MAMBA: Model-Based Software Analysis Utilizing OMG's SMM

Sören Frey, André van Hoorn, Reiner Jung, Benjamin Kiel, and Wilhelm Hasselbring

Software Engineering Group, University of Kiel, 24098 Kiel

Abstract

Most software system properties can be quantified through applying measurement processes. OMG's Structured Metrics Meta-Model (SMM) supports the meta-model agnostic definition of those measurement processes with an emphasis on architecture-driven modernization scenarios. We present the MAMBA framework that addresses major obstacles software engineers currently face when using SMM in practice. Among those are (1) the lack of appropriate tool support, (2) the cumbersome integration of precomputed measurement data, and (3) the complexity of specifying SMM models and queries.

1 Introduction

To statically or dynamically analyze a software system in a re-engineering context, selected quality attributes are often quantified following a measurement approach [3]. For example, the monitoring of response times during runtime or the determination of the module coupling metric as a decision support for restructuring a software architecture constitute measurement processes. To enable a meta-model agnostic definition of measurement processes, the Architecture-Driven Modernization (ADM) Task Force,[1] a sub-commitee of the Object Management Group (OMG), developed the Structured Metrics Meta-Model (SMM) [2]. In our previous work [1], we introduced MAMBA, a measurement architecture for model-based analysis that builds on SMM. Instances of SMM can be computed with our MAMBA framework or used for querying system models with the MAMBA Query Language (MQL).

In this paper, we (1) detail the basic metrics computation process of our tool,[2] (2) briefly describe how externally computed measurement data can be integrated, and (3) introduce the Metrics Definition Language (MDL) for simplifying the specification of the typically complex SMM models.

The remainder of the paper is structured as follows. Section 2 outlines the SMM concepts relevant to this paper. The MAMBA framework is presented in Section 3, before Section 4 draws the conclusions and describes future work.

This work is partly supported by the German Federal Ministry of Education and Research (BMBF) under grant number 01IS10051, the Program for the Future Economy of Schleswig-Holstein, and the European Regional Development Fund (ERDF).

[1] `http://adm.omg.org/`

[2] `http://mamba-framework.sf.net/`

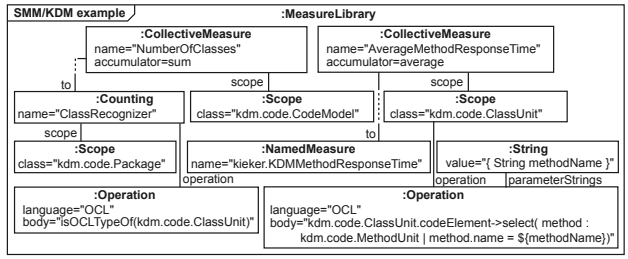


Figure 1: SMM example for KDM domain models

2 Structured Metrics Meta-Model

Central notions in the SMM standard are *measures* and *measurements*. A measure describes an algorithm for calculating specific properties of software system elements. A typical static measure is the number of code lines (LOC) that are contained in each class; an example dynamic measure is the average response time of software methods. In order to use an SMM measure, it must be contained in a *measure library*. SMM includes basic measures, such as *binary* (e.g., $\leq$) and *collective* (e.g., *sum*) measures. User-defined measures can be defined utilizing *direct* or *named* measures. Direct measures can be specified incorporating a query language (e.g., OCL); for *named measures* no executable semantics exist. A measurement is the counterpart of a measure, being produced through executing the latter. A model element that was measured is referenced via a *measurand* relationship of the respective measurement. Measurements observed in a concrete measurement process are grouped in *observations*, associated with additional information, such as the time of the measurement. An example SMM measure library is shown in Fig. 1.

3 MAMBA Framework

Fig. 2 provides an architectural view of the MAMBA framework in terms of core components and usage. The general idea is that users provide a set of domain models, e.g., instances of the Knowledge Discovery Meta-Model (KDM), along with a definition of requested measures in form of SMM models or textual representations which MAMBA translates to SMM. The framework executes the input SMM models and outputs these SMM models enriched by measurements (observations) for the requested measures. Note that the SMM models may contain additional MAMBA-provided extensions, such as user-defined (periodic) aggregate functions [1]. Measurement Providers are used to integrate external static or dynamic analy-

