SAMOS - A Framework for Model Analytics and Management

Önder Babur[a,b], Loek Cleophas[b,c], Mark van den Brand[b]

[a] *Wageningen University & Research*
Wageningen, The Netherlands
[b] *Eindhoven University of Technology*
Eindhoven, The Netherlands
[c] *Stellenbosch University*
Matieland, Republic of South Africa

Abstract

The increased popularity and adoption of model-* engineering paradigms, such as model-driven and model-based engineering, leads to an increase in the number of models, metamodels, model transformations and other related artifacts. This calls for automated techniques to analyze large collections of those artifacts to manage model-* ecosystems. SAMOS is a framework to address this challenge: it treats model-* artifacts as data, and applies various techniques—ranging from information retrieval to machine learning—to analyze those artifacts in a holistic, scalable and efficient way. Such analyses can help to understand and manage those ecosystems.

Keywords: model-driven engineering, model analytics, software ecosystems, information retrieval, machine learning

1. Introduction

Model-* engineering approaches such as model-driven engineering and model-based engineering promote the use of models, metamodels and model transformations as first-class citizens to tackle the complexity of building and maintaining software-intensive systems. As such approaches have gained popularity and are applied to larger problems, however, the complexity, size

Email addresses: `Onder.Babur@wur.nl` (Önder Babur), `L.G.W.A.Cleophas@tue.nl` (Loek Cleophas), `M.G.J.v.d.Brand@tue.nl` (Mark van den Brand)

and variety of the involved artifacts (notably models) increase. This observation holds for open source as well as industrial ecosystems [1, 2].

In this paper we present SAMOS (Statistical Analysis of MOdelS), a framework to address the scalability issue in building and managing model-* ecosystems. It treats collections of models as data and offers various large-scale analysis techniques. It has so far been applied for different modelling languages such as metamodels, feature models, statecharts, domain-specific models, and business process models; and for scenarios such as domain analysis, repository management, clustering, outlier detection, language usage analysis, and clone detection [3, 4, 5, 6, 7, 8, 9, 10, 11]. In this initial public version (1.0), it includes the core framework, feature extraction for Ecore metamodels, as well as domain clustering and clone detection functionalities.

The rest of this paper is organized as follows. Section 2 summarizes some background concepts. Section 3 outlines the SAMOS architecture and functionalities. We illustrate SAMOS for domain clustering and clone detection for Ecore metamodels (Section 4). Section 5 concludes with future work.

2. Background

Information Retrieval (IR) tackles the challenge of efficiently indexing, analyzing and searching various forms of content such as text documents [12]. IR consists of a typical workflow. First, documents are collected and indexed, e.g. using a Vector Space Model (VSM) with (1) a vector representation of vocabulary occurrence in a document, named *term frequency*, (2) *zones* (e.g. 'author', 'title'), (3) weighting schemes such as inverse document frequency (idf), and zone weights, (4) NLP techniques for handling synonyms.

With the VSM we effectively transform each document into an n-dimensional vector, yielding an $m \times n$ matrix for m documents. Document similarity can be defined as the distance (e.g. Euclidean or cosine) between vectors in the VSM. We can use these distances for identifying similar groups of documents —i.e. clustering as an unsupervised machine learning (ML) technique.

Finally, n-grams [13] are used in computational linguistics to build probabilistic models of natural language text, e.g. for estimating the next word given a sequence of words, or comparing text collections based on their n-gram profiles. In essence, n-grams represent a linear encoding of structural context. Alternative (and increasingly more complex) fragmentations of structural context include sub-trees and sub-graphs. We will refer to n-grams and sub-trees while discussing SAMOS in the next section.

