Towards a Body of Knowledge for Model-Based Software Engineering

Federico Ciccozzi, Michalis Famelis, Gerti Kappel, Leen Lambers, Sébastien Mosser, Richard Paige, Alfonso Pierantonio, Arend Rensink, Rick Salay, Gabi Taentzer, et al.

Towards a Body of Knowledge for Model-Based Software Engineering

Federico Ciccozzi
Mälardalen University
Västerås, Sweden
federico.ciccozzi@mdh.se

Michalis Famelis
Université de Montréal
Montreal, Canada
famelis@iro.umontreal.ca

Gerti Kappel
CDP, TU Wien
Vienna, Austria
gerti@big.tuwien.ac.at

Leen Lambers
Hasso-Plattner-Institut
Potsdam, Germany
Leen.Lambers@hpi.de

Sebastien Mosser
Université Côte d'Azur, CNRS, I3S
Sophia Antipolis, France
mosser@i3s.unice.fr

Richard F. Paige
University of York
York, UK
richard.paige@york.ac.uk

Alfonso Pierantonio
University of L'Aquila
L'Aquila, Italy
alfonso.pierantonio@univaq.it

Arend Rensink
University of Twente
Enschede, The Netherlands
arend.rensink@utwente.nl

Rick Salay
University of Toronto
Toronto, Canada
rsalay@cs.toronto.edu

Gabi Taentzer
Philipps-Universität Marburg
Marburg, Germany
taentzer@informatik.uni-marburg.de

Antonio Vallecillo
Universidad de Málaga
Málaga, Spain
av@lcc.uma.es

Manuel Wimmer
CDL-MINT, TU Wien
Vienna, Austria
wimmer@big.tuwien.ac.at

ABSTRACT

Model-based Software Engineering (MBSE) is now accepted as a Software Engineering (SE) discipline and is being taught as part of more general SE curricula. However, an agreed core of concepts, mechanisms and practices — which constitutes the Body of Knowledge of a discipline — has not been captured anywhere, and is only partially covered by the SE Body of Knowledge (SWEBOK). With the goals of characterizing the contents of the MBSE discipline, promoting a consistent view of it worldwide, clarifying its scope with regard to other SE disciplines, and defining a foundation for a curriculum development on MBSE, this paper provides a proposal for an extension of the contents of SWEBOK with the set of fundamental concepts, terms and mechanisms that should constitute the MBSE Body of Knowledge.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; **Model curricula**;

KEYWORDS

Model-Based Software Engineering, Body of Knowledge.

MODELS '18 Companion, October 14–19, 2018, Copenhagen, Denmark

ACM ISBN 978-1-4503-5965-8/18/10…$15.00
https://doi.org/10.1145/3270112.3270121

ACM Reference Format:
Federico Ciccozzi, Michalis Famelis, Gerti Kappel, Leen Lambers, Sebastien Mosser, Richard F. Paige, Alfonso Pierantonio, Arend Rensink, Rick Salay, Gabi Taentzer, Antonio Vallecillo, and Manuel Wimmer. 2018. Towards a Body of Knowledge for Model-Based Software Engineering. In *ACM/IEEE 21th International Conference on Model Driven Engineering Languages and Systems (MODELS '18 Companion), October 14–19, 2018, Copenhagen, Denmark.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3270112.3270121

1 INTRODUCTION

Model-based Software Engineering (MBSE) is becoming widely accepted as a software engineering (SE) discipline that promotes the use of models and model transformations, as the fundamental elements of all software development processes. As its adoption grows and it becomes part of more general SE curricula, there is the need to define an agreed core of the concepts, elements, mechanisms and practices that MBSE encompasses. Such a compendium corresponds to the concept of "Body of Knowledge" (BoK) used in many engineering disciplines, and which comprises the complete set of concepts, terms and activities that make up a professional domain, being a fundamental part of any profession [19].

Similarly to the objectives of other existing BoKs, the goal of a BoK for MBSE (hereinafter MBEBOK) would be threefold: (a) to characterize the contents of the MBSE discipline and to promote a consistent view of MBSE worldwide; (b) to clarify the scope and the place of MBSE within SE and with respect to other disciplines such as computer science and system engineering; and (c) to define a foundation for curriculum development and for individual certification and licensing material.

In 2004, the IEEE Computer Society published the "Guide to the Software Engineering Body of Knowledge" (hereinafter SWEBOK),

which was updated in 2014 [6] and provides a comprehensive and agreed description of the BoK for SE. It was also adopted by ISO/IEC as ISO/IEC TR 19759:2005. However, although it embraces the concept of model and modelling in Software Engineering (its Chapter 9 is entirely devoted to that), it does not provide a complete and detailed catalogue of all the core concepts needed to portray MBSE and its usage. In particular, currently it does not detail the fundamental concepts, techniques, methodologies and tool types that any MBSE practitioner should be aware of, or the skills required to master them.

