Delft University of Technology
Software Engineering Research Group
Technical Report Series

Towards a Catalog Format for Software Metrics

Eric Bouwers, Joost Visser and Arie van Deursen

Report TUD-SERG-2014-004

TUD-SERG-2014-004

Published, produced and distributed by:

Software Engineering Research Group
Department of Software Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

ISSN 1872-5392

Software Engineering Research Group Technical Reports:
`http://www.se.ewi.tudelft.nl/techreports/`

For more information about the Software Engineering Research Group:
`http://www.se.ewi.tudelft.nl/`

Note: Accepted for publication at the 5th International Workshop on Emerging Trends in Software Metrics (WETSoM 2014)

Towards a Catalog Format for Software Metrics

Eric Bouwers
Software Improvement Group
Amsterdam, The Netherlands
e.bouwers@sig.eu

Arie van Deursen
Delft University of Technology
Delft, The Netherlands
Arie.vanDeursen@tudelft.nl

Joost Visser
Software Improvement Group
Amsterdam, The Netherlands
j.visser@sig.eu

ABSTRACT

In the past two decades both the industry and the research community have proposed hundreds of metrics to track software projects, evaluate quality or estimate effort. Unfortunately, it is not always clear which metric works best in a particular context. Even worse, for some metrics there is little evidence whether the metric measures the attribute it was designed to measure.

In this paper we propose a catalog format for software metrics as a first step towards a consolidated overview of available software metrics. This format is designed to provide an overview of the status of a metric in a glance, while providing enough information to make an informed decision about the use of the metric. We envision this format to be implemented in a (semantic) wiki to ensure that relationships between metrics can be followed with ease.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics

General Terms

Documentation, Measurement

Keywords

Software Metrics, Catalog Format, Information Overview

1. INTRODUCTION

Software metrics are a popular tool to track the progress of a project, to determine the quality of a software product or process, or to evaluate a software development team. Over the past decades, literally hundreds of software metrics have been proposed, in varying level of detail and accompanied with different types of validation [12].

In most cases, the format in which a (suite of) software metrics is introduced is ad-hoc, differentiating in style, format, and notation. Moreover, the terminology used to introduce software metrics is not consistent, and the discussion of aspects of software metrics are often incomplete.

Given these diverse descriptions, it is hard to answer questions such as: "does the metric quantify what I want to measure?", "which side-effects has the introduction of this metric?", or "is it normal that this metric changes so much?". Because of this, it is challenging for both practitioners and researchers to choose the most appropriate metric(s) for the task at hand.

In order to assist practitioners and researchers to select the most appropriate set of metrics for their current task it is needed to define a catalog of software metrics which encodes, amongst others, the definition, the benefits, and the limitations of software metrics [1, 13].

As a first step towards such a catalog, this paper proposes a *Software Metric Catalog Format (SMCF)*, containing 16 fields separated into two columns. The purpose of this format is to provide a concise overview of a software metric, while containing enough information to make an informed decision about whether to use a software metric for a certain task.

The remainder of this paper is structured as follows. Section 2 discusses related work and the sources of inspiration for the proposed format. After this, Section 3 outlines the format and explains its fields. Section 4 discusses the format and explains how we intend to implement the format, after which Section 5 outlines the next steps.

2. BACKGROUND

The categorization of software metrics has long been a topic of interest and considerable effort has been made to create overviews of software metrics based on structured literature reviews. The result of these efforts are, for example, overviews of object-oriented metrics [12] or architecture metrics [8]. In addition, more qualitative overviews discussing the types of validations of software metrics have been performed as well [7].

Unfortunately, these overviews do not provide enough information to make a decision about which metrics to use in a specific situation. As Kaner et al. [6] point out, there are several questions that need to be answered before this decision can be made. These questions address basic properties of the metric such as "what attribute are we trying to measure?", up until more social questions such as "what are the natural and foreseeable side effects of using this instrument?" [6].

Especially this last question is often not considered in the introduction of a software metric, nor in the overview studies. However, according to our experience this is an important question to answer. One of the most common pitfalls of using software metrics is the situation in which changes are being made purely to satisfy a metric value [2].