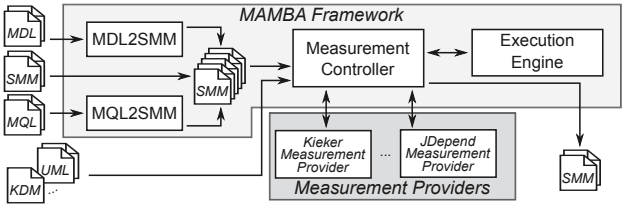


Figure 2: Framework with measurement providers

sis tools, e.g., by executing these and importing the resulting raw measurement data. Fig. 2 indicates the integration of the dynamic and, respectively, static analysis tools Kieker[3] and JDepend.[4]

3.1 Execution of SMM Models

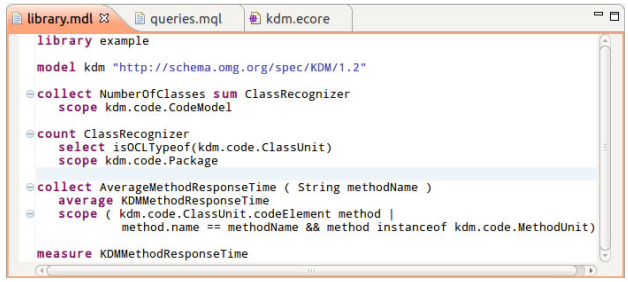
The Measurement Controller creates an instance of the Execution Engine and passes the SMM models, including the list of requested measures, to the latter. As a first step, the Execution Engine inspects the SMM models in order to determine whether at least one named measure is required for the computation of the requested measures. We distinguish between two different modes of execution: if no dependency to a named measure exists (*closed mode*), the Execution Engine can directly compute the SMM measurements simply based on the domain model(s); otherwise (*open mode*), the Execution Engine provides the list of required named measures to the Measurement Controller, which initializes appropriate Measurement Providers for these named measures. The Measurement Providers create observations (i.e., measurements for named measures, including additional meta-information) that are passed to the Execution Engine via the Measurement Controller. After the termination of each Measurement Provider, the Execution Engine executes the SMM models just like the way it executes in closed mode. For reasons of brevity, we do not detail the execution of periodic measures in this paper.

3.2 Metrics Definition Language (MDL)

MDL provides an easy-to-understand and compact textual representation of measure definitions. Fig. 3a shows the MDL representation for the SMM measure library from Fig. 1 in our MAMBA IDE.

Each MDL model comprises a library name, a list of domain meta-models (e.g., KDM), and a collection of measure definitions. These measure definitions follow the same pattern: they start with a keyword, identifying the measure type, followed by a unique name, an optional list of parameter declarations, specific measure properties, and a scope definition. The scope can be any class from an imported domain meta-model or a more complex selector expression, as illustrated with the AverageMethodResponseTime measure in the example.

[3] `http://kieker-monitoring.net/`
[4] `http://clarkware.com/software/JDepend.html`



(a) Measure library defined with MDL

(b) MQL query example

Figure 3: MDL and MQL examples

3.3 MAMBA Query Language (MQL)

MQL applies SMM measures to concrete domain models. Therefore, it references one or more measure libraries and one or more domain models. The application of measures is specified with a set of queries which can provide further analysis. MQL2SMM (see Fig. 2) extends the referenced measure library by transparently adding auxiliary measures. The example in Fig. 3b utilizes the AverageMethodResponseTime measure and returns those average response times which exceed 500 milliseconds.

4 Conclusions and Future Work

OMG's SMM [2] enables the meta-model agnostic definition of measurement processes. However, specifying SMM models is cumbersome and error-prone. Our presented MAMBA framework provides tool support for a simplified definition of those SMM models and enables their automatic execution. Currently, the core components of MAMBA are implemented and have been partly evaluated in industrial contexts [1]. The MDL and MQL components are in a prototype stage and will be refined in our future work.

References

[1] S. Frey, A. van Hoorn, R. Jung, W. Hasselbring, and B. Kiel. MAMBA: A measurement architecture for model-based analysis. Technical Report TR-1112, Department of Computer Science, University of Kiel, Germany, Dec. 2011.

[2] Object Management Group. Architecture-Driven Modernization (ADM): Structured Metrics Meta-Model (SMM) Version 1.0. `http://www.omg.org/spec/SMM/1.0/`, 2012.

[3] L. Tahvildari, K. Kontogiannis, and J. Mylopoulos. Quality-driven software re-engineering. *Journal of Systems and Software*, 66(3):225 – 239, 2003.