In January 2018, during the discussions that took place as part of the First Winter Modeling Meeting (WMM'18) in San Vigilio di Marebbe, Italy, the authors of this paper decided to start working on the main contents of a Guide to MBEBOK. Instead of starting from scratch, we decided to analyse and extend the current contents of the SWEBOK, and in particular its Chapter 9, trying to respect its structure but complementing it with what we considered the fundamental concepts and terms for MBSE.

This paper reports the results of those efforts and provides a proposal for a set of fundamental concepts, terms and mechanisms that should constitute the MBEBOK.

The paper is structured in 5 sections. After this introduction, Section 2 discusses what a BoK is, and briefly presents the SWEBOK and its current coverage of software models. Then, Section 3 identifies the basic MBSE concepts that are not included in the SWEBOK. Section 4 describes some issues that were identified during the development of the contents of the MBEBOK, and for which a discussion within the community would be needed. Finally, Section 5 concludes the paper and outlines some future activities. A final Annex provides an initial Glossary of MBSE terms, with the key definitions of the main concepts of the discipline. Such a glossary is pivotal to achieve the goals of the MBEBOK, being to provide a consistent view of MBSE and a foundation for curriculum development and individual certification.

2 BACKGROUND

2.1 Bodies of Knowledge

A Body of Knowledge (BoK) is a set of concepts, terminology and tasks that constitute a professional domain. Often, a BoK is developed by a professional association (e.g., ACM, IEEE), and captures the knowledge that is inherent, sometimes tacit, and often explicit in the interactions and literature that occur in that professional domain.

The main goals of a BoK on a given discipline are:

- To promote a consistent view of the discipline worldwide;
- To specify the scope of the discipline and to clarify its place with respect to other related disciplines;
- To characterize the contents and known practices of the discipline, organizing them in a coherent and comprehensive manner;
- To provide a foundation for curriculum development and, when applicable, for individual certification and licensing material.

A BoK should also provide concrete deliverables:

- A terminology that defines the set of main concepts of the discipline, as used by their practitioners. It constitutes the accepted ontology for the specific domain.
- A structured list of the main knowledge areas, skills and accepted practices of the discipline, covering all the basic knowledge that any practitioner should posses.

A BoK should always be descriptive, but not prescriptive: intentionally, it should not impose any particular method or tool, or any specific practice.

With respect to what should be considered as "generally recognized" or as a "good practice" of a discipline, the Project Management Body of Knowledge (PMBOK) [20] offers more precise descriptions. First, "generally recognized" means that the knowledge and practices described are generally applicable to many different kinds of projects in many situations, and there is consensus about their value and usefulness. In turn, when we say "Good practice" it means that there is general agreement that the application of these skills, tools, and techniques can enhance the chances of success over a wide range of projects. It does not mean, however, that the precise knowledge or practice should always be applied to all projects; the organization and/or project management team should be the ultimate responsible for determining what practices are more appropriate for a given project in a certain situation.

Currently there are a number of BoKs for various software-related disciplines; some are mature documents with a rigorous review and revision process (e.g., SWEBOK), whilst others are evolving (e.g., SLEBOK):

- Software Engineering Body of Knowledge (SWEBOK) [6]
- Systems Engineering Body of Knowledge (SEBOK) [2]
- Data Management Body of Knowledge (DMBOK) [10]
- Enterprise Architecture Body of Knowledge (EABOK) [12]
- Business Analysis Body of Knowledge (BABOK) [13]
- Project Management Body of Knowledge (PMBOK) [20]
- Automation Body of Knowledge (ABOK) [22]
- Software Language Engineering BoK (SLEBOK) [25]

2.2 The Software Engineering BoK (SWEBOK)

In 2004, the IEEE Computer Society established for the first time a baseline for the body of knowledge for the field of software engineering. It was the outcome of a joint committee with ACM, whose mission was "to establish the appropriate sets(s) of criteria and norms for professional practice of software engineering upon which industrial decisions, professional certification, and educational curricula can be based."