Closely related to our proposal is the template as proposed by Olsina et al. [11]. This template contains 29 fields including the name, the objective, related metrics and potential beneficiaries. In our opinion, this template is not sufficient because it does not contain a specific field dedicated to explaining the effects of using the metric, nor does it contain a space to explain the variability of the metric. In addition, some fields are considered to be too detailed for a first glance, a topic we discuss in Section 4.

Lastly, a recent proposal of Saraiva et al. [13] outlines several domains in which software metrics can be categorized. Based on these domains a catalog of metrics can be generated. We see the SCMF as one candidate to describe the metrics in such a generated catalog.

3. SOFTWARE METRIC CATALOG FORMAT

The catalog format we propose consists of 16 fields divided over two columns and is displayed in Figure 1. The left-hand side of the format contains qualitative information about the metric such as the name, the insights it can provide and possible solution strategies. The right-hand side of the catalog is reserved for more quantitative information such as the expected value of the metric, the expected range and the relationship towards other metrics.

3.1 Qualitative fields

The left-hand side of the format consists of eight fields which contain qualitative information about the metric. The combination of these fields should enable a reader to understand *what* the metric is, *how* it is measured, *why* this is an interesting metric, and the *actions* which should (or should not) be taken to react on a metric value in a particular *context*. Lastly, it should help a reader to identify situations in which actions are taken solely to treat the value of the metric.

Name: The full name of the metric accompanied by its abbreviation. When multiple metrics share the same name (or abbreviation) a postfix, for example the name of the author of the metric, is added. An example here is "Duplicated lines".

Entity, Attribute: A metric quantifies a certain attribute of an entity [4]. For example, "Duplicated lines" measure the "duplication" (attribute) of a "Software System" (entity).

Definition: In order to calculate a metric one or more steps need to be taken to transform the input (the entity) to a number. In some cases this can simply be a mathematical formula describing the transformation on numbers, while in other cases a more detailed description is needed. Since the aim of this format is to provide a condensed overview, the level of detail should be carefully chosen. For example, a naive definition of "Duplicated Lines" could be: *count all lines which appear more than once in the system.*

Rationale (theoretical): In some cases it is easy to understand why the definition given above quantifies the desired attribute, e.g. the definition given above for "Duplicated Lines" is a straight-forward implementation. In other cases the relationship might not be immediately clear and should be explained in more depth in this field.

Implications (practical): The fields above provide the necessary information to understand *what* the metric quantifies and *how* this is done. This field must provide the information towards *why* this metric is interesting to use. For example, the metric "Duplicated Lines" is interesting because duplication in a system is considered to be an indication of poor design [9]. Thus, it is desirable to have a low value for this metric.

Applicable in context: A metric can be useful in general, or can only be applied in certain contexts. This field is used to describe only those contexts in which it is useful, or to outline contexts in which the metric has less added value. For example, "Duplicated lines" is applicable in a wide range of contexts, but in a migration project with a well-defined end-date a higher value of "Duplicated lines" might be acceptable.

Solution strategies: This field offers one (or more) solution strategies to reach a more desired value of the metric. Each strategy consists of a short description and is marked as either *treating* or *solving*. The first category indicates strategies in which changes are made only to alter the value of the metric without addressing the underlying cause [2]. The second category marks more desirable strategies.

For example, a strategy to reduce the number of "Duplicated lines" is to rename variable or switch the ordering of statements. Since this does not address the root-cause of the higher value this strategy is marked "treating". Alternatively, "Duplicated lines" can be reduced by abstracting out common pieces of functionality in dedicated units or by introducing a library, both of which are "solving" strategies.

