Available online at www.sciencedirect.com
SCIENCE DIRECT®
The Journal of Systems and Software 74 (2005) 207–226
The Journal of Systems and Software
www.elsevier.com/locate/jss

Characterizing a data model for software measurement ☆

L. Chirinos [a], F. Losavio [a,*], J. Bøegh [b,1]

[a] Laboratorio de Tecnología del Software (LaTecS), Centro de Ingeniería de Software y Sistemas (ISYS), Facultad de Ciencias, Universidad Central de Venezuela, Apdo. 47567, Los Chaguaramos 1041-A, Caracas, Venezuela
[b] DELTA Danish Electronics, Light & Acoustics, Venlighedsvej 4, DK-2970 Hørsholm, Denmark

Received 6 June 2003; received in revised form 22 January 2004; accepted 24 January 2004
Available online 6 March 2004

Abstract

In order to develop or acquire a software product with appropriate quality, it is widely accepted that quality must be identified, planned, measured and controlled during the development process using quality measures based on a quality model. However, few practitioners in the software industry would call measurement a success story. This weakness arises, on one hand because the people involved are not always aware of the importance of collecting measures. The policy of the management board must make sure that people are sufficiently motivated and that data is actually collected in the specified way. On the other hand, software measures have been often poorly defined in industry. When software measurement definitions are incomplete and/or poorly documented, it is easy to collect invalid or incomparable measures from different data collectors. Thus, the primary issue is not only whether a definition for a measure is theoretically correct, but that everyone understands what the measured values represent. Then, the values can be collected consistently and other people, different from the collectors, can interpret the results correctly and apply them to reach valid conclusions. The objective of this paper is to present a data MOdel for Software MEasurement (MOSME) to explicitly define software measures, providing the elements required to describe a consistent measurement process. MOSME can be used for defining and modeling data sets of software products involving several software projects. The inspiration of this work comes from the SQUID (Software QUality In the Development process) approach, which combines many results from previous research on software quality and the European Commission funded projects SQUAD and CLARiFi. The application of MOSME is illustrated with a case study. We believe that a conceptual model of fully defined meaningful measures will help both the management board to give support to the data collection policy and the practitioner to avoid ambiguity in the definitions of the data measures.

Keywords: Software measurement; Software quality; Measure; Metric; Model for software measurement

1. Introduction

The behavior of real world objects can be characterized formally and quantitatively by measurement only if the measurements are *objective*, *empirical* and *reproducible*. Even if semantically the term "objective" means not involving human judgment, objective measurement means in this context that a written and agreed procedure or measurement method for assigning a number or category to an attribute (measurable property) of an object is established, preserving our intuition about the behavior of the object. Empirical measurement relates to the way of obtaining data (from observation or from a psychometrically valid questionnaire). Reproducible measurement means that the same result is obtained when different persons repeat the measurement. In order to fulfill these requirements it is important to identify and define all the elements involved in measurement, as well as the relationships existing among them (Briand et al., 2002). Many software measurement schemes fail due to poor definitions (Kitchenham et al., 2001).

Measurement paradigms such as GQM (Solingen and Berghout, 1999) and Ami (Ami Handbook, 1992)

☆ This work is a result of the CEE INCO SQUAD project EP 962019 and the CDCH ARCAS project 03.13.4584.00 of the Universidad Central de Venezuela, Caracas, Venezuela.
* Corresponding author. Tel.: +58-212-60516/212-7536984; fax: +58-212-7536984.
E-mail addresses: lchirinos@cantv.net (L. Chirinos), flosav@cantv.net (F. Losavio), jb@delta.dk (J. Bøegh).
[1] Tel.: +45-721-94397.

0164-1212/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jss.2004.01.019

provide methods for identifying the measures required in a specific situation. Likewise, for example, the metrics proposed by Chidamber and Kemerer (1994) and Paul (1993) identify specific sets of measures appropriate for particular contexts. However, these approaches do not define how such measures should be collected and stored. The quality of any measurement program is clearly dependent on careful data collection (Fenton and Pfleeger, 1997). Data collection is becoming a discipline in itself, where specialists work to ensure that measures are defined unambiguously, that data collection is consistent and complete, and that data integrity is not at risk.