The SWEBOK was developed as an international collective effort, in order to achieve the goal of providing a consistent worldwide view of software engineering. The committee appointed two chief editors, several co-editors to support them, and editors for each of the Knowledge Areas. All chapters were openly reviewed, in an editing process that engaged approximately 150 reviewers from 33 countries. Professional and scientific societies, as well as public agencies from all over the world involved in software engineering were contacted, made aware of this project, and invited to participate in the review process too. Presentations on the project were made at various international venues. The 2004 edition was revised

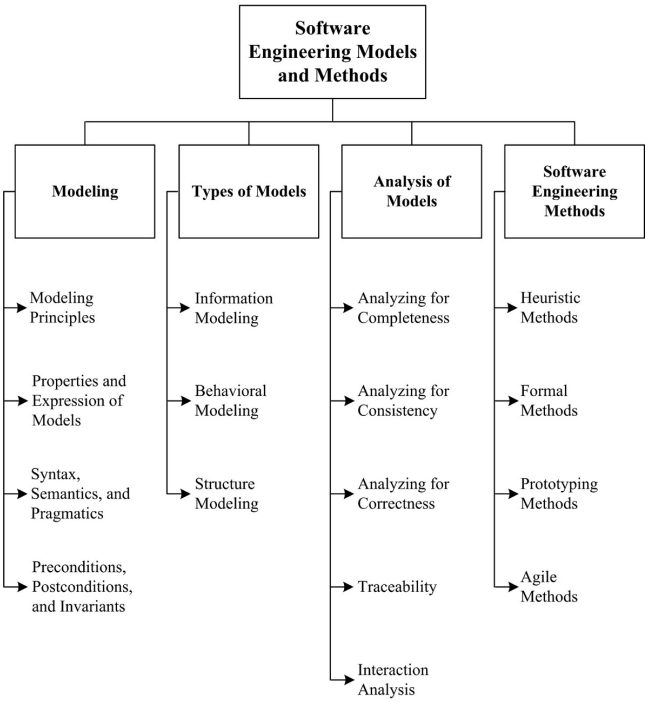

Figure 1: Breakdown of Topics for the Software Engineering Models and Methods KA in SWEBOK (from [6]).

in 2014, using the same edition process, giving birth in 2014 to the current version (v3) of the SWEBOK [6].

It should be noted that the SWEBOK provides a "Guide to the Software Engineering BoK", but it does not aim at describing the entire body of knowledge for software engineering. Instead, the SWEBOK serves as a guide to the existing BoK that has been developed since the start of the discipline.[1]

SWEBOK V3.0 defines 15 Knowledge Areas (KA), plus an Appendix that lists the IEEE and ISO/IEC International Standards supporting the SWEBOK.

One complete Knowledge Area (Chapter 9) of the SWEBOK is dedicated to *Software Engineering Models and Methods*. As stated in the SWEBOK, "software engineering models and methods impose structure on software engineering with the goal of making that activity systematic, repeatable, and ultimately more success-oriented. Using models provides an approach to problem solving, a notation, and procedures for model construction and analysis. Methods provide an approach to the systematic specification, design, construction, test, and verification of the end-item software and associated work products."

Figure 1 shows the breakdown of topics for the SE Models and Methods Knowledge Area (Chapter 9). We list them below:

- Modeling: discusses the general practice of modeling, and presents:
  - The basic modeling concepts and principles
  - Properties (completeness, consistency correctness) and expression of models (as typed and attributed element representing entities and associations representing relationships among them, using graphical or textual notations)
  - Syntax, Semantics and Pragmatics of models
  - Preconditions, postconditions and invariants as specification mechanisms
- Type of models: briefly discusses models and aggregation of submodels and provides some general characteristics of model types commonly found in the software engineering practice, including:
  - Information models (aka conceptual models)
  - Behaviour models (state machines, control-flow models, dataflow models)
  - Structure models (e.g., UML class, component, object, deployment, and packaging diagrams)
- Analysis of models: presents some of the common analysis techniques used in modeling to verify:
  - Completeness
  - Consistency
  - Correctness
  - Traceability
  - Interaction analysis
- Software Engineering Methods: presents a brief summary of commonly used software engineering methods, including heuristic methods, formal methods, prototyping, and agile methods. This part is more general, and aimed to apply to any SE discipline, not only to MBSE.

By looking at them, we see that the coverage of MBSE concepts and mechanisms is rather appropriate, but some essential concepts of MBSE — such as model transformations, executable models, or code generation, for instance — which should be part of the education of any MBSE practitioner, are not contemplated in the SWEBOK.

Furthermore, although the SWEBOK provides definitions for some MBSE concepts, not all of them are precisely defined, and even in some cases the SWEBOK definitions miss some important features and characteristics of the defined concepts that have been later identified by the modeling community. In this respect, providing precise definitions for all the main MBSE terms is another goal of this proposal.

Apart from the SWEBOK topics mentioned above, the SWEBOK lists in its Annex B the International Standards related to the Software Engineering Models and Methods KA. There are three groups of standards, depending on their scope.