3.2 Quantitative fields

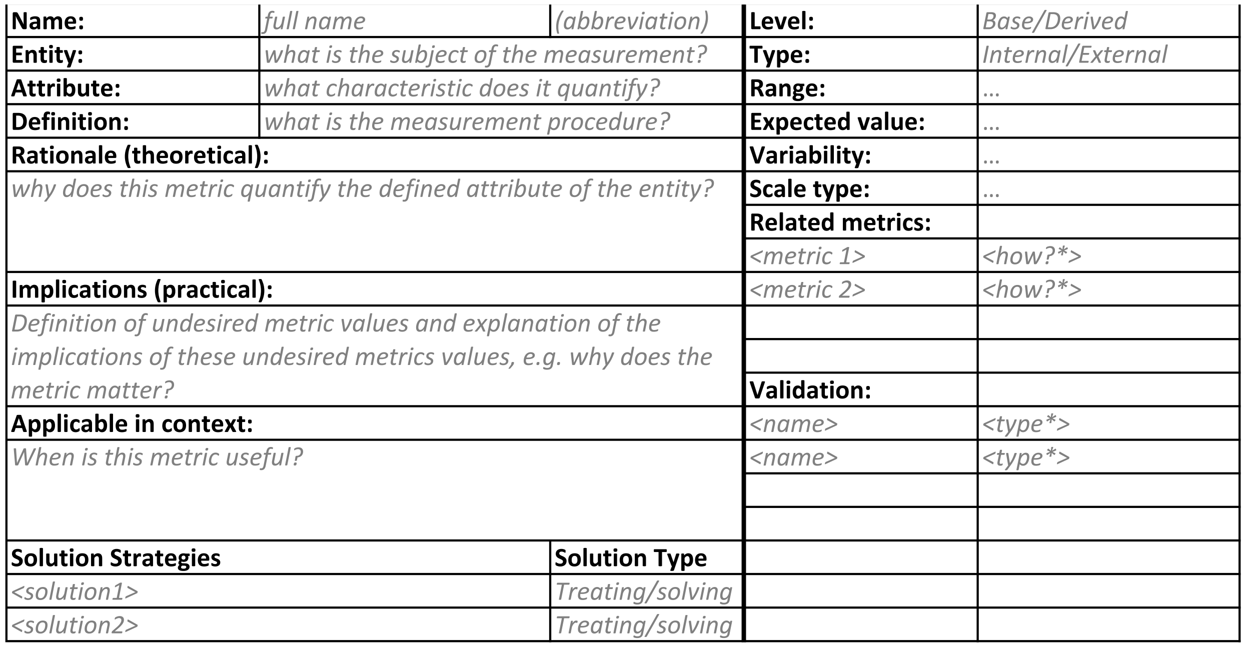
The right-hand side of the catalog is reserved for quantitative information about the metric. Together, these fields should enable a reader to place the information on the left-hand side in perspective.

Level: This field indicates whether the metric is either a *base* metric (cannot be expressed in other metrics), or a *derived* metric, i.e., a combination of other metrics. Our example metric "Duplicated lines" is a *base* metric, while a metric such as "Duplicated lines percentage" is derived by dividing "Duplicated lines" by the total "Lines of code".

Type: A metric can either be *internal* or *external* [5]. An *internal* metric typically quantifies static properties inherent to the measures entity, while an *external* metric typically quantifies dynamic properties of the entity in a context. "Duplicated lines" is therefore an *internal* metric, while "Number of requests" would be an *external* metric.

Range: The range of a metric specifies which values the metric can exhibit, for "Duplicated lines" this would be the range $[0, \infty]$, while for "Duplicated lines percentage" it would be $[0, 100]$. This information helps to detect incorrect measurements.

Expected value: This field indicates one (or a range of) values which are considered to be "normal" for a metric, for example based on a benchmark of systems. In addition, it is unlikely that there are no "Duplicated lines" in a system, thus the expected value would not be 0. This information provides the context for the implications described in the **Implications**-field. In other words, it helps to put the terms "high" and "low" used in that field in perspective.



