An empirical model for continuous and weighted metric aggregation
Preprint of CSMR 2011

Karine Mordal-Manet[*], Jannik Laval[†], Stéphane Ducasse[†], Nicolas Anquetil[†],
Françoise Balmas[*], Fabrice Bellingard[‡] Laurent Bouhier[§], Philippe Vaillergues[‡] and Tom McCabe[¶]

[*]*LIASD, University of Paris 8, France*
[†]*RMoD Team, INRIA, Lille, France*
[‡]*Qualixo, Paris, France*
[§]*Air France-KLM, France*
[¶]*McCabe Software, US*

Abstract—It is now understood that software metrics alone are not enough to characterize software quality. To cope with this problem, most of advanced and/or industrially validated quality models aggregate software metrics: for example, cyclomatic complexity is combined with test coverage to stress the fact that it is more important to cover complex methods than accessors. Yet, aggregating and weighting metrics to produce quality indexes is a difficult task. Indeed certain weighting approaches may lead to abnormal situations where a developer increasing the quality of a software component sees the overall quality degrade. Finally, mapping combinations of metric values to quality indexes may be a problem when using thresholds. In this paper, we present the problems we faced when designing the Squale quality model, then we present an empirical solution based on weighted aggregations and on continuous functions. The solution has been termed the Squale quality model and validated over 4 years with two large multinational companies: Air France-KLM and PSA Peugeot-Citroen.

I. INTRODUCTION

Over the years, software metrics have been more and more considered to measure objectively software quality. As a consequence there is a plethora of tools computing metrics for quality assessment. However, these metrics are still often computed individually, for each software component, and they fail to give adequate quality indication at a higher level (entire system). To overcome this drawback and provide a more comprehensive and non-technical quality assessment, aggregation models have been created such as the ISO 9126 model [ISO01]. They do give an overview of the project's quality but they are often theoretical and difficult to compute. For example, as concluded by Hiyam Al-Kilidar [AKCK05], the ISO 9126 provides no guidelines or procedures to aggregate the metrics' results into an overall evaluation [Mar02], [MR04].

We highlight main issues with ISO 9126 based model:

- Most of quality models based on ISO 9126 [KLPN01], [BD02] give a high-level representation of quality by computing averages (simple or weighted average) of metrics. This is not satisfactory because, for example, it smooths results, possibly diluting a very bad results

in the overall acceptable quality [VSvdB10].
- To give a synthetic global mark, metric results are often translated into a discrete scale (ex: good, average, bad). This can hide minor changes in quality which in turn may discourage small, progressive improvements. Such method is not fine grained enough to be useful.
- Models and their construction processes are too theoretical and may frighten away professionals.
- Models do not take into account recommendations of enterprise or developer practical experience. The same note is applied regardless of the level of requirement of the enterprise for which the software was developed.

Such high-level models do not provide a useful view for developers: they do not take into account the level of criticality of an error nor are they able to emphasize certain technical aspects (like complex methods should be more covered by tests than trivial ones). Developers prefer to interpret metric individual results rather than their aggregation to determine which component(s) of the project must be improved.

In this paper we expose how the Squale Model introduces formulas to aggregate results that reflect a useful quality of the project under analysis. These formulas reflect metric values taking into account project and company specificity. Thresholds are set to determine what mark is acceptable or not. Formulas aggregate marks to pinpoint weaknesses and give a guide for future software improvements. The translation between metrics and range is continuous to avoid threshold effects. The Squale model has been first designed and put in production by Qualixo and Air France-KLM in 2006. Since then it has been put in production in several large french companies such as PSA Peugeot-Citroen. It is intensively used to monitor large projects, for a total of seven MLOC with the full cooperation of developers. The model has been open-sourced as the open-source Squale quality model.

The contributions of this paper are: (1) identification of problems when aggregating metrics, (2) proposition for advanced continuous and weighted metrics aggregation and (3) validation of the proposition in the context of large

industrial projects.

The paper is structured as follows. Section II gives an overview of problems encountered when aggregating metrics by simple or weighted average. Section III gives the context of Squale and definition used. Section IV presents in details the Squale computation solution with its formulas and discussing the originality of this method. Section V gives examples of metrics and compare results of different methods. Section VI reports the industrial validation of the Squale model in the context of a large project. Section VII discusses other approaches and Section VIII concludes with perspectives.

II. Problems with ISO 9126 based models

ISO 9126 has defined criteria and sub-criteria to give an overview of quality. To determine which characteristic is achieved, a mark is computed for each characteristic. To compute this mark, a high-level model is often based on software metric aggregation. A quality model follows three steps: (i) compute and collect metrics on components of the project, (ii) aggregate metrics to compute a high-level mark for the project, (iii) translate each high-level mark into a given range.

To execute these three steps, an ISO 9126 quality model must solve two issues:

- how to translate all metrics in the same range, to provide a unified understanding. Note that all marks with the same value have the same meaning. For example in the range [0;3] (as recommended in ISO 9126), all marks with the value 3 are considered as excellent.
- how to compute a high-level mark for a complete project with metrics based on several different components of the project (methods, classes, packages...).

