]**
The contract view	Acquirer, Supplier	Sponsors
The management view	Manager	
Operating view	Operator, User	Users
Engineering view	Developer, Maintainer	Developers
Supporting view	Support process employer	

Similar to multiple perceptions of quality, software practitioners have a number of various perceptions of a model, and of a QM in particular. Software QMs range from formally defined hierarchical structures of quality attributes [24] through measurement systems [34], [18] to lists of key performance indicators [8] or quality gates [14]. However, in practice, it is not clear what constitutes a QM. In consequence, existing QMs create a mixture of heterogeneous approaches that focus on different objects, consider different quality characteristics, and serve different purposes. Typical software objects considered include products [24], processes [8], or resources [5]. Quality characteristics range from very specific ones, such as modifiability, to very abstract ones, such as maturity, and their interpretation is largely dependent on the object for which they are specified. Finally, common application purposes of QMs include, for example [25], quality planning, control, and improvement.

Kitchenham et al. make an attempt to define a QM by specifying its components in the form of a quality meta-model, called SQUID [26]. SQUID represents a hierarchical structure where quality characteristics are decomposed into a set of quality sub-characteristics, which can be further refined. Similar to the idea presented in the ISO 9126 [24], SQUID’s quality characteristics represent external properties of a product that are potentially influenced by internal properties. All components of a QM are measurable, i.e., corresponding metrics and values (actual and target) can be assigned. Similar, yet often not as comprehensive as SQUID, are quality meta-models such as Boehm’s Factor-Criteria-Metric [4], the IEEE standard 1061 [18], and the ISO/IEC standard 15939 [21]; in the latter model, quality is represented by so-called information products. Yet, although a number of existing QMs can be mapped to these meta-models, there are still QM components that are not specified in any meta-model. For example, the quantification of the relationship between a product’s quality characteristics and sub-characteristics or between its internal quality characteristics and quality in operation are hardly reflected in existing quality meta-models.

The abundance and heterogeneity of existing QMs and, at the same time, the lack of a clear definition of what a QM is make it difficult for software professionals to (1) select the model that best suits a particular application context and (2) identify which (if any) parts of the model need to be adapted. In other words, a systematic approach for classifying and evaluating QMs is required in practice. Yet, to the best of our knowledge, such approaches are largely missing. First attempts to define a generic scheme for evaluating QMs were made in [9], where three criteria were defined:

- A QM should support the five different perspectives of quality as defined in [15];
- A QM should be usable from the top to the bottom as defined in [18], i.e., it should allow for defining quality requirements and their further decomposition into appropriate quality characteristics, sub-characteristics, and measures;
- A QM should be usable from the bottom to the top as defined in [18], i.e., it should allow for required measurements and subsequent aggregation and evaluation of the results obtained.

Moreover, the Goal-Question-Metric paradigm (GQM) [1] provides a systematic scheme for specifying measurement goals. It considers several dimensions, such as measured *object* (e.g., a certain software product or process), *purpose* (e.g., evaluation or improvement), *quality focus* (e.g., reliability or maintainability), *viewpoint* (e.g., user or developer), and *context* (e.g., certain domain, company, or business unit). In addition, GQM defines five standard measurement purposes: characterization, understanding, evaluation, prediction, and improvement.

Finally, [10] defines a QM as "a model with the objective to describe, assess and/or predict quality" and proposes these three purposes as a simple classification for QMs.

Besides these few generic approaches, a number of studies are available where specific subclasses of QMs are classified and/or evaluated with respect to a set of purpose-specific criteria. An example of a purpose-specific classification might be schemes for classifying software cost estimation models [7], [32]. Another example are comparative evaluations of prediction models (e.g., fault-proneness [29]), where estimation accuracy is typically considered as the most important criterion for selecting or discarding a particular model. On the one hand, such evaluation and classification schemes are significantly inconsistent with respect to the model's characteristics (i.e., evaluation/classification criteria) they consider. On the other hand, they assume that a certain group of models has already been pre-selected (e.g., models considering a specific quality aspect and/or purpose).