First it lists the standards about modeling notations:

- IEEE Std. 1320.1-1998 Standard for Functional Modeling Language – Syntax and Semantics for IDEF0
- IEEE Std. 1320.2-1998 Standard for Conceptual Modeling Language – Syntax and Semantics for IDEF1X97 (IDEFobject)
- ISO/IEC 19501:2005 Information Technology – Open Distributed Processing – Unified Modeling Language (UML) Version 1.4.2
- ISO/IEC 19505:2012 [two parts] Information Technology – Object Management Group Unified Modeling Language (OMG UML)

[1] It is generally agreed that Software Engineering as a distinct discipline originated in the NATO conference of the same name, held in Germany in 1968 to discuss, for the first time, the software crisis.

- ISO/IEC 19506:2012 Information Technology – Object Management Group Architecture-Driven Modernization (ADM) – Knowledge Discovery Meta-Model (KDM)
- ISO/IEC 19507:2012 Information Technology – Object Management Group Object Constraint Language (OCL)

A second group of standards is related to tools:

- IEEE Std. 14102-2010 Standard Adoption of ISO/ IEC 14102:2008 Information Technology – Guideline for the Evaluation and Selection of CASE Tools
- IEEE Std. 14471-2010 Guide – Adoption of ISO/IEC TR 14471:2007 Information Technology – Software Engineering – Guidelines for the Adoption of CASE Tools
- IEEE Std. 1175.1-2002 Guide for CASE Tool Interconnections – Classification and Description
- IEEE Std. 1175.2-2006 Recommended Practice for CASE Tool Interconnection – Characterization of Interconnections
- IEEE Std. 1175.3-2004 Standard for CASE Tool Interconnections – Reference Model for Specifying Software Behavior
- IEEE Std. 1175.4-2008 Standard for CASE Tool Interconnections – Reference Model for Specifying System Behavior

The third group of standards included in the SWEBOK for the SE Models and Methods KA is related to the environments of the systems:

- ISO/IEC/IEEE 26515:2012 Systems and Software Engineering – Developing User Documentation in an Agile Environment
- ISO/IEC 15940:2006 Information Technology – Software Engineering Environment Services

Finally, there is one standard common to all SWEBOK Knowledge Areas, about terminology.

- ISO/IEC/IEEE 24765:2010 Systems and Software Engineering – Vocabulary

The inclusion of references to standards in the MBEBOK is something that will be discussed later in Section 4.

2.3 The Software Language Engineering BoK (SLEBOK)

The field of Software Language Engineering (SLE) has emerged to connect and integrate different research disciplines such as compiler construction, reverse engineering, software transformation, model-driven engineering, and ontologies, in order to identify the principles and practices of engineering software languages — i.e., artificial languages that may ultimately be implemented on a machine. SLEBOK is an ongoing initiative to capture a Body of Knowledge on SLE. A Dagstuhl seminar [9] was held to capture a preliminary set of artefacts, definitions, methods, best practices, open challenges, etc. The intent was for these to be consolidated in the SLEBOK, which is currently evolving. Whilst SWEBOK and several other BoKs have a mature process for their continued evolution and development, SLEBOK does not yet have such a process. However, SLEBOK is noteworthy in that it is very open, and via its Git repository, anyone can contribute to the revision process.

MBE principles and techniques can be used for Software Language Engineering (e.g., metamodels can be used to define the abstract syntax of software languages). As such, there are relationships between SLEBOK and MBEBOK. These relationships are still evolving, due to the relative immaturity of both BoKs, but we can make some key observations:

- SLEBOK defines notions of *model* and *metamodel* which are not incompatible with MBEBOK, though MBEBOK's definitions are more elaborated.
- SLEBOK does mention the notion of static semantics, but does not elaborate on language semantics, whereas MBEBOK explicitly captures semantics as a key concept.
- SLEBOK and MBEBOK treat abstract and concrete syntax differently; whilst these are first-class concepts in MBEBOK, their notions are distributed across multiple concepts in SLEBOK.

3 TOPICS FOR A BOK ON MODEL-BASED SOFTWARE ENGINEERING

The following list of topics contain those that we consider highly relevant for the MBEBOK, since they must be part of the knowledge that any MBSE practitioner should posses. The list below is in no particular order and the structure is indicative. Topics marked with an ampersand (&) indicate that they are already covered in the SWEBOK. Topics marked with a hash (#) indicate that they are partially covered in the SWEBOK, or simply mentioned. The numbers accompanying the marks show the SWEBOK section in which these topics are described.