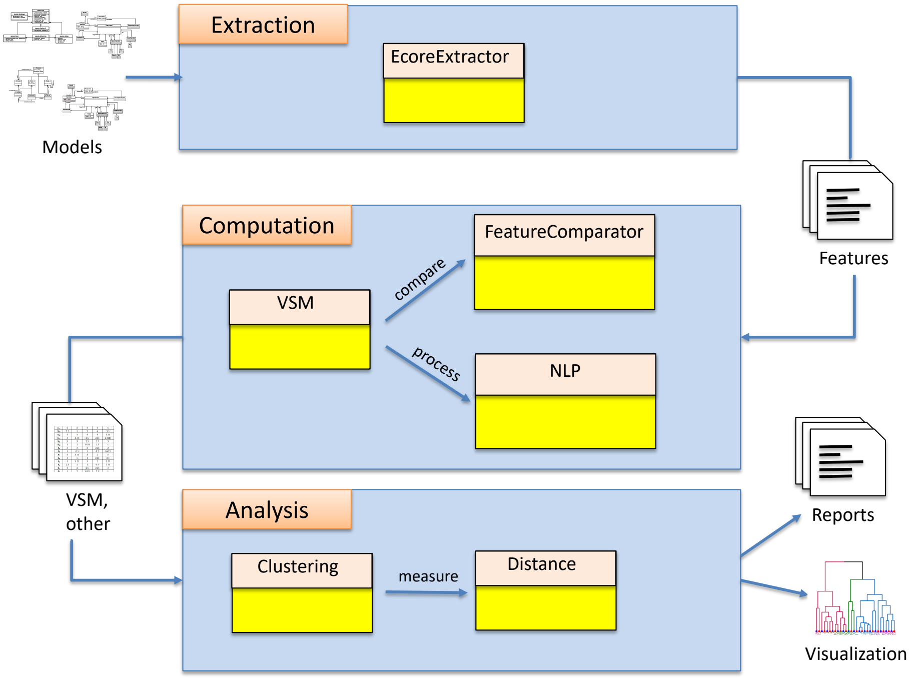

Figure 1: Overview of SAMOS Architecture.

3. SAMOS Framework

SAMOS applies the IR and ML-inspired approaches as summarized above to models. In this section, we outline the overall architecture, implementation details and the current functionalities offered by SAMOS.

3.1. Architecture

The architecture, as depicted in Figure 1, involves three main modules: extraction, computation and analysis. These are fairly independent components and the communication of data among them is achieved via a file-based pipeline. This approach so far has allowed us to plug-and-play different components, e.g. integrating an extractor for a different modelling language without changing the rest of the architecture.

A typical workflow for analytics starts with involves a metamodel-driven extraction (i.e. in module *Extraction*) of features from a set of input models. Features can be, for instance, singleton names of model elements (very similar to the vocabulary of documents) or larger fragments of the underlying graph structure such as n-grams or sub-trees. In our context, an example n-gram for a EMF metamodel would be for $n = 2$ an EClass containing an EAttribute [5]. The feature files are then fed to the *Computation* module. SAMOS computes a term-frequency based VSM, using feature comparison schemes (for instance maximum similar subsequence for n-grams), weighting schemes (for instance EClass weighted higher than EAttribute) and NLP such as tokenization, filtering, stemming and synonym checking. The results include a vector space model and other metadata files, which are then moved to the *Analysis* module. Applying various distance measures suitable to the problem at hand, SAMOS applies different clustering algorithms and can output automatically derived cluster labels or diagrams for visualization and manual inspection and exploration. For a detailed example-driven demonstration of the workflow, please refer to our previous work [3, 8].

3.2. Implementation

SAMOS is implemented in Java for feature extraction and VSM, and in R for the analytics and machine learning parts. SAMOS public version 1.0 as presented in this paper offers the following major functionalities:

- *Modelling languages:* SAMOS supports Ecore metamodels for feature extraction and subsequent steps of the analytics workflow.
- *Analyses:* SAMOS comes preconfigured to run in predefined settings for two types of analyses: domain clustering and clone detection.
- *Feature extraction:* Users can configure SAMOS to extract unigrams, bigrams and sub-trees with depth 1. Orthogonal to this configuration, They can choose to have basic vertex information (e.g. just the name of the model element) or the full details (i.e. all the attributes).
- *Fragmentation:* Users can specify the extraction and analysis scope: whole model or fragments (e.g. EPackages or EClasses in metamodels).
- *NLP components:* Users can configure NLP settings such as tokenization, lemmatization, WordNet synonym checking and threshold values.
- *Matching schemes:* Users can have type-based weights (e.g. EClass having more weight than EParameter) and idf (see Section 2).
- *Feature comparison:* Users can have different algorithms for feature comparison, e.g. maximum similar subsequence for n-grams or tree edit distance for subtrees [8].
- *Distance measures:* Different distance measures are used in the R scripts for different goals, e.g. cosine for domain clustering and masked Bray-Curtis for clone detection.