In order to ensure repeatability, Kitchenham et al. (1995) suggested the inclusion of the following elements for defining the measure: the *entity* (i.e. software object), the *attribute* (i.e. property), and the *unit of measurement*. They also showed that the entity–attribute–unit is insufficient for ensuring *comparability*. The *conditions* under which a measurement is taken must also be specified. This is called the *measurement protocol* by Kitchenham or the *counting rule* by Bøegh et al. (1999). A counting rule is closely related to a *data collection process*, which is also considered by GQM. It identifies the responsible for the data extraction, the point in the development process when the measure is taken and the methods and tools used to extract, record, and store the data values.

Based on these issues, Kitchenham et al. (2001) proposed a method for specifying models of software data sets in order to capture the definitions and possible relationships among software measures. It aims to provide not only a standard structure for defining software measures, but also a means of modeling complex data sets. It is primarily based on the lessons learned from a toolset developed by the ESPRIT project SQUID (Bøegh et al., 1999) to support software quality measurement and prediction. In Bøegh et al. (1999) and Kitchenham et al. (2001) the view of measurement modeling suggested by Kitchenham et al. (1995) is extended. Although the counting rule is essential for ensuring repeatable and comparable values, important issues such as the characterization of the entities with respect to the measurable attribute, have not been explicitly considered. Moreover, the circumstances for applying the counting rule have only been considered by GQM.

Recently, the International Standard Organization (ISO) has formulated a standard for software measurement (ISO/IEC 15939, 2002). It focuses on two key concepts: *a measurement process model* and a non-normative description of a *measurement information model* which is merely given as a recommendation. The measurement process model defines the activities of the measurement process for the specification of the measurement information required, the application of the measures, the analysis and evaluation of the results. The measurement process model is an adaptation of the Plan-Do-Check-Act cycle. The activities are sequenced in an iterative cycle allowing for continuous feedback and improvement of the measurement process, which is driven by the information needs of the organization. An *information need* (*goal* for GQM) is an insight necessary to manage goals, risks and problems. For each information need, the measurement process produces an information product, which is represented by one or more indicators and their associated interpretations that address information needs. On the other hand, the measurement information model supports the measurement process. It captures the measurement terminology and describes the links between measures and information needs, expressed by entities and measurable attributes of concern. This link is defined through the *measurement constructs*. Each construct may involve several types or levels of measures (base and derived measures and indicators). This issue is not considered by the other frameworks and it is important for industry because it enables the data collection (using base measures) to be uncoupled form the analysis (using indicators). Moreover, it associates pre-defined decision criteria with indicators, which is critical to industry. It provides a structure very similar to the GQM refinement (Solingen and Berghout, 1999). However, the relations defined between the elements involved in the measure definition are not normative.

This paper proposes a MOdel for Software MEasurement (MOSME), on the basis of the previous research, constituting a conceptual data model that explicitly identifies and accurately defines the main elements and relationships involved in the software measure definition. Our model contributes on one hand, to improve the software measurement process by providing the unambiguous capture of software measure definitions, in order to obtain consistently the measurement values during software development. On the other hand, it reduces some of the weaknesses identified in the common practice related to the analysis of the measured values (Bøegh et al., 1999; Kitchenham et al., 2001). In particular, the model provides a definition of the counting rule which takes into account the main aspects concerning this notion. It is based on the "context of use" issue, to enlighten the influence of the people involved in the measurement, as it is suggested by GQM, Moreover, MOSME gives explicit definitions of the main elements and relationships involved in the construction of a measure. This research is based on the works already mentioned (Kitchenham et al., 1995, 2001; Solingen and Berghout, 1999; Bøegh et al., 1999), focusing the basic measurement terminology of the ISO 15939 standard (ISO/IEC 15939, 2002) and being sufficiently general to be applied to define measures for new or for existing software systems. Finally, the concepts