For example, in ISO 9126 the *Changeability* sub-characteristic is defined as "The capability of the software product to enable a specified modification to be implemented". Metrics as number of lines of code (SLOC), cyclomatic complexity, number of methods per class or inheritance depth (DIT) [LK94], [FP96], [Mar97], [BDW98] are aggregated by different methods.

The most common approach is to use simple average (see Section II-B1) or weighted average (see Section II-B2) [BFNP98] to compute a high-level mark for a complete project. However, even if the average is the most obvious answer in translating low metrics to an overall high-level mark, it is not without pitfalls and we present another approach producing high-level quality model.

The third problem encountered with ISO 9126 based models is how to implement this theoretical model while maintaining an interest for developers. Developers often prefer to use raw metrics to evaluate their source code because of the inability to capture enterprise specific practices by quality models.

Table I
A discrete mapping example.

Normalized value	3	2	1	0
SLOC	≤ 35	$[35; 70]$	$[70; 160]$	≥ 160

The first part of this section describes the problem of translating mark and Section II-B describes problems encountered when averaging metrics.

A. Translating Mark in a Range

The simple approach to translate metrics to an appropriate range is to transform metrics to discrete marks within the selected range. Table I shows an example of this translation. A normalized value will be always 0, 1, 2 or 3.

A discrete marking system is simple to implement and easy to read but not adapted to all measures. This kind of translation is well adapted to translate human expertise like the existence and the quality of the specifications files of a project. But it is not adapted to translate metrics collected from the project like the number of Line of code for example or the cyclomatic complexity. Discrete mapping has the following drawbacks:

- *Hide modifications.* The discrete formula introduces staircase values and threshold effects, which hide detailed information and trigger wrong interpretation. When surveying the evolution of quality, it hides slight fluctuations – progression or regression – of an individual element.
  For example, according to Table I for a given project with methods of around 150 lines of code, each method has a normalized value of 1. If developers rewrite several methods to bring them around 80 lines of code, the quality of the project has really increased but the normalized values do not reflect this change.
- *Badly influence reengineer decision.* Working on components close to a quality threshold value provides more benefit to the overall quality than to work on components whose values are far from a threshold. Therefore, engineers can use this behavior to produce faster quality increase but at the cost of not fixing real problems. We saw this practice at Air-France where developers selected their tasks to maximize their impact on the quality marks.

To avoid these phenomena the Squale Model applies continuous formula to collected metrics (see Section IV).

B. Metrics Aggregation Background

To give a high-level representation of quality, the ISO 9126 model is based on internal metrics as described in its Part 3 [ISO03]. But it provides no indication or procedure to aggregate metrics into high-level marks. Simple or weighted average of collected metrics is often used by ISO 9126 based models to compute high-level mark.

Table II
TWO PROJECTS AVERAGED

Class	Project 1 # methods	Project 2 # methods
A	12	35
B	11	5
C	13	5
D	12	4
Avr	12.00	12.25

Table III
EXAMPLE OF WEIGHTS FOR SLOC.

Mark	≤ 35	$[35; 70]$	$[70; 160]$	≥ 160
Weight	1	3	9	27

Table IV
TWO PROJECTS AVERAGED

Method	SLOC	Weight
version 1		
A	30	1
B	50	3
C	70	9
D	300	27
Simple/Weighted Average	112.5	222.75
version 2		
A	25	1
B	30	1
C	50	3
D	300	27
Simple/Weighted Average	101.25	259.53

However, as we already reported it, averaging metrics is not completely satisfactory since it looses sensitivity as noticed by Bieman and others [Bie96], [SvdB10], [VSvdB10]. Let us review the exact problems.

1) Simple average: The problem encountered with high-level quality models is how to compute a global mark without loosing too much information on the individual components of a project. Computing the arithmetic mean of marks (*i.e.*, a simple average of each component mark) is not representative enough since it does not convey the standard deviation of the population as illustrated below — Note that there is nothing new here.

Table II presents the number of methods per class in two projects. In this example, the average is 13.75 for Project 1 and 12.25 for Project 2, it naively conveys the idea that the second project is better than the first, since its average is higher. This hides the fact that the second project has a class A which is clearly an outlier. Therefore while the mark is better, the quality of the project is probably lower. The average, because it smooths results, does not always represent reality [VSvdB10]. To give a more meaningful mark, a model must take into account his worse component(s) and reflect their differences. in the example, an appropriate quality indicator should highlight this single class by reporting a low aggregated mark. Measuring quality does not consist in computing a simple average but should also try to highlight application strengths and weaknesses. To be useful, a high-level quality model should be an assessment model but also a guideline to increase quality. A developer should know what component must be corrected and a manager should know if the project has problems. A simple average does not highlight bad component and even worse, it may hide very bad components. To remedy to these drawbacks, one could try to use weighted average, which has also some problems.

2) Weighted average: The main idea is to highlight bad components and to detect if there are critical components. Therefore the intuition is that the aggregation should raise an alarm by giving a bad global mark.

