The SOMIX 2 Metamodel

Rainer Wolfgang Winkler[a,*]

[a] *CubeServ GmbH, Am Prime-Parc 4, 65479 Raunheim, Germany*

Abstract

The SOMIX 2 Metamodel is specified.

Keywords:
Software exploration, Software maintenance, Software visualization, SAP development

1. Introduction

The SOMIX Metamodel was specified in a similar way as FAMIX [1]. SOMIX 2 is a more general metamodel which resembles the BWW model [2, 3] for software systems. This metamodel is a concrete specification of a type of metamodel described in [4]. Such a metamodel can be read and displayed by the JavaScript version [5] of Moose2Model [6]. The extraction tool SAP2Moose [7] provides currently models in the SOMIX format.

2. Software Metamodel

2.1. Overview of SOMIX 2

A model which is described by the SOMIX 2 metamodel contains components and couplings between these components. In a minimal version no specification of a component is required. It is also not required to specify the character of a coupling. To aid in the understanding of a model it is proposed to specify whether a component is code or data. That a component is used as a type is not specified explicitly. It is uses as a type when it is referenced in a coupling in the field typedBy. That a component is used as grouping

*Corresponding author
Email address: rainer.winkler@cubeserv.com (Rainer Wolfgang Winkler)

is not specified explicitly. It acts as a grouping when it is referenced in a coupling in the field parent. It is also proposed to specify whether a coupling is a parent-child relation, a call, an access or a typing.

Further specifications are possible by adding custom fields. These custom fields have to begin with an underscore. Applications which handle and display SOMIX 2 models are not required to handle custom fields.

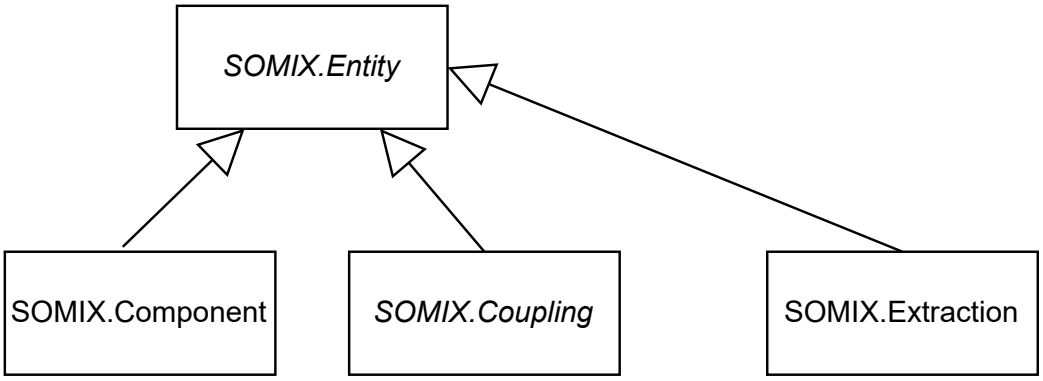

Figure 1: The classes of the SOMIX Metamodel

2.2. Classes

2.2.1. SOMIX.Entity

SOMIX.Entity is the abstract root class of the SOMIX metamodel entities.

Fields:

- In SAP2Moose: The mse model where it is contained.

2.2.2. SOMIX.Component extends SOMIX.Entity

SOMIX.Component is the abstract super class for all components in a software.

Fields:

- name: A mandatory string with the name of the grouping or component in the system.
- uniqueName: A mandatory unique string with the name of the grouping or component in the system.
  It has to be the same for all extractors which extract a certain element.

When this is not the case a mapping has to be provided to support matching elements.
uniqueName is always in lower case when the names are not case sensitive in the extracted system.
The combination of technicalType and uniqueName determines an element completly.

- title: A string with the title of the grouping or component in the system.
- technicalType: A mandatory string with information about the kind of grouping or component.
- linkToEditor: A link to open the specified element in an editor.
- hasCode: true when the component contains code
- hasData: true when the component contains data
- isPersistent: true when the component is persisten. In case of code this flag should in most cases be true. In case of data it marks data which is stored permanently, for instance in a database.
- isPartOf: Optional: When the component is part of another component: The instance of SOMIX.Component it is a part of.
- partSpecification: When isPartOf is set: A string where the kind of part is specified.

2.2.3. SOMIX.Coupling extends SOMIX.Entity

Parent-child relations:

- parent: An instance of SOMIX.Grouping.
- child: An instance of SOMIX.Element or SOMIX.Grouping.
- isMain: A boolean that the parent child relation should be shown always in a diagram. Only a single parent of a child should be flagged like this. Diagram tools should display the parent of a child always when this flag is set. This assures for instance that a class is always displayed when an attribute or method of a class is displayed in a diagram.

Calls:

- caller: An instance of SOMIX.Code.
- called: An instance of SOMIX.Code.

Accesses are used to specify read or write accesses. When all three flags isWrite, isRead, and isDependent are false, the Access has to be regarded as not existing by an analyzing application. When multiple Accesses exists for the same combination of components an analyzing application has to combine this into a single access. The fields isWrite, isRead, and isDependent are true when they are true in at least one of the combined accesses.

- accessor: An instance of SOMIX.Component.
- accessed: An instance of SOMIX.Data.
- isWrite: A flag whether a write access is made or might be made.
- isRead: A flag whether a read access is made or might be made.

It is possible to specify that a component types another component. In that case the typed component reads informations from the typing component to determine it's own properties.

- typedBy: The component which is used for typing.
- typed: The component which receives typing information.

2.2.4. SOMIX.Extraction extends SOMIX.Entity

Contains fields with metadata of the extraction. This is mandatory because the mse model itself contains no metadata. It is proposed in Ducasse et al. to store metadata as part of the model [1].

Fields:

- extractionTime: The date, time and timezone when the extraction was started.
- system: A string with the name of the extracted system.

Declaration of Competing Interests

The author declares that he has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was funded by CubeServ GmbH. Many colleagues provided valuable support to make this project possible. I thank Patrick Michels for providing resources and support.

References

[1] S. Ducasse, N. Anquetil, M.U. Bhatti, A.C. Hora, J. Laval, T. Girba, MSE and FAMIX 3.0: An Interexchange Format and Source Code Model Family. Research Report, Nov 2011. [Online], available: `https://hal.inria.fr/hal-00646884`.

[2] Y. Wand and R. Weber, An ontological model of an information system. IEEE transactions on software engineering, 16(11), 1282-1292, 1990.

[3] M.Bunge, Treatise on basic philosophy: Ontology II: A world of systems (Vol. 4). Boston, MA: Reidel, 1979.

[4] Winkler, Rainer Wolfgang, A Software Metamodel Restricted to Key Aspects of a Software System for Developers and a Method to Keep Manually Drawn Diagrams Up-to-Date and Correct. Available at SSRN: https://ssrn.com/abstract=4049604 or http://dx.doi.org/10.2139/ssrn.4049604.

[5] Moose2Model2, `https://github.com/Moose2Model/Moose2Model2` (accessed 12 September 2022).

[6] Moose2Model, `https://github.com/Moose2Model/Moose2Model` (accessed 30 December 2020).

[7] SAP2Moose, `https://github.com/SAP2Moose/SAP2Moose` (accessed 30 December 2020).