Summarizing, despite several attempts at standardizing QMs, a clear definition of a QM and a systematic approach to classifying QMs are still missing. In consequence, software professionals use existing QMs (often arbitrarily selected) merely as a reference for creating their own context-specific models. This leads to unnecessarily large costs of modeling quality, which often does not create the expected value in return. In order to effectively support software professionals in their decision-making processes, a systematic, standardized approach for selecting and adapting QMs is required. In this paper, we take the first step towards such an approach by proposing to systematically define basic components of a QM and a scheme for classifying/selecting QMs

IV. QMs and their Constructs

In general, QMs differ in appearance. For example, when comparing CMMI [8], IFPUG Function Points [23], and reliability growth models [31], there seem to be more differences than similarities. The reason for this is that they consist of different conceptual *constructs* as a result of their specific application purposes. When looking at QMs in detail, different constructs can be identified.

- *Refinement structure:* Several QMs decompose abstract concepts into less abstract ones (e.g., the ISO9126 decomposes quality into quality characteristics such as usability, reliability, safety, etc.)
- *Aggregation method:* When a concept is decomposed, one may like to obtain an overall statement for the decomposed concept by considering the individual results for the sub-concepts (e.g., a weighted sum of sub-results). An aggregation method is only reasonable if a refinement structure exists and the aggregated results are quantified.
- *Quantification specification:* When the results for a concept (or sub-concept) are to be compared with other results or used in calculations or statistics, they have to be quantified by defining a measure (e.g., lines of code for source code size or number of defects found by users for faultiness of delivered product).
- *Evaluation criteria:* A measured concept (or sub-concept) can be extended with evaluation criteria by providing, for instance, a threshold or target value (e.g., the system's response time should be less than 2 seconds for standard queries, or productivity in a project should be greater than 4 function points per person-month).

In addition, we may distinguish between the concept/characteristic of interest (i.e., *quality focus*) and the concepts/characteristics that influence this characteristic (i.e., *variation factors*). For example, the development effort predicted for a project by an effort estimation model such as COCOMO is part of the quality focus of the model and the domain or developers' experiences is a variation factor that influences this concept of interest. In order to identify and describe such interactions, an additional construct is required.

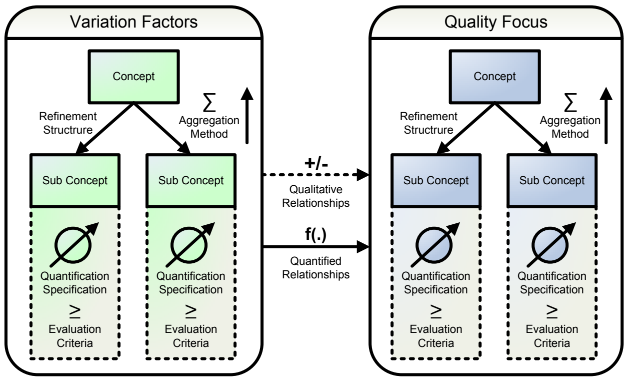

Figure 1. Overview of the conceptual constructs of QMs

- *Qualitative relationships* simply explain whether a variation factor influences the concept in focus positively or negatively (e.g., the number of inspectors increases the overall number of defects found).
- *Quantified relationships* allow quantitative statements about the impact of a variation factor on the concept in focus. For instance, in company X, inspectors with more than 5 years of domain experience (variation factor) find twice as many defects (quality focus) on average than inspectors with less than half a year of domain experience. Therefore, quantified relationships can be considered as an extension of qualitative relationships. They require a quantified quality focus as well as quantified variation factors.

While analyzing existing QMs, we observed that the conceptual constructs of a particular model determine the model's feasible application purposes. From the practical point of view, it would thus be useful to know how specific conceptual constructs map exactly to the purposes a model can be reasonably applied for. Assuming such a mapping is available, a quality manager may, for example, identify potentially applicable models for a set of relevant purposes or identify a conceptual component missing in a particular model to be applicable for relevant purposes. In order to find such a mapping a set of standard application purposes needs to be identified first.