The first way to achieve this goal is to compute a weighted average. The weight applied to a given mark represents the influence of this mark compared to others. A first version of the Squale model computed its marks following the principle: the highest weight was applied to the worst mark to increase the importance of these marks.

As an illustration of problems related to naive weighted average, let us consider the following example: Table III shows the weights that have been defined for SLOC in the first version of Squale. It indicates that the worst mark of the SLOC measure has a weight of 27 to increase its representation in the average. Table IV contains the weighted corresponding SLOC measures. For example, weights applied are different for method C because the SLOC metric has a different result. What is striking in this example, is that while methods B and C got changes and their line of code reduced, the weighted average shows the opposite in terms of general quality. In this example, the weighted arithmetic mean for the version 1 is 222,75 and 259,53 for the version 2. The result increased while the code is globally better (lower SLOC for methods B and C). This example shows that naively using a weighted average can decrease the aggregation mark even though developers increased code quality. A quality model must reflect as closely as possible all improvements. Weighted average aggregation should not be applied because of lack of confidence in its result.

C. Marks and enterprise requirements

As noticed by Rosenberg [Ros98], when metrics are used to evaluate projects, there is no guideline to interpret their results. Often qualifying the result is based on common sense and experience. Determining what is an acceptable value depends on enterprise requirements and developer experience. For example, some companies require that depth of inheritance does not exceed a given threshold, while others focus on the general architecture or on use of naming standards. Therefore a high-level model must take into account enterprise specific practices and exigences and capture them in the quality model computation. It should try to give a useful measure of quality that managers as well as developers can use to take corrective actions.

III. SQUALE CONTEXT

To set the context of the continuous aggregation presented in this paper, we present the Squale software quality model [MMBD⁺09]. It is a quality model targeting developers as well as managers. To give a coherent answer to the different needs and audience, the Squale model is inspired from the factors-criteria-metrics model (FCM) of McCall [MRW76]. ISO 9126 is also derived from this FCM model.

A. Definitions

Throughout the paper we use different terms that we define now. The Squale Model is composed of four levels divided into two groups (Figure 1):

- High-level:
  - A *factor* represents the highest quality assessment to provide an overview of project health. It is addressed to non-technical persons and based on the Factors of the ISO 9126 Model.
  - A *criterion* assesses one principle of software quality. It is addressed to managers as a detailed level to understand more finely project quality. The criteria used in the Squale model are adapted to face the special needs of Air France-KLM and PSA Peugeot-Citroen. In particular, they are tailored for the assessment of quality in *information systems*.
  - A *practice* assesses the respect of a technical principle in the project. It is directly addressed to developers in terms of good or bad property with respect to the project quality. Good practices should be fulfilled while bad practices should be avoided. The overall set of practices expresses rules to achieve optimum software quality from a developer's point of view. Around 50 practices have been defined based on Air France-KLM quality standards. However, the list of practices is open and such practices can be adjusted.
- Low-level:
  - A *measure* is a raw information extracted from the project data. Measures provide raw metrics which are used to compute high-level marks, such as criteria or factor marks.

The low-level model contains low-level marks: results of metrics or rule reports for example. They can take any kind of value — they are called raw marks. The high-level model is built using high-level marks based on the low-level marks, they take their values into a given range — the range [0;3] for the Squale model, for example.

B. The architecture of Squale Model

The Squale model architecture is composed of (i) metrics and (ii) high-level criteria and factors (see Figure 1). Each computed metric gives a mark in its own range while criteria

and factors give a mark between 0 and 3 (in respect of the ISO 9126 model). Transforming raw marks into global marks in a given interval occurs in a new level between criteria and metrics introduced by the Squale model and called *practices*. Practices constitute the central point in the model and transform the low-level metric into high-level marks reflecting the quality of software.
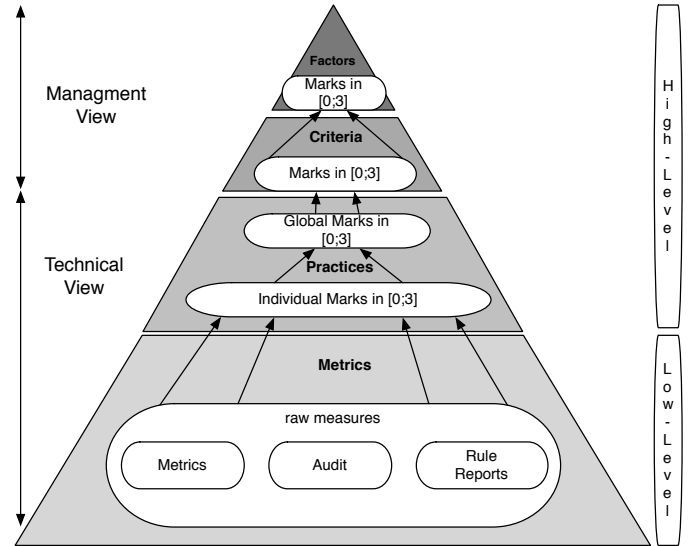

Figure 1. Data sources and levels of the Squale model.