Name:	*full name*	*(abbreviation)*	**Level:**	*Base/Derived*
Entity:	*what is the subject of the measurement?*		**Type:**	*Internal/External*
Attribute:	*what characteristic does it quantify?*		**Range:**	*…*
Definition:	*what is the measurement procedure?*		**Expected value:**	*…*
Rationale (theoretical):			**Variability:**	*…*
why does this metric quantify the defined attribute of the entity?			**Scale type:**	*…*
			Related metrics:	
			<metric 1>	*<how?*>*
Implications (practical):			*<metric 2>*	*<how?*>*
Definition of undesired metric values and explanation of the implications of these undesired metrics values, e.g. why does the metric matter?				
			Validation:	
Applicable in context:			*<name>*	*<type*>*
When is this metric useful?			*<name>*	*<type*>*
Solution Strategies		**Solution Type**		
<solution1>		*Treating/solving*		
<solution2>		*Treating/solving*		

Figure 1: The proposed Software Metric Catalog Format outlined in cells with black text. Cells with cursive, grey text indicate which information must be filled in.

Variability: The variability of a metric indicates the expected variance in metric values over a certain period of time. This helps to understand which changes in the value of the metric can be considered as outliers and require more attention. For example, the variability for the metric "Duplicated lines percentage" metric is expected to be low, maybe in the range of $[-1, 1]$. Note that this field is less useful for absolute metrics, since the variability for those metrics can theoretically be endless (e.g., $[-\infty, \infty]$ for "Duplicated lines").

Scale type: There are five different scale types which, amongst others, indicates which mathematical operations may be performed on a metric [4]. For example, "Duplicated lines" is on an absolute scale, which means that all operations are allowed. However, a metric such as "Project type" is nominal, which means that only equivalence operations are allowed.

Related Metrics: A metric is related to other metrics in various ways. Metrics can, for example, measure the same attribute, be orthogonal, be strongly correlated, or can influence each other. For example, "Duplicated lines" is influenced by "Lines of Code", since more "Lines of code" will typically lead to more "Duplicated lines". Moreover, "Duplicated lines" is related to a metric such as "Duplicated functionality" in the sense that all "Duplicated lines" normally imply "Duplicated functionality". By specifying these types of relationships it becomes easier to choose a metric with better (or worse) precision, or to find complementary metrics in order to make it more likely to reach a goal.

Validation: This field is dedicated to listing studies which confirm (or reject) the information contained in the other fields. These studies are listed in the same reference format as used within the research literature. Apart from this information, the field contains a *type* sub-field labeling each study with the type(s) of validation the study performs. The 47 different types of validation as introduced by Meneely et al. [10] is used as a starting point for this labeling. As an example, the study of McConnel [9] can be labelled with "external validity" as it associated "Duplicated lines" with several desirable properties. Note that this study should also be labelled with "experience based" since it does not contain (or refers to) experiments to validate its hypotheses.

4. DISCUSSION

The proposed format is designed to provide a concise overview of the most important information about a metric. Based on this information it should be possible to select a (set of) metric(s) to measure an attribute of interest.

With this information we can answer seven out of the 10 questions of Kaner et al. [6]. The ones that cannot be answered are questions related to the scale and the variability of the *attribute* to be measured (questions four and five), or the measurement error of the *measurement instrument* (question number eight).

Not being able to answer these questions using the proposed format is not seen as problematic. We consider the selection process of metrics to consist of three steps: identify the attribute to be measured, select the appropriate metric(s), and then choose the measurement tools.

The proposed format is targeted towards the second step, identifying the metric(s), thus we assume that the information about the attributes have already gathered. Moreover, we see the selection of the measurement instrument to be the next step in the process and thus out of scope of the format. Based on this reasoning the SMCF provides enough information for its purpose.

Although the SMCF only contains 16 fields our first experiences with the format indicate that it is not trivial to fill out all of them. For example, for most metrics the **Expected value** and the **Variability** data is not readily available. Moreover, creating a complete overview of all **Related metrics** and **Validation studies** is not a small task, and can potentially result in large amounts of information.