- *Clustering methods:* Similarly, different clustering methods are used for different goals, e.g. hierarchical clustering for domain clustering and density-based clustering for clone detection.

SAMOS is highly flexible and extendable, due to its simple and modular architecture. Over the course of the years parts of the workflow have been applied for various modelling languages and scenarios. These include, for instance, adding a feature extractor for a new modelling language [7] or adding a new statistical analysis at the end of the workflow [9]. We plan to gradually integrate these extensions into future release versions of SAMOS.

4. Illustrative Use Cases

While the reference work for SAMOS already includes plenty of demonstration and evaluation of SAMOS, we provide two introductory scenarios in this version. The sample runs contain a prerequisite step of crawling 19 Ecore metamodels from ATL Zoo[1], and running the following scenarios.

Domain clustering scenario. Users can run SAMOS with the domain clustering setting to see how conceptually related the model domains are. The predefined settings are: unigrams with just model element name information, whole model scoping, full components of NLP, type and idf weighting, simple n-gram comparison, cosine distance and hierarchical clustering. Clustering yields two types of output: one csv file with the list of cluster labels, and a pdf file visualizing the whole dataset in the form of a dendrogram. The dendrogram depicts how similar metamodels are to each other or other groups. The height values of the connection points of metamodels or groups map to intra-model or intra-group distances. The interpretation of the following dendrogram is that most of the models on the left main branch of the tree are conceptually very similar; after manual investigation they appear to be all conference management metamodels. The rightmost branch in turn has a cluster of bibliography management metamodels, although one item (Book.ecore) seems to be less similar to the rest of the cluster. For further reference on domain clustering on models, readers are referred to our previous work using SAMOS [3, 7, 9] and other related work [14].

[1] https://web.imt-atlantique.fr/x-info/atlanmod/index.php?title=Ecore

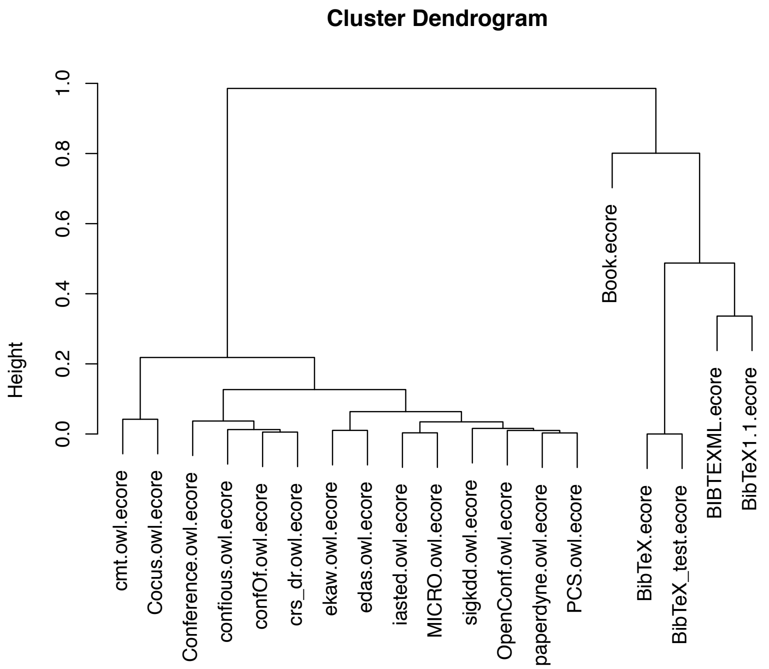

Figure 2: Plot of the dendrogram as output of domain clustering.

