Towards Systematic Reuse of Metric Specifications

[1]Matthias Vianden, [1] Horst Lichter, [3] Karl-Joachim Neumann
[1]*Research Group Software Construction – RWTH Aachen University,*
{vianden, lichter}@swc.rwth-aachen.de
[3]*Generali Deutschland Informatik Services GmbH,*
karl-joachim.neumann@generali.de

Abstract

This paper presents a novel approach for reusing metrics. Contrasting a lot of work related to this issue, we are focusing on reusing metric specifications. By introducing reusability concepts such as genericity and variation points to metric specification we enable the creation of reusable ones. On the other hand this allows deriving project specific concrete metrics based on reusable metric specifications. Early experience from our industry cooperation is promising and indicates less stress and effort for the metric users.

Keywords: *Metric Specification, Variability, Genericity, Reuse*

1. Introduction

Software metrics are an important means to measure the quality of both the development processes and software systems. Additionally, they are often used to show the efficiency of algorithms [1], [2]. Improvement reference models such as CMMI require that software development organizations build up abilities to systematically apply metrics to support project management. Based on quantifiable metrics project managers are able to review those processes that contributed to project success or failure. Hence, metrics are a necessity for objective process optimization. However, research shows that it is demanding to find the right metrics; 58% of all project managers and 50% of all senior managers find it difficult to collect, analyze, and use the right metrics [3].

On the one hand, metric frameworks like GQM help to derive metrics from abstract goals for the project [4]. On the other hand, defining metrics just for one project (in a multi project organization with a lot of similar projects) is costly and ineffective. Hence, it is wise to reuse metric experience (metric definitions, evaluations, and models) [5] as all experience can and should be reused [6]. Well-planned metric frameworks and reuse of existing metrics material is also mentioned as one key success factor for successful metric programs [7].

Different aspects of metrics can be reused. Measurement tool reuse occurs very often because many organizations use the same LOC counters or even complete measurement tool suits like sonar [8]. Measurement processes like GQM [4] are reused as well. The reuse of measurement values also increased during the last ten years. These (baseline) values are often used to enhance estimations. A popular example for this are the database and tools from the ISBSG [9]. Even though various measurement aspects are being reused metric specifications are very rarely reused. This ignores the fact that a database of metric specifications can spread metric knowledge across the organization and different projects (only 29% of the project managers and 24% of other practitioners know how measurement data was used in other measurement projects [10]). Furthermore, the tailoring of modern metric based project management cockpits to fit the need of specific project roles [11] is a form of (implicit) reuse of metric specifications

Although considerable research has been devoted to the modeling of metrics and metric frameworks, rather less attention has been paid to investigating how the results of this research (metric meta models, metric frameworks, and metric experience bases) can lead to a sound reuse concept for metrics and their specifications.

After thoroughly reviewing related work and metric specifications in the first two sections of this article we define two main goals for systematic reuse of metric specifications. This is followed by addressing the crucial aspect of variability in reusable metric specifications. We finish this paper with some early experience of using a tool to support systematic metric specification reuse in an industry environment.

2. Related Work

Metric reuse is often implied or indicated but very rarely it is addressed. For example, the first three steps in the CAME Framework (Choice, Adjustment, Migration, Efficiency) by Dumke et al. indicate the benefits of an explicit modeling of metric variability [12]. However, the formal description of measurement and evaluation by Dumke and Schmietendorf [13] only mentions the importance of maintaining a metric experience base. Later, Dumke et al. again imply the reuse of metrics because different usages and applications are modeled for measurement methods [14]. However, the concept of metric reuse is not covered in more detail. Hihn and Lewicki indicate a common set of standard metrics which is (re)used over several projects [15]. But no explicit tailoring of these metrics is mentioned nor is management or specification of variability. Starons and Medings pre configured wizards for the definition of metrics [16] also implies metric reuse by automatically adding pre configured base measures. Again, reuse is not addressed by a sound concept but rather used pragmatically.

Most of nowadays model driven measurement approaches also imply metric reuse. For example as proposed by Clavel et al. [17] and extended by McQuillan and Power [18] by defining metrics based on UML concepts and OCL. Yet, this only allows the (pragmatic) reuse of complete metric definitions; only reusing fragments of metrics or the modeling of metric variability is not addressed. Reuse of existing metric components (like line charts, project plan structure and MS Project Import) is mentioned by Heidrich and Münch [19]. But neither the reusable components nor their variability is explicitly modeled. Garcia et al. have proposed a model based environment for the integrated management of software measures [20]. They provide “generic metrics defined within the meta model scope” which according to the case study by Mora et al. on this environment homogenized the measurement process [21]. However, the variability of the metrics is again not reflected in the models (and in the meta model).

