# The Squale Model – A *Practice*-based Industrial Quality Model

Karine Mordal-Manet[1] Françoise Balmas[1] Simon Denier[2] Stéphane Ducasse[2]
Harald Wertz[1] Jannik Laval[2] Fabrice Bellingard[3]
Philippe Vaillergues[3]

[1] LIASD, University of Paris 8, France
[2] RMoD Team, INRIA, Lille, France
[3] Qualixo, Paris, France
kmordal@gmail.com
http://www.squale.org

## Abstract

*ISO 9126 promotes a three-level model of quality (factors, criteria, and metrics) which allows one to assess quality at the top level of factors and criteria. However, it is difficult to use this model as a tool to increase software quality.*

*In the Squale model, we add* practices *as an intermediate level between metrics and criteria. Practices abstract away from raw information (metrics, tool reports, audits) and provide technical guidelines to respect. Moreover, practice marks are adjusted using formulae to suit company development habits or exigences: for example bad marks are stressed to point to places which need most of the attention.*

*The Squale model has been developed and validated over the last couple of years in an industrial setting with Air France-KML and PSA Peugeot-Citroën.*

## 1 Introduction

Software quality aims at setting up standards to achieve and measuring the conformance of a project with the company requirements. Companies use software quality as a mean to assess risks in the course of software development. Such risks include for example faulty software, inertia to change, misunderstanding, which turn out in longer time to market and higher costs. It is then important to provide means to increase software quality in form of guidelines and recommendations to reduce these risks.

Software quality models often present multiple levels of quality assessment in a top-down fashion. ISO 9126 [6] describes such a quality model with the three levels of factors, criteria, and metrics. Factors ("non-technical" quality properties) are decomposed into criteria (high level technical properties), which are further decomposed in terms of metrics computed on project data. However, this model is difficult to use as a mean to increase software quality [8].

In the Squale, we add *practices* as an intermediate level between metrics and criteria. A practice abstracts from the raw nature of the extracted information and already supports quality concerns by combining and weighting different metrics. It is then possible to express that test covering complex methods is more important than simple accessors. Practices cover different aspects of a software project—including documentation, programming conventions, and test coverage—and constitute a technical guidelines to respect.

This model has been first implemented by a company, Qualixo, in a industrial setting with Air France-KLM (AF) in 2006. It has been intensively used to monitor multiple projects, for a total of seven MLOC. Since then other companies such as PSA Peugeot-Citroën (PSA) are using it. The Squale quality model is now open-source.

## 2 The Squale quality model

The Squale model is inspired by the factors-criteria-metrics model (FCM) of McCall [10]. The Squale model introduces the new level of *practices* between criteria and metrics. Practices are the key elements which bridge the gap between the low-level measures, based on metrics, rule checkers or human audits, and the top-level quality assessments—expressed through criteria and factors. Thus the Squale model is composed of four levels (see Figure 1): factors, criteria, practices, and measures.

The three top levels of Squale use the standard mark system defined by the ISO 9126 standard. All quality marks take their value in the range $[0; 3]$, to support an uniform

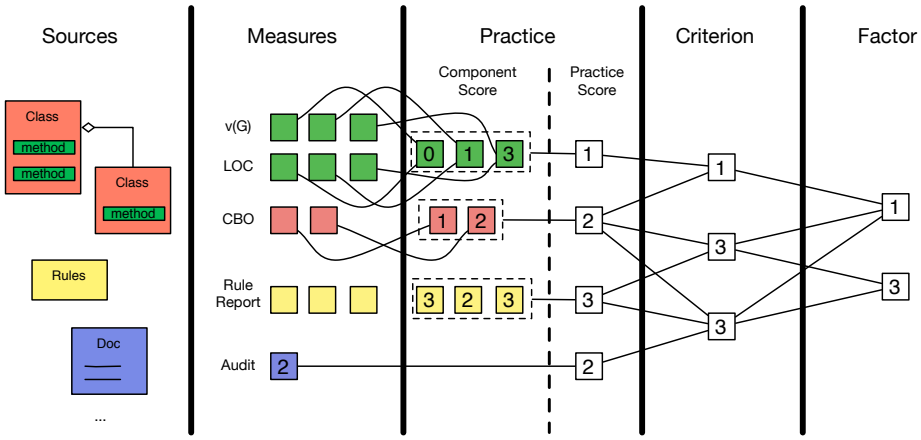


**Figure 1. Data sources and levels of the Squale model.**

interpretation and comparison. 3 represents an achieved goal, 0 represents a non achieved goal, 1 and 2 represent an achieved goal but with some reservations.

## 2.1 Measures

A *measure* is a raw information extracted from the project data.The Squale model takes into account different kinds of measure to assess the quality of a software project: automatically computable measures that can be computed easily and as often as needed, and manual measures which have a predefined life time and must be updated mainly after major changes to the software.