- Model Foundations
  - Syntax
    - Abstract vs. concrete syntax
    - Textual vs. visual models
  - Semantics
    - Structural (#2.2)
    - Behavioral (discrete vs. continuous) (#2.3)
    - Informational (&2.1)
  - Purpose/intent
    - Modeling principles (&1.1)
    - Exemplar purposes: such as Metamodeling or model transformation definition
- Multiview Modeling
- Model Quality
  - Completeness (&3.1)
  - Consistency (&3.2)
  - Correctness (&3.3)
  - Comprehensibility
  - Confinement (= fitness for purpose)
  - Changeability
- Modeling Languages
  - Language definition
    - Metamodels
    - Grammars
    - Semantics (by e.g. Abstract State Machines or model transformations)
  - Types of modeling languages
    - General purpose (GPL): UML+OCL, SysML
    - Domain-specific (DSL): UML Profiles, Language Workbenches, ADLs, ...
- Model Maintenance and Evolution
  - Versioning

  - Migration
- Model Visualization
  - Physics of notations
  - Layout
  - Animation
- Model Execution
  - Model simulation (#3.5)
    - Model co-simulation (simulation of a hybrid model)
  - Execution strategies (sequential vs. parallel)
  - Model debugging and testing
- Model Transformations
  - Model transformation languages
    - Syntax
    - Semantics
  - Model transformation types [16]
    - Text-to-Model, Model-to-Model, Model-to-Text
    - Exogenous vs. endogenous
    - In-place vs. out-place
    - Horizontal vs. vertical
    - Uni-directional vs. bidirectional
    - Syntactical vs. semantic
    - Transformation paradigm (declarative vs. operational)
  - Model transformation applications
    - Model translation (synthesis, code generation, reverse engineering, migration, optimization, refactoring, refinement, adaptation) [15]
    - Model merge
    - Model differencing
    - Model weaving
    - Model synchronization (#3.4)
    - Model interpretation (incl. execution)
- Analysis
  - Structural model analysis
    - Invariant checking (#1.4)
    - Instance generation
    - Metrics calculation
    - Smells detection
  - Behavioural model analysis
    - Pre-postcondition checking (#1.4)
    - Simulation (#3.5)
    - Reachability analysis
    - Temporal model checking
    - Performance
  - Model transformation analysis
    - Correctness (of transformed models, in syntax and semantics)
    - Completeness
    - Functional behaviour (termination, confluence)
    - Performance

4 ISSUES FOR DISCUSSION

This section identifies some issues that were discussed during the preparations of the contents of the MBEBOK, but for which no clear decision was reached. We think that they are sufficiently important to be discussed within the MBSE community, and in this sense the MODELS Educators Symposium provides an excellent forum where they can be debated.

4.1 Integration with the SWEBOK

Section 3 above has listed the main concepts that a MBEBOK should contain, in order to extend the current contents of the SWEBOK. However, there are several alternatives for implementing such an extension.

In the first place, we could try to respect the current SWEBOK structure and contents, just adding some paragraphs to the existing text, and some new subsections if needed. This kind of conservative approach to the extension was our original aim, but it somehow forces an unnatural structure in the contents of the chapter for the new concepts.

A second option would be to replace some complete sections of Chapter 9, providing a more natural and cohesive structure of the concepts. However, this would mean rewriting most parts of the chapter, something which would end up being really disruptive with respect to the current version of SWEBOK.

We even discussed the possibility, as the third alternative, of creating a complete BoK, separating from SWEBOK. This is the approach followed by, e.g., the SLEBOK. This option would have the main advantage of creating a dedicated BoK that would perfectly fit with our discipline, including not only the modeling concepts but also specializing other aspects such as design or testing. As weak points, being an Engineering discipline we would be forced to repeat many portions of the SWEBOK, since our intersection is by no means small (ranging from foundations to economic and professional issues, to mention just two extreme topics).

4.2 Integration with the SLEBOK

How to design, develop and maintain modeling languages could also be considered as very important for any MBSE practitioner. In fact, a number of topics listed in Section 3 already cover several aspects related to Software Language Engineering that apply to modeling languages. Although in theory the SLEBOK should address most of these topics, by looking at its current contents it does not seem to cover some MBSE topics adequately. Simple topics such as syntax and semantics are not covered in their generality, or at least with not enough level of detail for our purposes.

Similarly to what we are proposing here with the SWEBOK, the relationship between the MBEBOK and the SLEBOK should be clarified, stating the scopes of both BoKs, identifying their intersecting concepts and mechanisms, and making sure they are treated consistently in both Guides. This is not a trivial issue, given the current status of the SLEBOK and its lack of a stable version. However, the fact that the SLEBOK is still under development can also be an advantage: now that we have identified the topics that should be part of the MBEBOK, it would be a matter of defining an integration strategy that permit complementing the contents of both BoKs in a successful manner.

4.3 Further topics to be considered

In addition to the topics listed in Section 3, during the internal discussions carried out among the authors of this paper, some further topics that could also be of interest for the MBEBOK were

identified. It is still to be decided whether they should be part of the MBEBOK or not. Again, we wanted to raise these issues in this section, so that they can be easily identified and discussed within the MBSE community.