Reusable (sets of related) metrics are often represented by metric frameworks. According to Mendonsa and Basili these frameworks may also contain data collection mechanisms and information about data usage [22]. The framework implied by MIS-PyME – Software Measurement Maturity Model [5], [23], [24] suggests the reuse of existing measurement models of the organization, because “defining measurement programs for certain projects or products, … will be costly, difficult to handle and of little worth for future developments” [5]. Similarly, one of the goals of the INCAMI framework for (Web-based) metric documentation [25], [26] is to “allows an organization to run different projects by making use of common measurement and evaluation mechanisms” [27]. But neither MIS-PyME nor INCAMI provide sound concepts for metric reuse.

Sets of reusable metrics could also be stored in an organizational wide metric experience base of a Learning Organization. As research by Krein et al. shows: providing a knowledge repository helps to push information back to the consumer [28]; in our case: supports reuse. Learning Organizations also avoid local optimization of projects (and metrics) and focus on global optimization of the organization. The work of Althoff et al. indicates that learning organizations and avoidance of local optimizations reward reuse [29]. Palza et al. describe specific metric experience bases which store the definition of and experience with specific metrics [30]. But, they also do not model metric reuse or metric variability.

3. Metric Specifications

A metric specification includes a specification of the measurement (how the value is measured or calculated), an interpretation (guide) for the measurement results, and sometimes even specifications for the visualization of the metric values. The importance of specifying metrics is reflected in the fact that metric specifications are required to reach CMMI level 2 [31].

Metric specifications also act as a documentation of the metric. Hence, the topic of metric specification is addressed in a lot of research papers concerning metric documentation [30], [32]. Most of these approaches are based on metric meta-models or on metric ontologies resulting in more formal specification rather than informal plain text. However, our experience shows that most of the metric specifications used in the industry (if they are used at all) are plain text documents. Sometimes these documents are on a more formal level by containing dedicated sections for specific attributes. For example the twelve steps to useful software metrics by Linda Westfall [33], the required specifications for CMMI [31], or at least “goal”, “question” and “metric” sections [4].

4. Goals

Reusing metric specifications must be easy. Forcing people to specify their metrics is hard enough anyhow. But what makes reusing metric specifications easy? Before we address this, we like to give two examples of typical scenarios for metric specification reuse:

1. **Example A:** A project manager would like to know whether his projects costs are developing like planed. He/She needs to find, that the CPI[1] metric can answer this question. When using this metric the project manager does not want to specify the details of this metric again. He/She rather likes to focus on project specific changes to the metric (e.g. the timing of the measurement: daily, weekly, or monthly).

2. **Example B:** Many metrics are based on counting entities in a specific state (e.g. change requests with states such as "new" and "accepted"). Often the results of these metrics are visualized as staked bar charts. They can indicate problems with the process if the state change of the entities is visible (the bars for the specific states remain at the same heights over time). Hence, this concept could be specified in a reusable metric specification. When reused, a metric user should only need to specify the entity and its states.

These examples and our experience show that reusable metric specifications often need to contain "adaption points". These points are resolved (tailored) by the metric users upon using the reusable specification. Another important aspect of reusable metric specifications is documentation, because metric users can only reuse specifications that they are aware of and that fit their need. Hence, we deduced that "easy reuse" means:

1. **It is easy to find the metric that answers the questions of the metric users.**

   From our experience, metric users often browse lists of reusable metric specifications to find metrics that best fit their need(s). Besides the general description of the metric, metric users need to see an example of the metric visualization. This problem is well known and covered by a lot of approaches and tools.

2. **The specific change (tailoring) of the metric specification is guided.**

   We believe that tailoring assistance is crucial. Our experience and the literature show that most of the reusable metric specifications need to be enriched with specific information [34]. However, metric users in general do not want to be bothered with the details of the metric specification. Therefore, the reusable metric specifications, the reuse process and the tool support need to deal with the variability of the reusable metric specifications. Contrasting the documentation and following our argumentation in the introduction and our analysis of the related work, research does not address tailoring assistance for variability in (reusable) metric specifications.

5. Addressing Variability in Reusable Metric Specifications

Variability is an everlasting problem in software development and addressed in special areas like product line engineering [35], [36]. We will focus on two possible solutions for dealing with variability: *Parameterization* (especially genericity), and *variation points*.

Genericity, a special form of parameterized polymorphism, is a well known concept of programming languages like Java. Following Betrand Meyer, *genericity* "is a technique for defining elements that have more than one interpretation depending on parameters representing types" [37]. In instantiating a concrete element from a generic one, the formal generic parameters need to be replaced by concrete types.