The GQM method defines a set of measurement purposes that form a partial order with respect to an organization's measuring capabilities: *characterize* (describing the concept of interest), *understand* (explaining dependencies between different concepts), *evaluate* (assessing the achievement of the goal with respect to the concept of interest), *predict* (determining the expected value of the concept of interest beforehand), *motivate and improve* (determining what needs to be done for improving concepts of interest quantitatively).

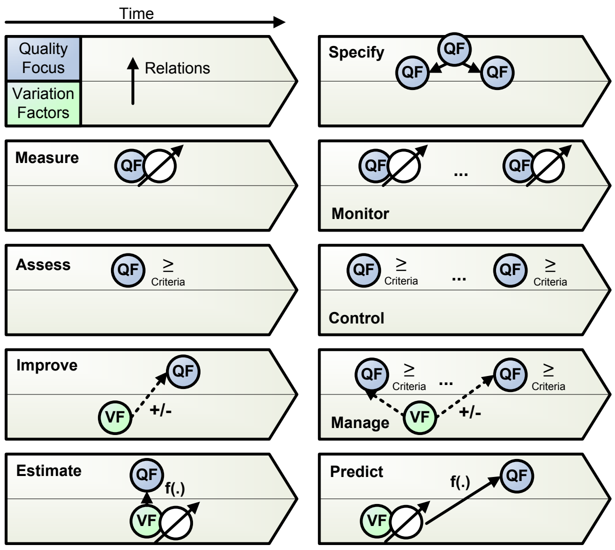

Figure 2. Typical application purposes for QMs

These purpose definitions are well suited for describing logical steps of how companies can improve in the area of quality measurement, and are mainly suited for measurement experts. However, since the QM landscapes should support practitioners in identifying relevant QMs, we provide a second set of more practitioner-oriented application purposes. The foundation of the additional set are terms typically used when describing usage scenarios of QMs, such as specifying, measuring, assessing, monitoring, evaluating, controlling, improving, managing, estimating, and predicting quality. In the following, these terms will be specified more precisely and relevant QM constructs needed for those purposes will be identified.

- *Specify* – A concept is described more precisely by refining it into sub-concepts, e.g., for supporting communication between stakeholders (e.g., process quality decomposed into its effectiveness, cost, and duration).
- *Measure* – A concept is quantified and measured to determine the actual value for the quantified concept (e.g., a product's quality measures are collected and stored in a project data repository in order to create a quantitative basis for decision making within an organization).
- *Monitor* – A concept is quantified and measured periodically or at certain points in time to identify trends (e.g., the development of the number of open issues during the testing phase).
- *Assess* – A concept is quantified, measured and compared to defined evaluation criteria to check the fulfillment of the criteria (e.g., response time of the application is less than 2 seconds for each query in a defined set of typical user queries).
- *Control* – A concept is quantified, measured, and compared periodically or at certain points against defined evaluation criteria to check the fulfillment of the criteria (e.g., controlling product complexity on a regular – weekly – basis against a defined threshold).
- *Improve* – Relevant factors (i.e., variation factors) influencing the concept of interest are known. Therefore, the concept of interest can be improved by improving its influencing factors (e.g., based on CMMI, process quality can be improved by providing traceability between products in different states of development).
- *Manage* – A concept is quantified, measured, and compared periodically or at certain points against defined evaluation criteria to check the fulfillment of the criteria. If defined criteria are missed, possible reasons and reactions can be identified due to the understanding of the variation factors influencing the measured concept (e.g., by involving more experienced developers in the project).
- *Estimate* – A concept of interest is not measured directly but estimated by measuring related concepts (e.g., capture-recapture models estimate the remaining number of defects by using the overlap of defects found by a group of inspectors).