The initial list of topics that we think that would require some discussion are the following:

4.3.1 Application domains. Models are currently used with a significant level of success in different research and industrial settings, where the application of Model-Based Software Engineering practices can provide interesting benefits. Automotive and Cyber-Physical Systems are examples of these application domains. Although a description on how models are used in these domains would be beneficial for any MBSE practitioner, and should be part of their background, it was not clear whether they need to be incorporated into the MBEBOK or not.

4.3.2 Advanced topics. In addition to the topics listed in Section 3, some further topics were also considered for inclusion in the MBEBOK, including:

- Megamodels
- Models@run.time
- Multilevel modeling

Should these, as well as other emerging concepts (e.g., streaming models or partial models) and techniques (e.g., incremental or non-deterministic model transformations) be included in the MBEBOK? Where should the line be drawn? In this respect, drawing a line is also a matter of abstraction. What are the general topics? Which ones are too specific?

4.3.3 International Standards. The references to modeling standards in the SWEBOK (and hence in the MBEBOK) were questioned. Some of the standards currently listed in the SWEBOK do not seem to be relevant or widely used any more by the modeling community (in particular, most of the tool-related standards). Other references and documented practices are however highly recognized and widely adopted (e.g. those from the Eclipse Foundation, acting as de-facto standards. Should they be included in the MBEBOK, too? The problem with MBSE-related standards is that they may evolve too rapidly, which is an argument for their exclusion (like for MBSE tools, which for that reason we chose to omit — see Section 4.4). For example, the SWEBOK mentions the initial versions of UML (1.4, 2.0), now completely superseded. On the other hand, not including any international standard would probably send a wrong message to industry — specially because MBSE practitioners do use standards and do care about interoperability and reusability of the artefacts of the discipline.

4.3.4 Application scenarios. In addition to the code generation and analysis possibilities provided by MBSE, mentioned in Section 3, there are other scenarios in which principles of MBSE play an important role. A non-exhaustive list:

- *Model mining* — the principle of automatically deriving models from existing repositories and logs; for instance, structural models [7], feature models [11], database schemas [8] or process models [1].
- *Model learning* — the principle of actively deriving models by well-chosen invocation of functionality; e.g., [21].
- *Model-based testing* — the principle of automatically deriving tests from behavioural models; e.g., [24].
- *Model-based modernization* — the principle of using high-level models of an existing system to represent it, and later use model-driven techniques on these models to refactor, improve, or migrate it to other platforms; see, e.g., [14, 18].
- *Model checking* — the principle of verifying a system by systematically exploring its behavioural model; see, e.g., [3].

A description of these application scenarios can help MBSE practitioners understand the scope of the discipline and of its current application areas (within the broader Software Engineering field). The question is whether these (or other) application scenarios should be included in the MBEBOK or not; and if so, at which level of detail.

4.4 Topics not covered in this proposal

Some topics were by design omitted from this proposal: in particular, concrete tools and engineering processes. The former were omitted because they typically evolve too rapidly to be part of a BoK. With respect to specific MBSE processes, the current coverage of SE processes in the SWEBOK could be sufficient for our purposes; otherwise we would need a specialization of the complete SWEBOK, something that was not the main goal of this initial proposal.

5 CONCLUSIONS AND FUTURE WORK

This paper has presented an initial proposal for an extension of the contents of SWEBOK with the set of fundamental concepts, terms and mechanisms that should constitute the MBSE Body of Knowledge. As such, it aims to characterize the contents and known practices of the MBSE discipline and, in particular, to assist universities and other institutions that provide teaching courses on SE to develop their MBSE curricula.

This proposal is designed to serve as a working document that can be used to foster the discussions within the MBSE community about the main contents of the MBEBOK, its relationships with other BoKs, and about how to extend the current contents of the SWEBOK with the specific MBSE topics identified here.

Once this proposal is discussed among the members of the modeling community, and completed with their suggestions and recommendations, we plan to prepare a white paper that could serve to raise the discussions on a MBEBOK to wider forums and other communities (SLE, general SE), and eventually lead to a concrete and agreed proposal for the contents of a MBEBOK.

Acknowledgements. We would like to thank the reviewers for their valuable comments and suggestions. This paper has been partially funded by the following research projects and grants: Spanish Research Project TIN2014-52034-R, by the Austrian Research Promotion Agency (FFG) via the Austrian Competence Center for Digital Production (CDP) under the contract number 854187, by the Austrian Federal Ministry of Science, Research and Economy and the National Foundation for Research, Technology and Development, and the Knowledge Foundation (KKS) through the MOMENTUM project.