Variation points are a concept from product line engineering. They are used to model the variability of a set of software products. Variation points are used to scope the system i.e. to determine what

[1] The cost performance index (CPI) is one of the key metrics of the earned value framework [39].

should be realized in a product line and what needs to be realized individually. They can specify a set of possible variations[2] or be left open[3].

We suggest applying a combination of genericity and variation points to realize variability in metric specifications. The "adaptation points" of reusable metric specifications are modeled as variation points. Of course, these variation points and the variants need to be clearly marked in the specification and need to be documented to ease and support tailoring. These variation points are the formal parameters of the reusable metric specification. When reused, concrete values for all formal parameters need to be specified to derive a fully specified metric specification.

5.1. Variability Application Example

The following two examples show fragments of reusable metric specifications based on examples A and B from chapter 4. The categories are based on the ISO 15939 standard [38] as well as the requirements of CMMI for development at level 3 [31]. The first one (Example A from above - CPI) has a closed variation point (timing), the second one (Example B from above - Counting Metric) has an open variation point (entity of measurement).

5.1.1. Example: CPI Metric

The timing of the CPI metric should be variable. Hence, the timing is modeled by a variation point"`<VP: CPI-Timing>`". Because the timing should be limited to a fixed set of values, these values are listed afterwards.

```
Metric Name:           CPI (Cost Performance Index)
Entity of Measurement: Project
Answered Question(s):  Are the costs developing like foreseen?
Interpretation:        If the values of this metric are below 1.0 …
Measurement Function:  Earned Value / Actual Costs
Timing:                <VP: CPI-Timing>
                       <Values: {daily, weekly, monthly}> …
```

If a project manager would like to use this metric, he/she simply needs to refer to this reusable specification and select a concrete timing value, e.g.: `MyCPI := CPI(CPI-Timing ← weekly)`.

5.1.2. Example: Counting Metric

The variable aspect of this metric is the measured type of entities and their states used in the measurement function. Contrasting the CPI example, the entity of measurement should not be restricted but left open for any "entity with states". Because the number of states is unknown, the metric result is a vector. This metric could be specified as follows:

```
Metric Name:           CountM (Counting Metric)
Entity of Measurement: <VP: CountM-EoM>
                       <Values: Entity with States>
Measurement Function:  (Vector: ForAll state x count(#e with e.state = x))…
```

This abstract specification can be used to derive a broad variety of metrics specifications. For example, based on the entity type `Change Request` defining states such as `open`, `postponed`, or `accepted` the instantiation `CRProcessMetric := CountM(CountM-EoM ← Change Request)` results a metric to measure change requests data.

[2] e.g. the timing can either be "daily", "weekly", or "monthly"
[3] e.g. the entity of measurement is often restricted to a specific type ("project") but not to a specific set of "projects"

6. Early Experience

This early experience comes from our work at the IT-department of a large insurance company (about 1.300 people). Since the last 3 years we are working on the metrics topic as part of the CMMI level 3 initiative at the company. During this time new metric processes, a new standard metric cockpit for project managers, and a metric documentation tool was developed. All these things are now used at the company and help the project managers to ease working with metrics.

To ease the reuse of metric specifications and to fulfill the goals listed in chapters 4 and 5 we built a web based support tool. This tool was developed in an iterative process together with metric experts from the company. Because it did now reach CMMI level 3 every project is required to specify their metrics according to the company wide standards. Additionally, it needs to be documented where a project is using and tailoring a companywide defined (reusable) metric specification and what metrics are specifically defined for the project. Until now, this is done using Excel specification sheets. The goal of the tool is to supersede the Excel sheets.

Because we received a lot of positive feedback concerning the tool and the idea of metric specification reuse from the metric experts and some selected project managers, we like to share some of our design decisions regarding the tool and the process of reusing metric specifications in this concrete environment.

- **Process setup**. We suggest forming a (metric) expert group inside the organization. This group needs to develop and prepare reusable metric specifications. It is responsible for managing the set of reusable metric specifications. That is: adding new reusable metric specifications to the repository and deprecating old ones which should not be used anymore. The metric experts also need to optimize the set of reusable metric specification. They need to investigate why certain metric specifications are not reused and then change or deprecate them. Additionally, they need to ensure that the tailoring dimensions are sufficient and that the reusable metric specification is as flexible (or inflexible) as anticipated.
- **Metric specifications**. In the tool the metrics are specified as plain text. However, this text follows a schema that is related to the twelve steps proposed in [33] and in the requirements of CMMI. The specifications are organized in five categories (budget, time, risk, content, and quality).
  The reusable metric specifications document the set of “answers” they provide when used. Therefore, the questions are used as additional categories. We experienced that the “goal” from GQM is far less important to keep in the documentation. Additionally, we realized that, besides the general description of the metric, metric users need to see an example of the visualization of the metric.