TABLE 2. QM APPLICATION PURPOSE / CONSTRUCT RELATIONSHIP

Construct * (* X: required (x): beneficial) / Purpose	Characterize	Understand	Evaluate	Predict	Motivate and Improve	Specify	Measure / Monitor	Assess / Control	Improve	Manage	Estimate / Predict
QF - Refinement structure	(x)	(x)	(x)	(x)	(x)	X	(x)	(x)	(x)	(x)	(x)
QF - Aggregation method	(x)	(x)	(x)	(x)	(x)	(x)	(x)	(x)	(x)	(x)	(x)
QF - Quantification specification	X	X	X	X	X	(x)	X	X	(x)	X	X
QF - Evaluation criteria			X		X			X	(x)	X	
VF - Refinement structure		(x)	(x)	(x)	(x)				(x)	(x)	(x)
VF - Aggregation method		(x)	(x)	(x)	(x)				(x)	(x)	(x)
VF - Quantification specification		(x)	(x)	X	X				(x)	(x)	X
VF - Evaluation criteria					X				(x)	(x)	
VF/QF - Qualitative relationships		X	X	X	X				X	X	X
VF/QF - Quantified relationships		(x)	(x)	X	X				(x)	(x)	X

- *Predict* – The future value of a concept of interest is predicted by considering factors indicating its future value (e.g., COCOMO predicts the development effort by considering software size and a set of effort drivers).

V. CLASSIFICATION SCHEME FOR QMS

In general, categories of a classification scheme need to be (1) meaningful/minimal in the sense that they contribute to at least one usage scenario, (2) complete in the sense that all QMs can be categorized, and (3) orthogonal in the sense that the classification is as unambiguous as possible. In order to systematize and evaluate QMs, we created a classification scheme including major dimensions based on the measurement goal template provided by the well-established Goal/Question/Metric (GQM) paradigm [1]. The GQM goal template specifies five aspects that should be considered when defining goals of software measurement. We utilize this template as follows:

- *Object* specifies what is being examined by a QM. The major classes of objects are products, processes, and resources. Examples of products include whole SE products, but also parts such as the source code or specific components; examples of processes include the whole development process, its instantiations (i.e., a project), but also specific processes such as the system test process.
- *Purpose* specifies the intent/motivation of quality modeling. A detailed list was already discussed in Section 4.
- *Quality Focus* specifies the quality characteristic being modeled. Example software-related qualities are reliability of products, maturity of processes, or productivity of personnel.
- *Viewpoint (Stakeholder)* specifies the perspective from which the quality characteristic is modeled. Typically, the perspective refers to a stakeholder from whose viewpoint the quality attribute is perceived (e.g., product complexity may differ depending on whether the developers' perspective is used or the users'). We distinguish on the highest level between developer organization, customer, and user.
- *Context* specifies the environment in which the quality modeling takes place. The context characteristics should, in particular, cover aspects such as:
  - *Scope*, which specifies the comprehension of an organizational and process area covered by a QM. An example organizational scope might be the whole organization, business unit, group, or a specific project, whereas an example process scope might be development process, phase, or activity.
  - *Domain*, which specifies the domain(s) a QM covers (and is intended for). Typical software application domains include: embedded software systems, management & information systems, and web application.

Another dimension distinguishing QMs with respect to their applicability in a certain context is their level of abstraction. There are models that are not a 'model' in the narrower sense, because they define only a meta-model (e.g., defect flow models), which has the advantage of being flexible but the disadvantage of requiring explicit instantiation before application. If no model *instance* is available, an *instantiation method* can be provided to guide the model instantiation; if the model is an instance, an *adaptation method* can be useful for adapting the predefined model to the specific needs and capabilities.

Further, motivated by industrial demands, we propose considering such aspects as empirical evidence and utilization support. Since no objective measure for empirical evidence is known, we consider the model's *dissemination* and distinguish between academic models, applied models (documented in at least one case study), and models that are standards (or quasi-standards). *Utilization support* mainly refers to the amount and quality of documentation for a QM; it also includes the existence of *tools* supporting the utilization of a QM.

These characteristics serve as a basis for pre-selecting a group of QMs. As, in practice, such a group will probably contain several, largely heterogeneous, models, additional aspects need to be considered in order to select a narrower group of QMs that will fit particular demands and capabilities. It must be ensured that a model's *critical prerequisites* are fulfilled before it can be utilized. One essential aspect to consider are inputs required by a specific QM. This includes *type* of data (e.g., objective-subjective, categorical-numerical, or certain-uncertain), *amount of data*, and *quality of data* (e.g., completeness).