REFERENCES

[1] W.M.P van der Aalst. 2011. Process discovery: An Introduction. In *Process Mining*. Springer, Chapter 5, 125–156. https://doi.org/10.1007/978-3-642-19345-3_5

[2] Rick Adcock (Ed.). 2017. *Guide to the Systems Engineering Body of Knowledge (SEBoK), Version 1.9*. https://www.sebokwiki.org

[3] C. P. Baier and J. P. Katoen. 2008. *Principles of Model Checking*. MIT Press.

[4] Jean Bézivin, Frédéric Jouault, and Patrick Valduriez. 2004. On the Need for Megamodels. In *Proceedings of the OOPSLA/GPCE 2004 workshop on Best Practices for Model-Driven Software Development*. https://hal.archives-ouvertes.fr/hal-01222947

[5] Gordon S. Blair, Nelly Bencomo, and Robert B. France. 2009. Models@run.time. *IEEE Computer* 42, 10 (2009), 22–27. https://doi.org/10.1109/MC.2009.326

[6] Pierre Bourque and Richard E. Fairley (Eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 3.0*. IEEE Computer Society Press. https://www.computer.org/web/swebok

[7] Hugo Brunelière, Jordi Cabot, Grégoire Dupé, and Frédéric Madiot. 2014. MoDisco: A model driven reverse engineering framework. *Information & Software Technology* 56, 8 (2014), 1012–1032. https://doi.org/10.1016/j.infsof.2014.04.007

[8] Roger H. L. Chiang, Terence M. Barron, and Veda C. Storey. 1994. Reverse Engineering of Relational Databases: Extraction of an EER Model from a Relational Database. *Data Knowl. Eng.* 12, 2 (1994), 107–142. https://doi.org/10.1016/0169-023X(94)90011-6

[9] Benoit Combemale, Ralf Lämmel, and Eric Van Wyk. 2018. SLEBOK: The Software Language Engineering Body of Knowledge (Dagstuhl Seminar 17342). *Dagstuhl Reports* 7, 8 (2018), 45–54. https://doi.org/10.4230/DagRep.7.8.45

[10] DAMA International (Ed.). 2017. *Data Management Body of Knowledge (DMBOK), 2nd edition*. https://technicspub.com/dmbok/

[11] Jean-Marc Davril, Edouard Delfosse, Negar Hariri, Mathieu Acher, Jane Cleland-Huang, and Patrick Heymans. 2013. Feature model extraction from large collections of informal product descriptions. In *Joint Meeting European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, Bertrand Meyer, Luciano Baresi, and Mira Mezini (Eds.). ACM, 290–300. https://doi.org/10.1145/2491411.2491455

[12] EA community (Ed.). 2015. *Enterprise Architecture Body of Knowledge (EABOK)*. http://www.eabok.org/

[13] IIBA (Ed.). 2015. *Business Analysis Body of Knowledge (BABOK), version 3*. International Institute of Business Analysis (IIBA). http://www.iiba.org/babok-guide.aspx

[14] Vitaly Khusidman and William Ulrich. 2007. Architecture-Driven Modernization: Transforming the Enterprise. OMG Document admtf/07-12-01. https://www.omg.org/cgi-bin/doc?admtf/07-12-01.pdf http://www.omgwiki.org/admtf/doku.php.

[15] Levi Lúcio, Moussa Amrani, Juergen Dingel, Leen Lambers, Rick Salay, Gehan M. K. Selim, Eugene Syriani, and Manuel Wimmer. 2016. Model transformation intents and their properties. *Software and System Modeling* 15, 3 (2016), 647–684.

[16] Tom Mens and Pieter Van Gorp. 2006. A Taxonomy of Model Transformation. *Electr. Notes Theor. Comput. Sci.* 152 (2006), 125–142. https://doi.org/10.1016/j.entcs.2005.10.021

[17] Parastoo Mohagheghi, Vegard Dehlen, and Tor Neple. 2009. Definitions and Approaches to Model Quality in Model-based Software Development - A Review of Literature. *Inf. Softw. Technol.* 51, 12 (Dec. 2009), 1646–1669. https://doi.org/10.1016/j.infsof.2009.04.004

[18] Object Management Group. 2018. *OMG Software Modernization standards*. https://www.omg.org/spec/category/software-modernization/

[19] Gary R. Oliver. 2012. *Foundations of the Assumed Business Operations and Strategy Body of Knowledge (BOSBOK): An Outline of Shareable Knowledge*. Darlington Press.