To counter both problems we plan to build upon the proposal of Olsina et al. [11] and implement an on-line wiki in which the semantic links between metrics and studies can be stored in a structured manner. Using an on-line environment would allow for the incremental collection of information. In addition, this approach allows us to dynamically hide or show information based on the information need of the user. For example, we can show only a condensed explanation of the definition on the front-page to allow users to quickly scan the metrics, while still providing a link to all the details of the metric for those users that need to implement the metric in a tool.

5. CONCLUSION AND NEXT STEPS

This paper introduces the Software Metrics Catalog Format (SMCF) consisting of 16 fields divided into two columns, one column for qualitative information and one column for quantitative information. The purpose of the SMCF is to provide practitioners and researchers with a concise overview of information about a metric. Based on this information it should be clear whether the metric is appropriate to be used in a given context.

In order to validate whether SMCF can be used for this goal we plan to perform a validation study inside the Software Improvement Group[1], a consultancy firm specialized in conducting risk assessments on software systems using software metrics [3].

To conduct this study, we will create a first version of the catalog with those software metrics which are often used during the risk assessments, consolidating the information available for those metrics. After this version is made available we plan to validate the usefulness and usability of the catalog format both quantitatively as well as qualitatively. For the first part we monitor the online usage of the catalog, while for the second part we conduct in-depth interviews with users.

After this first iteration, we plan to make the catalog available to a larger audience. By again monitoring the usage as well as conducting interviews with users we plan to identify which information fields are unnecessary and which information is missing.

As a side-product, this catalog can provide a quantitative basis to describe the current body of knowledge about software metrics. This way, the catalog does not only provide researchers with the means to easily navigate related work, but it also allows for the identification of those research areas that are in need of more attention.

[1]`http://www.sig.eu`

6. REFERENCES

[1] E. Bouwers. The next step in software metrics research. *TinyToCS*, 2, 2013.

[2] E. Bouwers, J. Visser, and A. van Deursen. Getting what you measure. *Communications of the ACM*, 55(7):54–59, July 2012.

[3] A. v. Deursen and T. Kuipers. Source-based software risk assessment. In *ICSM '03: Proceedings of the International Conference on Software Maintenance*. IEEE Computer Society, 2003.

[4] N. Fenton and S. Pfleeger. *Software Metrics: A Rigorous and Practical Approach*. PWS Publishing Co., 2nd edition, 1998.

[5] International Organization for Standardization. ISO/IEC 25010: Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models, 2011.

[6] C. Kaner and W. Bond. Software engineering metrics: What do they measure and how do we know? In *10th International Software Metrics Symposium - Metrics 2004*. IEEE Computer Society Press, 2004.

[7] B. Kitchenham. What's up with software metrics? A preliminary mapping study. *Journal of Systems and Software*, 83(1):37 – 51, 2010.

[8] H. Koziolek. Sustainability evaluation of software architectures: a systematic review. In *Proceedings of the joint ACM SIGSOFT conference – QoSA and ACM SIGSOFT symposium – ISARCS on Quality of software architectures – QoSA and architecting critical systems – ISARCS*, QoSA-ISARCS '11, pages 3–12. ACM, 2011.

[9] S. McConnell. Why you should use routines...routinely. *Software, IEEE*, 15(4):96, 94–95, 1998.

[10] A. Meneely, B. Smith, and L. Williams. Validating software metrics: A spectrum of philosophies. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 21, 2012.

[11] L. Olsina, G. Lafuente, and O. Pastor. Towards a reusable repository for web metrics. *Journal of Web Engineering*, 1(1):61–73, Oct. 2002.

[12] J. Saraiva, E. Barreiros, A. Almeida, F. Lima, A. Alencar, G. Lima, S. Soares, and F. Castor. Aspect-oriented software maintenance metrics: A systematic mapping study. In *Evaluation Assessment in Software Engineering (EASE 2012), 16th International Conference on*, pages 253–262, 2012.

[13] J. Saraiva, S. Soares, and F. Castor. Towards a catalog of object-oriented software maintainability metrics. In *Emerging Trends in Software Metrics (WETSoM), 2013 4th International Workshop on*, pages 84–87, 2013.

TUD-SERG-2014-004
ISSN 1872-5392