The schema was evaluated by classifying a given range of popular QMs. Table 3 summarizes the results for a selection of classification dimensions used and characterizes the models in terms of the conceptual constructs they provide. For providing the *main purposes*, we used the practical purpose classification and focused on the purposes mentioned by the model descriptions as being the most relevant ones (i.e., some models are at least theoretically also applicable for further purposes).

VI. CHALLENGES OF CREATING QM LANDSCAPES

A number of challenges need to be faced when creating QM landscapes. On the one hand, the classification schema must be theoretically valid and useful in practice. This includes such aspects as clear and unambiguous specification of classification criteria that ensures objective classification. On the other hand, a systematic classification process must be defined. Still, although these requirements are largely met, the objectivity of the classification results is partially threatened by the quantity and quality of information available on existing QMs. For example, the characteristics of proprietary models (e.g., company-specific models) are typically not available in the public domain. Yet, non-proprietary models are also often missing complete and unambiguous description.

In this paper, we present an example QM landscape in which several popular QMs are classified. Since the landscape considers only three dimensions and a limited number of classified QMs, it uses rather simple visualization means. However, the way of presenting potentially large and multi-dimensional QM landscapes to human decision makers is another issue that must be taken up.

VII. Comprehensive QM Landscapes

There exists a variety of different QMs for different usage scenarios. Depending on the specific scenario, different dimensions and visualization mechanisms may be important for creating a comprehensive QM landscape. There is no general landscape of QMs that fits all needs; the concrete shape of a landscape depends on the concrete usage scenario. Furthermore, it may be helpful to restrict the QMs presented in a landscape by fixing one or more dimensions. For instance, one might consider product QMs only (by restricting the "object" dimension) or one might only consider models relevant in a certain context (e.g., by restricting the scope/domain). QMs may consist of several sub-models (e.g., ISO 9126 has models for internal quality, external quality, and quality in use). Depending on the level of detail of a particular landscape, those sub-models may have to be classified separately.

The main steps for constructing a comprehensive QM landscape are as follows: (1) Characterize the context you are working in. (2) Specify the goals you want to reach by using QMs. (3) Select relevant classes of QMs based on the classification scheme and determine the dimensions that need to be visualized and the degree of detail. (4) Create the CQML based on your selection. (5) Analyze the QMs within the landscape according to your goals and their suitability for your context. (6) Package the analysis results in order to arrive at a decision regarding relevant QMs depending on your concrete usage scenario.

To give an example, let us consider usage scenario S1 of Section 1 (i.e., support orientation) for a company that wants to get a general overview of QMs for different objects, purposes, and quality foci. Figure 3 gives a preliminary landscape visualization of selected QMs according to those three dimensions. The landscape uses quite simple classes for the three dimensions:

- *Object:* A QM may consider different objects. For instance, ISO 9126 [24] makes statements about the general software product quality, CoBRA [6] addresses project costs, and reliability growth models [31] make statements about product reliability. For our landscape, we want to classify the models into three classes: product, process, and project. A QM may be assigned to more than one class or even only address parts of a class (e.g., a certain sub-process such as testing or a certain part of a product such as the design document).
- *Purpose:* A QM may support different purposes. In our landscape, we use the set of typical purposes discussed in Section 4. The ISO 9126 is mainly used to *specify* product quality and defines sample metrics for *measuring* quality.
- *Quality Focus:* Creating universal classes for the quality focus would probably be as difficult as creating a universal QM. There exist a variety of qualities and sub-qualities, which are mostly ordered and defined differently in different QMs. For our landscape, we simply want to distinguish between QMs addressing a single, specific quality focus (such as productivity as one aspect of process quality [17] or reliability as one aspect of product quality [31]) and models claiming to have a universal quality focus (usually refining qualities by sub-qualities), such as the ISO 9126 for product quality.

The set of classified QMs is not representative and the landscape does not need to contain all relevant QMs. However, from the sample landscape, one can get a quick overview of QMs and focus on specific areas for further investigation depending on specific goals and contexts.