The automatically computable measures are divided into three groups. The first group is composed of metrics [5, 9, 3] such as Lines of Code [4], Hierarchy Nesting Level [7] or cyclomatic complexity [**?**]. A deep analysis selected only relevant metrics [1]. The second group is composed of rules checking analysis such as syntactic rules or naming rules, which verify that programming conventions are enforced in the source code. The third group is composed of measures which qualify the quality of tests applied to the project such as test coverage. This group may also contain security vulnerability analysis results.

The manual measures express human driven analysis performed during audits. These measures qualify the documentation needed for a project, such as specification documents or quality assurance plan. Around 100 different measures are used in various instances of the Squale model.

## 2.2 Practices

A *practice* assesses the respect of a technical principle in the project. However contrary to raw measures, a practice provides the possibility to stress a quality aspect such as *complex classes should be more documented than trivial ones*. In addition a practice is the place to adjust certain company quality conventions. A practice combines and weights different measures to assess the fulfillment of technical principles. The overall set of practices expresses rules to achieve optimum software quality from a developer's point of view: it provides a technical guidelines addressed to developers. It is directly targeted to developers in terms of good or bad property with respect to the project quality.

For example, the *comment rate* practice combines the *comment rate per method LOC* and *cyclomatic complexity* of a method: it expressed the quality concern that the more complex the method, the more comments it should have. Section 3 will detail this furthermore. Around 50 practices have been defined based on AF quality standards. However, the list of practices is not closed and the different practices can be adjusted to suit company requirements.

## 2.3 Criteria

A *criterion* assesses one principle of software quality (*safety, simplicity*, or *modularity* for example). It is addressed to managers as a detailed level to understand more finely project quality. Currently, the criteria used in the Squale model are adapted from ISO 9126 to face the special needs of AF and PSA. In particular, they are tailored for the assessment of quality in *information systems*.

A criterion aggregates a set of practices. A criterion marks is computed as the weighted average of the composed practice marks. Currently around 15 criteria are defined. For example, the *understanding* criterion is defined based on the following practices:

- comment rate (per method with respect to cyclomatic complexity)
- inheritance depth

- documentation achievement (human audit with respect to project requirements)
- documentation quality (rule checking of programming conventions)

### 2.4 Factors

A *factor* represents the highest quality assessment to provide an overview of project health (*functional capacity* or *reliability* for example). It is addressed to non-technical persons. A factor aggregates a set of criteria. A factor mark is computed as the average of the composed criteria marks.

The six factors used in the Squale model are inspired by the ISO 9126 factors and refined based on the experience and needs of engineers from PSA, AF, and Qualixo.

For example, the criteria Homogeneity, Understanding, Simplicity and Integration Capacity define the *capacity to correct* factor. This means that a system should be easier to correct when it is homogeneous (respect of architectural layers and of programming conventions for names), simple to understand and modify (good documentation, manageable size), and conveniently coupled.

## 3 Practices level

We now present in detail the practice layer and its specific aspects as it defines the backbone of the Squale model.

A global mark for a practice is computed in two steps:

**Individual mark.** Each element (method, class, or package in object-oriented programs) targeted by a practice is given a mark with respect to its measures. For example, the two metrics composing the *comment rate* practice, *cyclomatic complexity* and *source line of code*, are defined at the method level; thus a *comment rate* mark is computed for each method and the scope of this practice is *method*.

**Global mark.** A global mark for the practice is computed using a weighted average of the previous individual marks.

Weighting is chosen according to the project team's aims to highlight critical practices:

- a hard weighting is applied when there is a really low tolerance for bad individual marks in this practice. It accentuates the effect of poor marks in the computation of the practice mark.
- a medium weighting is applied when there is a medium tolerance for bad individual marks.
- a soft weighting is applied when there is a large tolerance for bad individual marks.

| Practice name | *Comment rate* |
|---|---|
| Definition | Qualify comments in lines of code rate. |
| Measures | Cyclomatic complexity: v(G)<br>and SLOC |
| Scope | Method |
| Individual mark | If $v(G) < 5$ and $sloc < 30$<br>then Imark = 3 else :<br>$Imark = \%\_comments\_per\_loc$<br>$/(1 - 10^{(-v(G)/15)})$ |
| Practice mark | $mark = -log_{weight}(average(weight^{-Imark}))$ |

**Table 1. The *comment rate* practice**

Table 1 summarizes elements that define the *comment rate* practice. Its definition— lines of code comments rate— determines which measures are used to compute its mark: the cyclomatic complexity and the number of lines of code. The scope defined for this practice is a method scope. The formulae used in the two steps mentioned above are regrouped in a set of functions for individual mark and a function to obtain the global practice mark from individual mark.

Using a formal definition which are tuned according to company standards, practices bridge the gap between the developer's point of view and the leader's point of view, or between a source code-oriented point of view and a higher-level one. A practice is considered as a quality element to respect or to avoid for developers. The overall practices constitute a guideline for the developers to correct their code and obtain a project following the quality standards of their company. The global practice mark represents a comparative reference for each individual mark and allows one to focus on low individual marks with respect to the practice. The mark computed for a class could be easily compared with the mark of the practice to determine if this class is in the average of the project or abnormally high— the weight applied to the metric possibly strengthening this abnormal result.