To continue our work we plan to use the tool in a wider scope with more projects. Our final goal is to store all the metric specifications of the company and the projects in the tool within the next year. Additionally we plan to do a companywide qualitative as well as quantitative evaluation of the benefits of the tool and the underlying metric specification reuse processes and ideas. From our current perspective the feedback is very promising and we hope to leverage the specification of metrics with the help of the tool. Hopefully this helps the project managers (as well as other stakeholders) to cope with the difficult task of specifying metrics.

7. Conclusion

In this paper we argued that it is wise to reuse metric specifications. We identified the crucial aspect of variability for reusable metric specifications and showed two examples of reusable metric specifications. Early experience in an industry environment, during the development of a tool support for tailoring of reusable metric specifications, is promising. However, the topic of reusable metric specifications needs more investigation. Right know we are focusing on evaluating the tool support. Another interesting aspect that we did not address in this paper is the question how the reusable metrics and their variation points are identified.

8. References

[1] F. Wei, J. Xu, X. Liu, T. Wen, and H. Yan, “User-Oriented QoE Metrics and Subjective Assessment for Video Streaming Services on 3G Mobile Internet,” *JDCTA: International Journal of Digital Content Technology and its Applications*, vol. 6, no. 9, pp. 17–25, 2012.

[2] X. YANG and X. YANG, “An Efficient Coding Algorithm for Photogrammetric Image,” *JDCTA: International Journal of Digital Content Technology and its Applications*, vol. 6, no. 9, pp. 76–84, 2012.

[3] T. Hall, N. Baddoo, and D. Wilson, “Measurement in software process improvement programmes: An empirical study,” *New Approaches in Software Measurement*, pp. 73–82, 2001.

[4] P. Berander and P. Jönsson, “A goal question metric based approach for efficient measurement framework definition,” in *Proceedings of the 2006 ACM/IEEE international symposium on International symposium on empirical software engineering - ISESE ’06*, 2006, p. 316.

[5] M. Diaz-Ley, F. Garcia, and M. Piattini, “Implementing Software Measurement Programs in Non Mature Small Settings,” *Software Process and Product Measurement*, pp. 154–167, 2008.

[6] V. R. Basili and H. D. Rombach, “Support for comprehensive reuse,” *Software Engineering Journal*, vol. 6, no. 5, pp. 303–316, Jul. 1991.

[7] T. Hall and N. Fenton, “Implementing effective software metrics programs,” *IEEE Software*, vol. 14, no. 2, pp. 55–65, 1997.

[8] “Sonar Web Page.” [Online]. Available: http://www.sonarsource.org/.

[9] C. Lokan, T. Wright, P. Hill, and M. Stringer, “Organizational benchmarking using the ISBSG Data Repository,” *IEEE Software*, vol. 18, no. 5, pp. 26–32, 2001.

[10] S. T. Parkinson, S. Counsell, M. Norman, R. M. Hierons, and M. Lycett, “The precursor to an industrial software metrics program,” in *ITI 2008 - 30th International Conference on Information Technology Interfaces*, 2008, pp. 221–226.

[11] J. Heidrich, J. Münch, and A. Wickenkamp, “Usage Scenarios for Measurement-based Project Control,” in *Proceedings of the 3rd Software Measurement European Forum (SMEF 2006),(Ton Dekkers, Ed.)*, 2006, pp. 47–60.

[12] R. Dumke and R. Koeppe, “Conception of a Web-Based SPE Development Infrastructure,” in *Performance Engineering, State of the Art and Current Trends*, 2001, pp. 1–19.

[13] R. Dumke and A. Schmietendorf, “Formal Description of Software Measurement and Evaluation - A Short Overview and Evaluation,” 2006.

[14] R. Dumke, H. Yazbek, E. Asfoura, and K. Georgieva, “A General Model for Measurement Improvement,” in *Lecture notes in computer science, IWSM/Mensura 2009*, 2009, no. 2, pp. 48–61.

[15] J. Hihn and S. Lewicki, “Bootstrapping Process Improvement Metrics: CMMI Level 4 Process Improvement Metrics in a Level 3 World,” in *Proceedings of the 44th Hawaii International Conference on System Sciences*, 2011, pp. 1–10.

[16] M. Staron and W. Meding, “Using Models to Develop Measurement Systems: A Method and Its Industrial Use,” *Software Process and Product Measurement*, pp. 212–226, 2009.