Clone detection scenario. Users can run SAMOS with the clone detection setting to see whether there are highly similar model fragments (*clones*) within or among metamodels. The predefined settings are: n-trees (depth 1) with just full model element attributes, EClass scoping, full components of NLP, type weighting only, Hungarian distance for subtree comparison, masked Bray-Curtis distance and density-based clustering. Clone detection yields three csv files corresponding to clusters of different clone types: Type A for identical fragments except layout and cosmetic differences, Type B for slightly different fragments (distance threshold 10%), and Type C for substantially different fragments (distance threshold 30%), with the threshold values chosen with respect to our previous work [8]. Table 1 depicts an excerpt for Type B clone fragments resulting from the sample run. Each row corresponds to a clone cluster with a cluster label (first column), list of model fragments separated by whitespace (second column), and the average size of the cluster in terms of total number of model elements. The model fragments are encoded in the format `MODEL_NAME$FRAGMENT_NAME`, and the clusters are sorted with respect to their average sizes.

cluster label	models	average cluster size
4	MICRO.owl.ecore$Person sigkdd.owl.ecore$Person OpenConf.owl.ecore$Person …	50.5
15	PCS.owl.ecore$Paper OpenConf.owl.ecore$Paper sigkdd.owl.ecore$Paper …	46.5
14	MICRO.owl.ecore$Conference PCS.owl.ecore$Conference edas.owl.ecore$Conference …	33.25
...	...	...

Table 1: Clone detection results for Type B clones in the sample dataset.

SAMOS can also download our labelled dataset on Zenodo [15]. Users can apply domain clustering and clone detection on this significantly larger dataset. While there is no other related work for metamodel clone detection, readers can refer to literature for clone detection on other (non-meta-)model types, for instance, for Simulink models [16] and UML models [17].

5. Conclusions and Future Work

In this paper we presented the SAMOS framework for model analytics and management, in its public version 1.0. We briefly outlined the motivation for developing this framework, its workflow and main functionalities. We also provided two illustrative scenarios for domain clustering and clone detection. As already mentioned in the introduction, SAMOS has been in development for many years with a lot of (partly experimental, not all published) extensions for different languages, settings and analyses. We plan to gradually integrate those extensions into SAMOS, including but not limited to:

- Major performance optimizations, including iterative clone detection [8],
- Support for other languages: feature models [7], UML [6], BPMN [10], Simulink,
- Other analyses and ML algorithms such as topic modelling [9],
- High performance computing using Apache Spark [18],
- Mining software repositories and Neo4j bridges [11].

References

[1] Ö. Babur, L. Cleophas, M. van den Brand, B. Tekinerdogan, M. Aksit, Models, more models, and then a lot more, in: M. Seidl, S. Zschaler (Eds.), Software Technologies: Applications and Foundations, Springer International Publishing, Cham, 2018, pp. 129–135.

[2] B. Tekinerdogan, Ö. Babur, L. Cleophas, M. van den Brand, M. Aksit, Introduction to model management and analytics for large scale systems, in: B. Tekinerdogan et al. (Ed.), Model Management and Analytics for Large Scale Systems, Elsevier, 2019, pp. 3–11.

[3] Ö. Babur, L. Cleophas, M. van den Brand, Hierarchical clustering of metamodels for comparative analysis and visualization, in: Proc. of the 12th European Conf. on Modelling Foundations and Applications, 2016, pp. 2–18.

[4] Ö. Babur, Statistical analysis of large sets of models, in: Proc. of the 31st IEEE/ACM Int. Conf. on Automated Software Engineering, ACM, 2016, pp. 888–891.

[5] Ö. Babur, L. Cleophas, Using n-grams for the automated clustering of structural models, in: 43rd Int. Conf. on Current Trends in Theory and Practice of Computer Science, 2017, pp. 510–524.

[6] D. Wille, Ö. Babur, L. Cleophas, C. Seidl, M. van den Brand, I. Schaefer, Improving custom-tailored variability mining using outlier and cluster detection, Science of Computer Programming 163 (2018) 62–84.