Furthermore, practices provide more information than simple metrics, as can we see with the *comment rate* practice. A low mark for this practice means that there are globally not enough comments in the source code. Moreover, due to the cyclomatic complexity metric used in the definition of this practice, individual marks give us more indications: we can determine which methods need to be more documented. Indeed, the methods with the lowest marks for this practice are not well commented with respect to their complexity—it is not the ones that simply have the lowest ratio of comment lines per lines of code.

Note that the actual set of practices used in the Squale model, along with the formula and weights, depends on the type of project and the quality standards for the company. For example, Air France does not use the same set of prac-

tices for its information system than PSA (although most are shared). The Squale model is customized for each project it is applied to.

## 4 Industrial evaluation

The Squale model was first designed by the Qualixo company and AF in 2006. After several months of experiments and validation of successive versions, they implemented the Squale quality platform. Since 2008, the Squale model is being reviewed by a French research consortium to enhance it [2] and the Squale quality platform is now released as open source software[1].

The validation of the Squale model is based on industrial feedback from AF and PSA. One hundred projects are currently monitored by Squale at AF, including business applications for freight or marketing, management applications for personnel management, or technical applications like frameworks. Of these hundred monitored projects, twenty are actively using it to improve their code base, which led to $6,000$ increased marks during one year. In total, Squale monitors about 7 MLOC with a frequency average of 130 audits per month.

The Squale software has also been in use by PSA for nearly one year. It monitors around $0.9$ MLOC dispatched in ten Java applications. The most important application, near 200 KLOC, supports the coordination of the flow of vehicles in factories.

In these companies, the Squale model is well accepted by developers as well as by managers which show interest in the model results. They noted an improvement of the quality for some projects but we cannot yet quantify this improvement, since the Squale project is still in an early stage of deployment from an industrial perspective.

## 5 Conclusions and Perspectives

This paper presents the Squale model for software quality. Our model is inspired by the ISO 9126 standard and introduces a new level for the assessment of practices in the hierarchy of factors, criteria, and measures.

It allows one to determine the quality of a project and control its evolution during the maintenance of a project, preventing deterioration. The Squale model stresses bad quality instead of averaging the quality in order to quickly focus on the wrong parts. It uses a set of measures combined into practices using formulae which take into account company standards and project technical specificity. Practice weights are customized with respect to these overall constraints.

[1] http://www.squale.org

Since 2008, the Squale project composed of the Qualixo company, the Paqtigo company, Air France-KLM, PSA Peugeot-Citroen, INRIA and University of Paris 8[2] aims at formalizing new practices and a Squale metamodel which would decrease the time spent to customize it. In future work we will study how the Squale model can be used to automatically describe remediation plans to increase the quality of a project. Such remediation plans should also assess the return on investment. The ultime goal is to provide strong arguments for managers dealing with quality process in their company.

## References


[1] F. Balmas, A. Bergel, S. Denier, S. Ducasse, J. Laval, K. Mordal-Manet, H. Abdeen, and F. Bellinguard. Software metrics for java and cpp practices, v1, http://www.squale.org/quality-models-site/, 2009.

[2] A. Bergel, S. Denier, S. Ducasse, J. Laval, F. Bellinguard, P. Vaillergues, F. Balmas, and K. Mordal-Manet. Squale – software quality enhancement. In *Proceedings of the 13th European Conference on Software Maintenance and Reengineering (CSMR 2009), European Projects Track*, March 2009.

[3] L. C. Briand, J. W. Daly, and J. Wüst. A Unified Framework for Cohesion Measurement in Object-Oriented Systems. *Empirical Software Engineering: An International Journal*, 3(1):65–117, 1998.

[4] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, June 1994.

[5] N. Fenton and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press, London, UK, second edition, 1996.

[6] ISO/IEC. Iso/iec 9126-1 software engineering -product quality- part 1: Quality model, 2001.

[7] M. Lorenz and J. Kidd. *Object-Oriented Software Metrics: A Practical Guide*. Prentice-Hall, 1994.

[8] R. Marinescu and D. Raţiu. Quantifying the quality of object-oriented design: the factor-strategy model. In *Proceedings 11th Working Conference on Reverse Engineering (WCRE'04)*, pages 192–201, Los Alamitos CA, 2004. IEEE Computer Society Press.

[9] R. C. Martin. Stability, 1997. www.objectmentor.com.

[10] J. McCall, P. Richards, and G. Walters. *Factors in Software Quality*. NTIS Springfield, 1976.



[2] This project is supported and labelled by the "Systematic - PARIS Region" competitive Cluster, and partially funded by Paris region and the DGE ("Direction Générale des Entreprises") in the context of the French Inter-ministerial R&D project 2006–2008 ("Projet R&D du Fonds Unique Interministériel").