The three top levels of Squale use the mark system defined by the ISO 9126 standard. Each quality mark takes its value in the range $[0;3]$, as shown in Figure 1, to support an uniform interpretation and comparison: $[0;1]$ the goal is not achieved; $[1;2]$ the goal is achieved but with some reservations; $[2;3]$, the goal is achieved.

IV. MARK COMPUTATION IN SQUALE

This section explains how the Squale model resolved the problems introduced in Section II and how the model computes high-level quality marks in its practice level.

The model collects different raw measures and translates them into high-level marks in the [0;3] range. To compute them, it does not use simple average and discrete functions but computes marks using continuous functions. This computation is made in the Practice Level in two steps that will be explained in detail in subsequent sections.

- *Individual marks.* Each element (method, class, or package) 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. This mark takes into account the weight of one metric over other and associated metrics and produces an Individual Mark for each project

components. This individual mark is given in the range [0;3] to enable comparison between practices on a common scale.

- *Global marks.* A global mark of a practice is computed using all the individual marks associated to a continuous weighted function.

A. Individual Mark

An individual mark is computed from measures in multiple ranges into a single mark in the range $[0;3]$.

As described previously, discrete marking is not adapted to metrics-based practices. For this reason a continuous formula is used when it is possible. It better translates the variations of metric values on the mark scale. We also need to tailor the formula to the specific needs of the company. Therefore the formula is first built around some measure-mark binding, agreed upon by the experts, by defining the equation (linear or not) which best approximate those special values and allows one to interpolate marks for other values. The example of *Method Size* practice illustrates this.

This practice is computed at the method level, *i.e.*, the individual practice mark is computed for each method and the global practice mark is computed with a weighted function and a medium weight. This practice needs the SLOC (number of source lines of code) metric to be computed. It is defined as the practice to highlight methods which are too long, hence too difficult to maintain and understand. To compute the Individual Mark for this practice, pairs of measure/mark bindings have been determined and are given in Table V. From this, the formula to obtain the individual mark is derived: $IM = 2^{(70-sloc)/21}$. This formula has been validated by Air France-KLM developers: thresholds are defined with the requirements of Air France-KLM. Each instantiation of the Squale model can change such thresholds to fit their needs.

Table V
MEASURE/MARK BINDINGS TO FIND THE SLOC QUALITY FORMULA IN ONE COMPANY

SLOC	≤ 37	42	49	58	70	91	≥ 162
Practice	3	2.5	2	1.5	1	0.5	0

Figure 2 shows the curve that corresponds to this function. First there is a threshold of 40 below which the mark is set to 3. It is the maximal value which allows one to achieve the goal. Above this threshold, the individual mark decreases following an exponential curve: the individual mark tends quickly towards zero.

Exception to continuous aggregation: In the Squale model, a discrete marking system is applied for *manual measures* such as audits. For example, the practice for *functional specifications* is given a mark in a discrete range. If there is no functional specification, the mark 0 is given. If functional specifications are consistent with the client
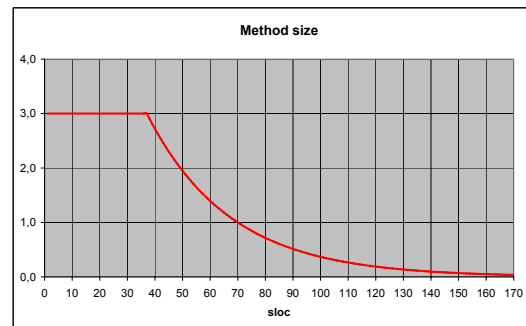

Figure 2. Method size curve weighting function

requirements, the mark 3 is given. The two intermediate marks 1 and 2 are used to qualify existing yet incorrect functional specifications. Thus this mark assesses two pieces of information: the existence of functional specifications and their consistency. While the practice can only be evaluated by an expert, the discrete range limits the subjectivity of the given mark.

B. Global Marks

Each component of the project has an individual mark. These marks are aggregated to produce a Global Mark: a mark for a given practice at the project level.

The global practice mark is obtained from the individual marks through a weighted aggregation. The weighting function allows one to adjust individual marks for the given practice in order to stress or loosen tolerance for bad marks. This allows to highlight critical practices: hard weighting leads to a low practice mark much faster than soft weighting:

- 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. The global mark falls in the range $[0;1]$ as soon as there is a few low individual marks.
- a medium weighting is applied when there is a medium tolerance for bad individual marks. The global mark falls in the range $[0;1]$ only when there is an average number of low individual marks.
- a soft weighting is applied when there is a large tolerance for bad individual marks. The global mark falls in the range $[0;1]$ only when there is a large number of low individual marks.

The computation of the practice mark is a two-step process. First a weighting function is applied to each individual mark: $g(IM) = \lambda^{-IM}$ where IM is the individual mark and λ the constant defining the hard, medium, or soft weighting, λ being greater for a hard weighting and smaller for a soft one. This formula translates individual marks into a new

space where low marks may have significantly more weight than others. The average of the weighted marks will reflect the more important weight of the low marks. Then the inverse function: $g^{-1}(Wavg(IMs)) = -log_\lambda(Wavg(IMs))$ is applied on the average to come back in the range $[0;3]$. The *Wavg(IMs)* is the weighted average for Individual Marks.