VIII. Conclusions and Outlook

This article presented a comprehensive classification scheme for QMs based on the GQM paradigm. Selected QMs were characterized using the classification scheme, including their suitability for different purposes and contexts. Moreover, typical constructs were defined for characterizing the basic structure and features of QMs (such as aggregation or refinement capabilities). For providing an integrated overview of the variety of QMs, the concept of CQMLs was presented, which helps to get an overview of existing QMs and their relationships. As can be seen from the classification of selected QMs, there is a variety of different model types for different usage purposes.

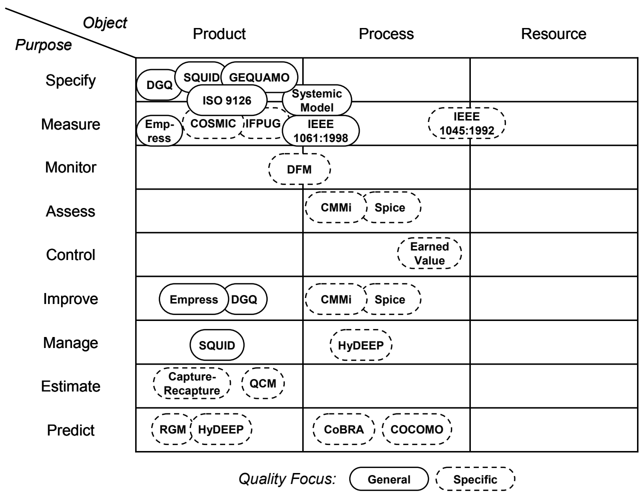

Figure 3. Sample landscape of QMs.

As there is no universal QM that fits all purposes, there is also no universal landscape of QMs. Different views on the classification scheme (making use of different dimensions of the scheme) are required depending on the purpose of a landscape. For instance, in order to get a quick overview, the three main dimensions (object, purpose, and quality focus) may be sufficient. For more thorough research in a selected area, some dimensions may need to be fixed or may need to be refined. For instance, if you want to get an overview of effort prediction models, you need to fix the quality focus on 'effort' and the purpose on 'predict' and take a closer look at other dimensions of interest (such as scope, meta-model components, or model dissemination).

Future work will be conducted in the area of refining and empirically evaluating the classification scheme for accessing and improving its applicability and usefulness for the stated scenarios. More existing QMs should be reviewed and classified based on the validated scheme to build up a database of existing QMs. Moreover, the classification scheme and the landscapes should be integrated into a comprehensive model selection and adaptation process (inspired by the comprehensive reuse process [3]) for building up an experience base of QMs. Finally, dedicated visualization means for presenting large, multidimensional landscapes should be provided. This work is planned to be conducted as part of the publicly funded BMBF project QuaMoCo.

ACKNOWLEDGMENT

This work was supported in part by the German Federal Ministry of Education and Research (QuaMoCo Project, No. 01 IS 08 023 C). We would also like to thank Sonnhild Namingha from Fraunhofer IESE for reviewing a first version of this article.

REFERENCES

[1] V. Basili, Software Modeling and Measurement: The Goal/Question/Metric paradigm, Technical Report CS-TR-2956, Department of Computer Science, University of Maryland, College Park, MD 20742, 1992.

[2] V. Basili, L. Briand, W. Melo, "A validation of object-oriented design metrics as quality indicators," IEEE Trans. Software Engineering, vol. 22, pp. 751–761, Oct. 1996

[3] V. Basili, H. Rombach, "Support for comprehensive reuse," Software Engineering Journal, 6(5):303-316, 1991.

[4] B.Boehm, J. Brown, H. Kaspar, M. Lipow, G. MacLeod, and M. Merritt, Characteristics of Software Quality. North Holland Publishing Company, 1978.

[5] B. Boehm, C. Abts, A. Brown, S. Chulani, B. Clark, E. Horowitz, R. Madachy, D. Refer, B. Steece, Software Cost Estimation with COCOMO II. Prentice Hall, 2000.

[6] L. Briand, K. El Emam, F. Bomarius, "CoBRA: A Hybrid Method for Software Cost Estimation, Benchmarking and Risk Assessment", Proc. 20th Int'l Conf. on Software Engineering, 1998, pp. 390-399.