[20] Project Management Institute (Ed.). 2017. *A guide to the Project Management Body of Knowledge (PMBOK guide), 6th edition*. https://www.pmi.org/pmbok-guide-standards/foundational/pmbok

[21] Harald Raffelt and Bernhard Steffen. 2006. LearnLib: A Library for Automata Learning and Experimentation. In *Fundamental Approaches to Software Engineering (FASE) (Lecture Notes in Computer Science)*, Luciano Baresi and Reiko Heckel (Eds.), Vol. 3922. Springer, 377–380. https://doi.org/10.1007/11693017_28

[22] Nicholas P. Sands and Ian Verhappen (Eds.). 2018. *Automation Body of Knowledge (ABOK), 3rd edition*. The International Society of Automation (ISA). https://www.isa.org/store/a-guide-to-the-automation-body-of-knowledge,-third-edition/60891879

[23] Bran Selic. 2003. The Pragmatics of Model-Driven Development. *IEEE Software* 20, 5 (2003), 19–25. https://doi.org/10.1109/MS.2003.1231146

[24] Mark Utting, Alexander Pretschner, and Bruno Legeard. 2012. A taxonomy of model-based testing approaches. *Softw. Test., Verif. Reliab.* 22, 5 (2012), 297–312. https://doi.org/10.1002/stvr.456

[25] Vadim Zaytsev (Ed.). 2017. *Software Language Engineering Body of Knowledge (SLEBOK)*. http://slebok.github.io/

ANNEX I: GLOSSARY OF TERMS

This glossary of terms contains the definitions of some of the main MBSE concepts, which could be useful for providing a consistent view of MBSE as a discipline, and as part of the foundations for curriculum development and individual certification of MBSE practitioners. In its current form, it mirrors the ongoing discussion and it does not claim to be complete. It is part of our future work to further develop it.

Model: A representation or specification of a system consisting of (a combination of) components, applications and actors and their interconnection, from a given point of view, and with a particular purpose.

Notes:

(1) In the SWEBOK, a model is defined as "an abstraction or simplification of a software component," which does not comprises all essential aspects of a software model.
(2) The purpose of models [23]
   - To understand the interesting characteristics of an existing or desired (complex) system and its environment
   - To predict the interesting characteristics of the system by analysing its model(s)
   - To communicate their understanding and design intent (to others and to oneself!)
   - To specify the implementation of the system (models as blueprints)
(3) Characteristics of useful Engineering Models [23]:
   - Purposeful: Constructed to address a specific set of concerns/audience
   - Abstract: Emphasize important aspects while removing irrelevant ones
   - Understandable: Expressed in a form that is readily understood by observers
   - Accurate: Faithfully represents the modeled system
   - Predictive: Can be used to answer questions about the modeled system
   - Cost effective: Should be much cheaper and faster to construct than actual system.

Domain Specific Model: A model written in a domain specific language.

Domain Specific Language: A language which offers concepts and notations closer to the domain experts, at an appropriate level of abstraction, and with a particular purpose.

Model Quality Terms [17]:

- A model is **comprehensible** if it is understandable by the intended users, being humans or tools.
- A model is **confined** if it suits to the modeling purpose and the type of system. This definition also includes relevant diagrams at the right abstraction level. A confined model does not have unnecessary information and is not more complex or detailed than necessary.
- A model is **changeable** if it can be evolved rapidly and continuously

Modeling Language:A specification of a family of models. A language definition can take more than one form; in the context of model-driven engineering, a very common form is through a metamodel, but another well-known form is a grammar. The family of (well-formed) models specified as a language is sometimes called the extension of the language.

Models@Run.time: A causally connected self-representation of the associated system that emphasizes the structure, behavior, or goals of the system from a problem space perspective. [5]

Model Transformation: (1) An algorithmic specification (declarative or operational) of the relationship between models. (2) A model transformation is the automatic generation of one or more target models from one or more source models, according to a transformation definition [16]. A model transformation can be defined by an algorithmic specification (declarative or operational) of the relationship between the source and target models.

Metamodel: A model that specifies the abstract syntax of a modeling language, including the language concepts, the relationships and constraints among them, and the well-formedness rules of the language.

Megamodel: A model whose elements are modeling languages, models, metamodels, transformations, etc. [4]

Multi-view Modeling: An approach to modeling systems using a set of well defined viewpoints, each one expressing a different concern of the same system, and related by a set of viewpoint correspondences.

Notes:

(1) This approach enables separation of concerns, since each viewpoint focuses on one particular aspect of the system
(2) It is also referred to as Multi-paradigm modeling, when the languages used in each viewpoint are of different nature (e.g., discrete and continuous, operational and functional)

Viewpoint (on a system): A form of abstraction achieved using a selected set of architectural concepts and structuring rules, in order to focus on particular concerns within a system. Normally a viewpoint is defined by a metamodel, which defines the concepts of interest to the viewpoint, their relationships, and their integrity constraints.

Viewpoint correspondence: A statement that some elements in the specification of one viewpoint are related to (e.g., describe the same entities as) elements in a specification of a second viewpoint.

View: A model of a system from a particular viewpoint (i.e., the model conforms to the metamodel defined for that viewpoint).