Thus the global mark (*i.e.*, taking into account n components) for a practice is: $mark = -log_\lambda\left(\frac{\sum_1^n \lambda^{-IM_n}}{n}\right)$ where λ varies to give a hard, medium, or soft weighting.
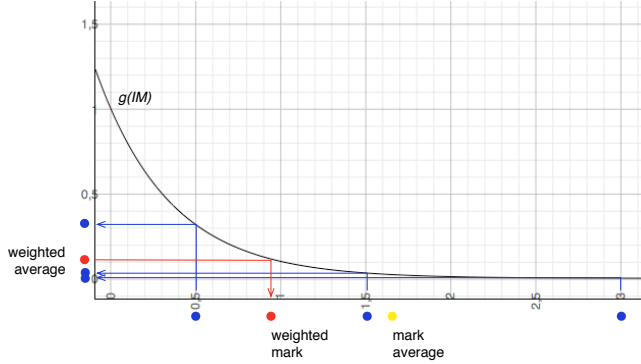
Figure 3. Principle of weighting: individual marks are lowered when translated in the weighted space.

Figure 3 illustrates how the $g(IM)$ function and its inverse work to reflect low individual marks in the practice mark. Here, $\lambda = 0$, which is a medium weighting. There are three individual marks (blue dots on the x axis) at 0.5, 1.5, and 3. This series would give an average around 1.67 (yellow dot). Instead, the marks are translated in the weighted space (blue arrows) where the 0.5 mark is significantly higher than the two other marks. The weighted average (red dot on y axis) is then translated back in the mark range (red arrow) with a final value of 0.93. The lower weighted mark for the practice, compared to the normal average, is a clear indication that something is wrong, despite one good Individual Mark of 3.

V. EXAMPLES OF PRACTICES

This section gives examples to explain how Squale computes global quality marks.

A. Method Size

Table VI presents an example with different methods to compute a mark for the *Method Size* practice. The first column gives the SLOC metric. The second gives the translation of the SLOC metric in the range [0;3] as explained in Section II-A for each method. The third gives a weight (and "classical" weighted average on last line) associated to each individual mark as explained in Section II-B2. The last column gives the resulting Squale individual mark. The last line of this table gives the global marks associated to

Table VI
TWO PROJECT VERSIONS AVERAGED

Method	SLOC metric	[0;3] range	Weighted average	Squale mark
Version 1				
A	30	3	1	3
B	50	2	3	1,93
C	70	1	9	1
D	300	0	27	0
Global mark	112.5	1.5	0.54	0.397
Version 2				
A	25	3	1	3
B	30	3	1	3
C	50	2	6	1,93
D	300	0	27	0
Global mark	101.25	2	0.34	0.407

Table VII
MAIN VALUES FOR INDIVIDUAL MARKS OF *Efferent Coupling* PRACTICE

Ce	≤ 6	7	8	9	10	12	≥ 19
Practice	3	2.8	2	1.4	1	0.5	0

each type of computation. The first global mark is a simple average of the SLOC metric, the second is a simple average of the high-level mark, the third is a weighted average of the marks previously computed associated to the determined weight and the last global mark is computed as explained in Section IV-B.

Table VI shows one of the problems of simple average: from version 1 to version 2, the average SLOC decreases from 112.5 to 101.25. The discrete simple average —second column— grows from 1.5 to 2 because of a threshold effect (difficult to completely avoid with discrete value). However, according to the criteria of Air France-KLM, a project with a 300 lines (of code) method cannot be accepted. Something that the average failed to highlight. The weighted average decreased while the quality increased (since three methods are shorter) as already explained in Section II-B2 so this weighted average is not useful. The Squale final mark increases slightly to reflect the improvement but still highlight the problem of method D with 300 lines: the quality of the project has increased but is still unacceptable for this company.

B. Efferent Coupling

This practice is computed for each class of a project and is based on the Ce (efferent coupling) metric. It qualifies the efferent coupling for a class and analyzes the dependency between one class and the other classes as well as the public data of the project. A class which uses many other classes should be potentially more affected by any other class modifications. This practice highlights the most dependent classes.

To compute the individual marks of this practice, thresholds have been determined according to Air France-KLM requirements and are given in Table VII. The corresponding

Table VIII
DIFFERENT AVERAGE FOR TWO PROJECT VERSIONS

Class	Ce metric	[0;3] range	Weighted average	Squale mark
Version 1				
A	3	3	1	3
B	7	2.8	3	2
C	9	1.4	9	1
D	12	0.0	27	0.5
global mark	7.75	1.8	0.60	0.86
Version 2				
A	3	3	1	3
B	6	3	1	3
C	8	2	3	2
D	20	0	27	0
global mark	9.25	2	0.47	0.44

formula to obtain the individual mark is : $IM = 2^{(10-ce)/2}$. Figure 4 shows this function.
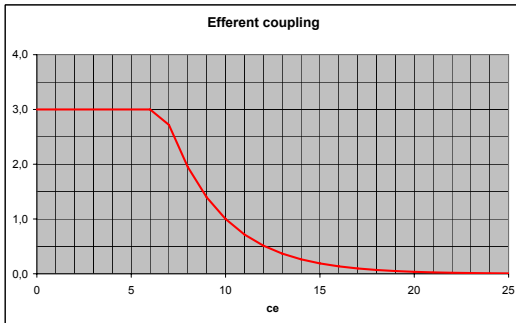
Figure 4.   Efferent coupling Individual Mark function.