[7] L. Briand, I. Wieczorek, "Resource Modeling in Software Engineering," in Encyclopedia of Software Engineering, 2nd Edition. Wiley, 2002.

[8] CMMI for Development, Version 1.2, CMU/SEI-2006-TR-008, Carnegie Mellon University, 2006.

[9] M. Cote, W. Suryn, E. Georgiadou, "In search for a widely applicable and accepted software quality model for software quality engineering," Software Quality Journal, 15:401-416, 2007.

[10] F. Deissenboeck, E. Juergens, K. Lochmann, S. Wagner, „Software Quality Models: Purposes, Usage Scenarios and Requirements", Int'l Workshop on Software Quality, Vancouver, Canady, 2009.

[11] Dt. Gesellschaft für Qualität, Software-Qualitätssicherung. No. DGQ-NTG-Schrift 12-51, VDE-Verlag Berlin, 1986.

[12] J. Doerr, D. Kerkow, T. Koenig, T. Olsson, T. Suzuki, "Non-functional requirements in industry - three case studies adopting the ASPIRE NFR method," IESE-Report, 025.05/E, Fraunhofer IESE, Kaiserslautern, 2005.

[13] Q. Fleming, J. Koppelmann, Earned Value Project Management 2nd ed. Project Management Institute, 2000.

[14] B. Freimut, C. Denger, M. Ketterer, "An industrial case study of implementing and validating defect classification for process improvement and quality management," 11th Int'l Symposium on Software Metrics, 2005.

[15] D. Garvin, "What does 'Product Quality' really mean?," Sloan Management Review, (1):25-43, 1984.

[16] E. Georgiadou, "GEQUAMO - A Generic, Multilayered, Customisable, Software Quality Model," Software Quality Journal, 11:13-323, 2003.

[17] IEEE 1045-1992: Software Productivity Metrics, 1992.

[18] IEEE 1061-1998: IEEE Standard for a Software Quality Metrics Methodology, 1998.

[19] ISO/IEC 12207: International Standard, Systems and software engineering - Software life cycle processes, 2008.

[20] ISO/IEC 15504-1: Information technology - Process assessment, - Concepts and vocabulary, 2004.

[21] ISO/IEC 15939: International Standard, Software engineering - Software measurement process, 2007

[22] ISO/IEC 19761: Software engineering - COSMIC-FFP - A functional size measurement method, 2003.

[23] ISO/IEC 20926:2003 Software engineering - IFPUG 4.1 Unadjusted functional size measurement method - Counting practices manual, 2003.

[24] ISO/IEC 9126-1:2001 International Standard, Software engineering – Product quality, Part 1: Quality model, 2001.

[25] J.M. Juran, Juran's Quality Control Handbook, McGraw-Hill, 1988.

[26] B. Kitchenahma, S. linkman, A. Pasquini , V. Nanni, "The SQUID Approach to Defining a Quality Model," Software Quality Journal, 6:211-233, 1997.

[27] M. Kläs, J. Heidrich, J. Münch, and A. Trendowicz, "Comprehensive Landscapes for Software-related Quality Models," 2nd Software-Qualitätsmodellierung und -bewertung. SQMB'09 - Workshop-Band, 2009, in press.

[28] M. Kläs, H. Nakao, F. Elberzhager, J. Münch, "Support Planning and Controlling of Early Quality Assurance by Combining Expert Judgment and Defect Data - A Case Study," Empirical Software Engineering Journal, 2009, doi: 10.1007/s10664-009-9112-1, in press.

[29] F. Lanubile, G. Visaggio, "Evaluating predictive quality models derived from software measures: lessons learned," Journal of Systems and Software, 3(3):225 – 234, 1997.

[30] L. Lundberg, M. Mattsson, C. Wohlin (eds.), "Software Quality Attributes and Trade-Offs." Technical Report, Blekinge Institute of Technology, Ronneby, Sweden, 2005.

[31] M. Lyu, "Software Reliability Theory," in Encyclopedia of Software Engineering, John Wiley & Sons, 2002.