[7] Ö. Babur, L. Cleophas, M. van den Brand, Model analytics for feature models: case studies for S.P.L.O.T. repository, in: Proc. of MODELS 2018 Workshops, co-located with ACM/IEEE 21st Int. Conf. on Model Driven Engineering Languages and Systems, 2018, pp. 787–792.

[8] Ö. Babur, L. Cleophas, M. van den Brand, Metamodel clone detection with SAMOS, Journal of Computer Languages 51 (2019) 57 – 74.

[9] Ö. Babur, A. Suresh, W. Alberts, L. Cleophas, R. Schiffelers, M. van den Brand, Model analytics for industrial MDE ecosystems, in: Model Management and Analytics for Large Scale Systems, Elsevier, 2020, pp. 273–316.

[10] M. S. Nikoo, Ö. Babur, M. van den Brand, Business process model clone detection, Submitted to a journal (2022) .

[11] Ö. Babur, E. Constantinou, A. Serebrenik, Language usage analysis for EMF metamodels on GitHub, Submitted to a journal (2022) .

[12] C. D. Manning, P. Raghavan, H. Schütze, et al., Introduction to information retrieval, Vol. 1, Cambridge University Press, 2008.

[13] C. D. Manning, H. Schütze, Foundations of statistical natural language processing, Vol. 999, MIT Press, 1999.

[14] F. Basciani, J. Di Rocco, D. Di Ruscio, L. Iovino, A. Pierantonio, Automated clustering of metamodel repositories, in: Int. Conf. on Advanced Information Systems Engineering, Springer, 2016, pp. 342–358.

[15] Ö. Babur, A labeled Ecore metamodel dataset for domain clustering (Mar. 2019). doi:10.5281/zenodo.2585456.
URL `https://doi.org/10.5281/zenodo.2585456`

[16] M. H. Alalfi, J. R. Cordy, T. R. Dean, M. Stephan, A. Stevenson, Models are code too: Near-miss clone detection for simulink models, in: 28th Int. Conf. on Software Maintenance, IEEE, 2012, pp. 295–304.

[17] H. Störrle, Towards clone detection in UML domain models, Software & Systems Modeling 12 (2) (2013) 307–329.

[18] Ö. Babur, L. Cleophas, M. van den Brand, Towards distributed model analytics with Apache Spark, in: Proc. of the 6th Int. Conf. on Model-Driven Engineering and Software Development, 2018, pp. 767–772.

Required Metadata

Current executable software version

Nr.	**(executable) Software metadata description**	
S1	Current software version	1.0
S2	Permanent link to executables of this version	`https://github.com/onderbabur/samos/archive/refs/heads/main.zip`
S3	Legal Software License	MIT license
S4	Computing platform/Operating System	MacOS, Linux, Windows
S5	Installation requirements & dependencies	Java SE JDK 1.8, Eclipse Modeling Tools 2021-03, Eclipse plugin ”m2e”, R version 3.6.1, R package ”rJava” version 3.6.2 and custom package ”vegan”
S6	If available, link to user manual - if formally published include a reference to the publication in the reference list	`https://onderbabur.github.io/samos/`
S7	Support email for questions	`Onder.Babur@wur.nl`

Table 2: Software metadata (optional)

Current code version

Nr.	Code metadata description	
C1	Current code version	1.0
C2	Permanent link to code/repository used of this code version	For example: `https://github.com/onderbabur/samos`
C3	Legal Code License	MIT license
C4	Code versioning system used	git
C5	Software code languages, tools, and services used	Java SE JDK 1.8, Eclipse Modeling-Tools 2021-03, Eclipse plugin "m2e" version 1.18, R version 3.6.1, R package "rJava" version 3.6.2 and custom package "vegan"
C6	Compilation requirements, operating environments & dependencies	MacOS, Linux, Windows
C7	If available Link to developer documentation/manual	`https://onderbabur.github.io/samos/`
C8	Support email for questions	`Onder.Babur@wur.nl`

Table 3: Code metadata (mandatory)