Table VIII presents an example with different classes to compute the mark of this practice. The first column gives the Ce metric. The second gives the translation of the Ce metric in the range [0;3] as explained in Section II-A for each class. The third gives the weights applied to each mark as explained in Section II-B2. The last column gives the Squale individual mark. The last line of this table gives the global marks obtained with each type of aggregation.

In this table, the different global marks can be interpreted differently. Ce metric average shows a slight difference in disfavor of the first version. The simple average reduced to the range [0;3] increases slightly from one version to the other and the weighted average decreases slightly. However, the Squale global mark varies significantly enough to highlight that class D worsened a lot. The simple average gives a mark near 2, which means that the project is acceptable. However, the low rating of class D was overshadowed by the results of other classes. The weighted average stressed bad mark but this result is not satisfactory enough due to its too small variation. The Squale global mark gives both a

mark that reflects the class D but further refines the results between the two versions.

VI. SQUALE IN ACTION

A. Protocol

The Squale model was first designed by the Qualixo company and Air France-KLM in 2006. In the first version, measures collected were translated in the range [0; 3] using a discrete function. The thresholds introduced by this method resulted in negative feedback of the industrial, because practices did not reflect the state of the system as explained in Section II-A. The second version of the Squale model introduced weighted average to highlight bad component. This version was not successful either because of the paradox described in Section II-B2. The current Squale model [MMBD+09] introduces continuous formulas described in Section IV. We have used this version to develop the Squale software which have been deployed in several french companies.

B. Industrial validation

Figure 5 shows a Squale report for an industrial project at PSA Peugeot-Citroen. Each factor is detailed with: its marks for the previous evaluation; its mark for the current evaluation; a meteorologic symbol which gives a symbolic meaning to the mark and an arrow whose direction indicates the change with respect to the previous evaluation.

The validation of the Squale model is based on industrial feedback from Air France-KLM and PSA Peugeot-Citroen. One hundred projects are currently monitored by Squale at Air France, 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 Squale to improve their source code, which led to 6, 000 increased marks during one year. On the whole, Squale monitors about seven MLOC.

The Squale software has also been used at PSA Peugeot-Citroen for two years. The first year, it monitored about 0.9 MLOC dispatched in ten Java applications. Currently, it realizes around 640 audits and monitors about 10 MLOC dispatched in 90 Java applications with 350 modules. It is deployed systematically with the following rules:

- Each team in PSA Peugeot-Citroen determines its own requirements.
- The Squale model is adapted to reflect these requirements.
- Projects are audited with Squale which determines whether they conform to rules.

In these companies, the Squale software is well accepted by developers as well as managers who show interest in the model results. The PSA Peugeot-Citroen managers support and sponsor it. Squale is becoming part of the PSA Peugeot-Citroen software engineering process.

Facteur	Note précédente	Note	Evolution
Architecture	3.0 ☀	3.0 ☀	→
Capacité fonctionnelle	-	-	-
Evolutivité	2.2 ☀	2.2 ☀	→
Maintenabilité	2.5 ☀	2.5 ☀	→
Fiabilité	1.8 ⛅	1.8 ⛅	→
Réutilisabilité	2.1 ☀	2.1 ☀	→

Figure 5.   The Squale audit view (column titles: Factor; Previous mark; Mark; Evolution).

Table IX
MARK FOR **REUSABILITY** FACTOR

Criteria Name	Mark
Comprehension	3.0
Exploitability	3.0
Internal dependency	2.5
Technical tests	0.1
global mark	2.1

Table X
INDIVIDUAL MARK FOR **EFFERENT COUPLING** PRACTICE

Component Name	Individual Mark	efferent coupling: ce
VehicleMakeAction	3.0	6
ImportAction	3.0	5
PSAPAction	3.0	5
ExportAction	3.0	4
SpCsPerimeterAction	3.0	4
AdminParameterAction	3.0	3
VehicleModelAction	3.0	3
IsAliveAction	3.0	3
CompanyAction	3.0	2
SelectCompanyAction	3.0	1
global mark	3.0	3.6

Table XI
INDIVIDUAL MARK FOR **VEHICLEMAKEACTION** CLASS

Practice Name	Mark
inheritance depth	3.0
efferent coupling	3.0
class cohesion	2.0
swiss army knife	3.0
afferent coupling	3.0
class size JSP	3.0
number of method	3.0

Metric Name	result
class cohesion	22.0
afferent coupling	0.0
number of method per class	13.0
number of derived class	0.0
number of public methods	12.0
inheritance depth	2.0
coupling between class	6.0
number of accessing methods	40.0
lines test code coverage	0.0
branch test code coverage	0.0
cyclomatic complexity max	9.0
somme of cyclomatic complexity	21.0
number of methods	12.0
number of javadoc	9.0
number of classes	0.0
number of source lines of code	54.0

C. One audit in PSA Peugeot-Citroen

For confidential reason, we cannot provide detailed industrial information about source code and Squale results. Figures used in this section come from an audit at PSA Peugeot-Citroen.