[32] S. MacDonell, M. Shepperd, "Combining Techniques to Optimize Effort Predictions in Software Project Management," J. Systems and Software, 66:91-98, 2003.

[33] F. Matthes, „Softwarekartographie [Software Cartography]," in Enzyklopädie der Wirtschaftsinformatik, 1. Auflage, 2008 (in German).

[34] J. McGarry, et al., Practical Software Measurement: Objective Information for Decision Makers. Addison-Wesley Professional, 2001.

[35] M. Ortega, M. Perez, and T. Rojaz, “Construction of a Systemic Quality Model for Evaluating a Software Product,” Software Quality Journal, 11:219-242, 2003.

[36] H. Petersson, T. Thelin, P. Runeson, C. Wohlin, “Capture-recapture in software inspections after 10 years research. Theory, evaluation and application,” Journal of Systems and Software, no. 2, 2004

TABLE 3. CHARACTERIZATION OF A SELECTION OF QUALITY MODELS

		Goal Definition								*Model Elements*									
Name	*Reference*	*Object*	*Main Purpose*	*Quality Focus*	*Viewpoint*	*Instance Available*	*Instantiation /Adapt. Method*	*Dissemination (in SE)*	*Tool*	*QF Refinement Structure*	*QF Aggregation Method*	*QF Quantification Spec.*	*QF Evaluation Criteria*	*VF Refinement Structure*	*VF Aggregation Method*	*VF Quantification Spec.*	*VF Evaluation Criteria*	*Qualitative Relationships*	*Quantified Relationships*
Capture-Recapture	[36]	Pd	Estimate	Defects	D	X		A	X			X						X	X
CMMi	[8]	Pc	Assess/ Improve	Maturity	D	X		St	X	X	X	X	(x)					X	
CoBRA	[6]	Pc	Predict	Cost	D		X	A	X			X		X	X	X		X	X
COCOMO	[5]	Pc	Predict	Cost	D	X	X	A	X			X		X	X	X		X	X
COSMIC FP	[22]	Pd	Measure	Functionality	C/U	X		St	X	X	X	X							
Defect Flow Models (DFM)	[14]	Pd/ Pc	Monitor	Defects/ Effectiveness	D		X	A		(x)		X							
DGQ	[11]	Pd	Specify/ Improve	General	D	X		Sc		X		(x)						X	
Earned Value	[13]	Pc	Control	Performance	D	X		A	X			X	X						
Empress	[12]	Pd	Measure/ Improve	General	C/U	X		Sc		X		X						X	
GEQUAMO	[16]	Pd	Specify	General	D/C/U		X	A		X									
HyDEEP	[28]	Pd/ Pc	Predict/ Manage	Defects/ Effectiveness	D		X	A		X	X	X		X	X	X		X	X
IEEE 1045:1992	[17]	Pc/ R	Measure	Productivity	D	X		St				X							
IEEE 1061:1998	[18]	Pd/ Pc	Measure	General	D/C/U		X	St		X		X	X						
IFPUG FP (excl. VAF)	[23]	Pd	Measure	Functionality	C/U	X		St	X	X	X	X							
ISO 9126	[24]	Pd	Specify/ Measure	General	D/C/U	X		St		X		X		(x)		(x)		(x)	
QCM	[2]	Pd	Estimate	Defects	D		X	A	(x)			X						X	X
RGM	[31]	Pd	Predict	Reliability	C/U	X		A	X			X						X	X
Spice (ISO 15504)	[20]	Pc	Assess/ Improve	Maturity	D	X		St	X	X	X	X	(x)					X	
SQUID	[26]	Pd	Specify/ Manage	General	C/U		X	A	X	X		X	X	X		X	X		
Systemic Model	[35]	Pd/ Pc	Specify/ Measure	General	C/U	X	X	A		X									

Object: Product (Pd), Process (Pc), and Resource (R)
Viewpoint: Development Organization (D), Customer (C), and User (U)
Dissemination (in SE): Scientific (Sc), Applied (A), and Standard (St)

X: Exist
(x): Exist partially, implicitly, or on meta model level