[17] M. Clavel, M. Egea, and V. Torres, “Model Metrication in MOVA: A Metamodel-Based Approach using OCL,” Citeseer, 2007.

[18] J. A. McQuillan and J. F. Power, “Towards re-usability of software metric definitions at the meta level,” in *PhD Workshop of the 20th European Conference on Object-Oriented Programming (ECOOP 2006)*, 2006.

[19] J. Heidrich and J. Münch, “Goal-oriented setup and usage of custom-tailored software cockpits,” in *PROFES ’08: Proceedings of the 9th international conference on Product-Focused Software Process Improvement*, 2008, pp. 4–18.

[20] F. Garcia, M. Serrano, J. Cruzlemus, F. Ruiz, and M. Piattini, “Managing software process measurement: A metamodel-based approach,” *Information Sciences*, vol. 177, no. 12, pp. 2570–2586, 2007.

[21] B. Mora, F. Garcia, F. Ruiz, and M. Piattini, “Model-Driven Software Measurement Framework: A Case Study,” in *2009 Ninth International Conference on Quality Software*, 2009, pp. 239–248.

[22] M. G. Mendonsa and V. R. Basili, “Validation of an approach for improving existing measurement frameworks,” *IEEE Transactions on Software Engineering*, vol. 26, no. 6, pp. 484–499, Jun. 2000.

[23] M. Díaz-Ley, F. García, and M. Piattini, “MIS-PyME software measurement capability maturity model – Supporting the definition of software measurement programs and capability determination,” *Advances in Engineering Software*, vol. 41, no. 10–11, pp. 1223–1237, Oct. 2010.

[24] M. Diaz-Ley, F. García, and M. Piattini, “MIS-PyME Software Measurement Maturity Model-Supporting the Definition of Software Measurement Programs,” in *PROFES ’08: Proceedings of the 9th international conference on Product-Focused Software Process Improvement*, 2008, pp. 19–33.

[25] L. Olsina and M. D. L. A. Martín, “Ontology for Software Metrics and Indicators: Building Process and Decisions Taken,” in *Web Engineering*, 2004, p. 778.

[26] M. de Los Angeles Martin and L. Olsina, “Towards an Ontology for Software Metrics and Indicators as the Foundation for a Cataloging Web System,” in *Web Congress*, 2003, pp. 103–113.

[27] H. Molina and L. Olsina, “Towards the Support of Contextual Information to a Measurement and Evaluation Framework,” in *6th International Conference on the Quality of Information and Communications Technology (QUATIC 2007)*, 2007, pp. 154–166.

[28] J. L. Krein, P. Wagstrom, S. M. Sutton Jr, C. Williams, and C. D. Knutson, “The problem of private information in large software organizations,” in *Proceeding of the 2nd workshop on Software engineering for sensor network applications*, 2011, pp. 218–222.

[29] K.-D. Althoff, F. Bomarius, and C. Tautz, “Knowledge Management for Building Learning Software Organizations,” *Information Systems Frontiers*, vol. 2, no. 3–4, pp. 349–367, 2000.

[30] E. Palza, A. Abran, C. Fuhrman, and E. Miranda, “V & V Measurement Management Tool for Safety-Critical Software,” in *Proceedings of IWSM/MetriKon 2004*, 2004.

[31] C. P. Team, “CMMI® for Development, Version 1.3 CMMI-DEV, V1.3,” 2010.

[32] M. de Los Angeles Martin and L. Olsina, “Towards an Ontology for Software Metrics and Indicators as the Foundation for a Cataloging Web System,” in *Web Congress*, 2003, pp. 103–113.

[33] L. Westfall, “12 Steps to Useful Software Metrics.” The Westfall Team, 2005.

[34] C. Ebert, R. Dumke, M. Bundschuh, and A. Schmietendorf, *Best Practices in Software Measurement*. 2004, p. 295.

[35] C. Krueger, “Variation management for software production lines,” *Software Product Lines*, pp. 107–108, 2002.

[36] D. Rombach, “Integrated software process and product lines,” in *International Software Process Workshop (SPW) 2005, Beijing*, 2005, pp. 83–90.

[37] S. Microsystems, “Java Programming Language (1.5),” 2004, 2012. [Online]. Available: http://docs.oracle.com/javase/1.5.0/docs/guide/language/index.html.

[38] *ISO/IEC 15939: 2002 “Software Engineering - Software Measurement Process”*. 2002.

[39] F. T. Anbari, “Earned value project management method and extensions,” *IEEE Engineering Management Review*, vol. 32, no. 3, pp. 97–97, 2004.