Table IX shows marks for the *Reusability* Factor which evaluates whether code can be reused in other contexts. This factor is composed of 4 criteria, each one composed by several practices. When computing the Factor mark, Squale offers the possibility to weight criteria marks if the enterprise wants to use weighted average. Computing criteria marks follows the same principle: each criteria is decomposed in several practices which are then aggregated as described earlier.

Table X contains the results of *Ce* metric and the associated marks to each project component. In this case, results of *Ce* metric are all acceptable: each individual mark is 3 and the global Mark is at the maximum.

Table XI contains results of a given component: the metrics raw values and individual practice marks that were derived from these metrics. Metrics used include both oriented-object metrics like DIT, traditional metrics like cyclomatic

complexity but also test coverage measure. Note that a metric can be also as a threshold: the *number of methods* practice uses the cyclomatic complexity to determine how is computed its mark.

Table XII contains severals marks for one method. There are two practices named *spaghetti code* because of the difference needed by PSA Peugeot-Citroen between JSP and Java. The *spaghetti code* practice is computed for all methods which have a *SLOC* metric higher than 30. The *process* method obtains a poor *spaghetti code* mark because the value of its cyclomatic complexity is too high. It highlights that this method would be too difficult to maintain.

D. Threats to validity

The validation of the Squale model is dependent on the requirements of industrial partners that apply it. It is based on its implementation and deployment inside Air France-KLM and PSA Peugeot-Citroen in the context of information system analysis and evaluation. To manage this threat and to understand the possible variations within the model, a

Practice Name	Mark
spaghetti code JSP	0.1
method size	3.0
spaghetti code	0.2
Acceptance test code coverage	0.0

Metric Name	result
code coverage	0.0
The code coverage per branch	0.0
cyclomatic complexity	12.0
javadoc number	1.0
number of line of source code	33.0

database with anonymous data is under construction by Qualixo and its partners. Such database will makeit possible to compare large amount of data gathered from different projects and domains. The Squale model should be applied to other kind of projects to determine if the aggregating functions used for computing marks can be easily tuned and are relevant in other domains.

The model has been in production for a couple of years already and got improved during two revisions. Quality experts manually validated the models and the results in production. Now it would be scientifically good to have a second validation phase based on manual audits and controlling the process and variable. The idea would be to compare expert opinion with results given by the model.

VII. DISCUSSION

Hierarchic quality models like the ISO 9126 Standard [ISO01] or the McCall model [MRW76] give an overall quality assessment of a system but they do not describe enough the low-level details and metrics needed to qualify this quality: they clearly lack the connexion with source code. Another difficulty with these models is that they fail to translate the influence of individual components. The Squale model keeps the advantage of providing a quality overall view but it brings a new dimension to it by keeping all the details. Practices give at the same time the quality of the project and the way to improve its quality.

GQM (Goal-Questions-Metrics) is an approach to software quality that has been promoted by Basili [BCR94]. It defines a measurement model on three levels: Conceptual level (the Goal level), Operational level (the Question level) and Quantitative Level (the Metrics level). GQM or FCM have been defined as top-down models to express the quality of a system, from the requirements to the metrics that allow one to measure them. It implies that measuring the quality can only start after the model has been completely specified, and the first results must wait until sufficient data has been collected. The Squale model promotes a bottom-up approach, aggregating low-level measures into more abstract quality elements. This approach ensures that the computation of top-level quality marks is always grounded by concrete repeatable measures or audit on actual project components. The Squale model supports quality assessment as far in the past as the available source code.

The Quality Model for Object-Oriented Design (QMOOD) model is also a hierarchic model based on ISO 9126, but it is specialized in object-oriented design and does not take into account the quality of low level implementation or if programming style rules are respected. The Squale Model gives a more complete view of quality and proposes ways to increase it.

Marinescu and Ratiu [MR04] raised the following question: *How should we deal with measurement results?* and proposed to link quality factor to source code entities using detection strategies. They introduced detection strategies [Mar04] for analyzing a source code model using metrics. This model named *Factor-strategy* is relevant to measure object-oriented design but as the QMOOD model, it does not define the overall quality of a project. The adaptability of the Squale model allows one to qualify any paradigm and practices provide a complete view of quality.

Recently Vasilescu [VSvdB10], Vasa [VLBN09] and Serebrenik [SvdB10] started to use different aggregation functions such as the Gini coefficient or Theil index. This is clearly the sign that software metrics require more advanced aggregating functions.

VIII. PERSPECTIVES

This paper presents how the Squale model has solved practical issues of average computation. Our model, inspired by the ISO 9126 standard, introduces continuous formulas to compute high-level marks.

It allows one to determine the quality of a project and to control its evolution during the maintenance of a project, preventing its deterioration. Instead of averaging the quality, the Squale model stresses bad quality in order to quickly focus on the wrong parts. It uses a set of measures grouped in practices using formulas which take into account organizations' standards and project technical specificities. Air France-KLM and PSA Peugeot-Citroen have validated their own instances of the Squale model to monitor different information systems.

Since 2008, the Squale project is composed of the Qualixo company, the Paqtigo company, Air France-KLM, PSA Peugeot-Citroen, INRIA and University of Paris 8[1]. After formalizing practices and metamodel of Squale [BBD+10], the team will improve the model. One approach consists in determining if all practices are really relevant for developers and if some practices are redundant. We also want to know whether practices are more or less useful

depending on the point in the life-cycle of the system one performs the audit. Another research axis is to determine how to measure more accurately the packages quality and what kind of metrics to use for it.

Finally, we are studying how to improve the remediation plans based on the Squale model. Such remediation plans should also assess the return on investment. The final goal is to provide strong arguments for managers dealing with quality process in their company. In future work we will also check which measures and practices are needed for other domains than information systems.

REFERENCES

[AKCK05] H. Al-Kilidar, K. Cox, and B. Kitchenham. The use and usefulness of the iso/iec 9126 quality standard. In *International Symposium on Empirical Software Engineering*, nov 2005.

[BBD⁺10] Francoise Balmas, Alexandre Bergel, Simon Denier, Stéphane Ducasse, Jannik Laval, Karine Mordal-Manet, H. Abdeen, Fabrice Bellinguard, and Bertrand Franchet. The squale quality model, v2, http://www.squale.org/quality-models-site/, 2010.

[BCR94] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. The goal question metric approach. In *Encyclopedia of Software Engineering*. Wiley, 1994.

[BD02] Jagdish Bansiya and Carl Davis. A hierarchical model for object-oriented design quality assessment. *IEEE Transactions on Software Engineering*, 28(1):4–17, January 2002.

[BDW98] Lionel C. Briand, John W. Daly, and Jürgen Wüst. A Unified Framework for Cohesion Measurement in Object-Oriented Systems. *Empirical Software Engineering: An International Journal*, 3(1):65–117, 1998.

[BFNP98] G. Bucci, F. Fioravanti, P. Nesi, and S. Perlini. Metrics and tool for system assessment. In *Proceedings of IEEE Conference on Complex Computer Systems. USA: IEEE Publ*, pages 36–46, 1998.

[Bie96] James M. Bieman. Metric development for object-oriented software, 1996.

[FP96] Norman Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. International Thomson Computer Press, London, UK, second edition, 1996.

[ISO01] ISO/IEC. Iso/iec 9126-1 software engineering - product quality- part 1: Quality model, 2001.

[ISO03] ISO/IEC. Iso/iec 9126-3 software engineering - product quality- part 3: Internal metrics, 2003.

[KLPN01] Barbara Kitchenham, Steve Linkman, Alberto Pasquini, and Vincenzo Nanni. The squid approach to defining a quality model. *Software Quality Journal*, 6(3):211–233, 1997-09-01.

[LK94] Mark Lorenz and Jeff Kidd. *Object-Oriented Software Metrics: A Practical Guide*. Prentice-Hall, 1994.

[Mar97] Robert C. Martin. Stability, 1997. www.objectmentor.com.

[Mar02] Radu Marinescu. *Measurement and Quality in Object-Oriented Design*. PhD thesis, Department of Computer Science, Politehnica University of Timişoara, 2002.

[Mar04] Radu Marinescu. Detection strategies: Metrics-based rules for detecting design flaws. In *ICSM'04*, pages 350–359, Los Alamitos CA, 2004. IEEE CS Press.

[MMBD⁺09] Karine Mordal-Manet, Françoise Balmas, Simon Denier, Stéphane Ducasse, Harald Wertz, Jannik Laval, Fabrice Bellingard, and Philippe Vaillergues. The squale model – a practice-based industrial quality model. In *ICSM '09*, pages 94–103, 2009.

[MR04] Radu Marinescu and Daniel Raţiu. Quantifying the quality of object-oriented design: the factor-strategy model. In *WCRE'04*, pages 192–201. IEEE CS Press, 2004.

[MRW76] Jim McCall, Paul Richards, and Gene Walters. *Factors in Software Quality*. NTIS Springfield, 1976.

[Ros98] Linda H. Rosenberg. Applying and interpreting object oriented metrics, Software Technology Conference (Utah - April 1998).

[SvdB10] Alexander Serebrenik and Mark van den Brand. Theil index for aggregation of software metrics values. In *Proceedings of ICSM 2010*, 2010.

[VLBN09] Rajesh Vasa, Markus Lumpe, Philip Branch, and Oscar Nierstrasz. Comparative analysis of evolving software systems using the Gini coefficient. In *ICSM 2009*, pages 179–188. IEEE Computer Society, 2009.

[VSvdB10] Bogdan Vasilescu, Alexander Serebrenik, and Mark van den Brand. Comparative study of software metrics' aggregation techniques. In *Proceedings of the International Worskhop Benevol 2010*, 2010.