Software & Systems Modeling (2019) 18:3193–3205
https://doi.org/10.1007/s10270-019-00746-9

EXPERT VOICE

Contents for a Model-Based Software Engineering Body of Knowledge

Loli Burgueño[1,2] · Federico Ciccozzi[3] · Michalis Famelis[4] · Gerti Kappel[5] · Leen Lambers[6] · Sebastien Mosser[7] · Richard F. Paige[8] · Alfonso Pierantonio[9] · Arend Rensink[10] · Rick Salay[11] · Gabriele Taentzer[12] · Antonio Vallecillo[13] · Manuel Wimmer[14]

Received: 23 May 2019 / Revised: 29 June 2019 / Accepted: 2 July 2019 / Published online: 18 July 2019

Abstract
Although *Model-Based Software Engineering* (MBE) is a widely accepted *Software Engineering* (SE) discipline, no agreed-upon core set of concepts and practices (i.e., a Body of Knowledge) has been defined for it yet. With the goals of characterizing the contents of the MBE discipline, promoting a global consistent view of it, clarifying its scope with regard to other SE disciplines, and defining a foundation for the development of educational curricula on MBE, this paper proposes the contents for a Body of Knowledge for MBE. We also describe the methodology that we have used to come up with the proposed list of contents, as well as the results of a survey study that we conducted to sound out the opinion of the community on the importance of the proposed topics and their level of coverage in the existing SE curricula.

Keywords Model-Based Software Engineering · Body of Knowledge · Core concepts · Education

1 Introduction

Model-Based Software Engineering (MBE) is a widely accepted *Software Engineering* (SE) discipline that promotes

Communicated by Bernhard Rumpe.

✉ Federico Ciccozzi
federico.ciccozzi@mdh.se

Loli Burgueño
lburguenoc@uoc.edu

Michalis Famelis
famelis@iro.umontreal.ca

Gerti Kappel
gerti@big.tuwien.ac.at

Leen Lambers
Leen.Lambers@hpi.de

Sebastien Mosser
mosser.sebastien@uqam.ca

Richard F. Paige
paigeri@mcmaster.ca

Alfonso Pierantonio
alfonso.pierantonio@univaq.it

Arend Rensink
arend.rensink@utwente.nl

Rick Salay
rsalay@cs.toronto.edu

Gabriele Taentzer
taentzer@informatik.uni-marburg.de

Antonio Vallecillo
av@lcc.uma.es

Manuel Wimmer
manuel.wimmer@jku.at

1 UOC, Barcelona, Spain
2 CEA LIST, Paris, France
3 Mälardalen University, Västerås, Sweden
4 Université de Montréal, Montreal, Canada
5 CDP, TU Wien, Vienna, Austria
6 Hasso-Plattner-Institut, Potsdam, Germany
7 UQAM, Montreal, Canada
8 McMaster University, Hamilton, Canada
9 University of L'Aquila, L'Aquila, Italy
10 University of Twente, Enschede, The Netherlands
11 University of Toronto, Toronto, Canada
12 Philipps-University Marburg, Marburg, Germany
13 Universidad de Málaga, Málaga, Spain
14 JKU Linz, Linz, Austria

the use of models and model transformations as the fundamental elements of software development [5]. With almost 20 years of existence [3,18], MBE is already part of most SE curricula, and the industrial use of its concepts and practices keeps growing.[1]

Despite its expanding adoption in industry and academia, no widely agreed-upon core set of concepts, elements and practices encompassed by MBE has been described anywhere yet. Such a compendium corresponds to the notion of a *"Body of Knowledge"* (BoK) used in many engineering disciplines, which comprises the set of concepts, terms, and activities that make up a professional domain, being a fundamental part of any profession [15].

In 2018, we started working on the basic contents of a BoK for MBE (hereinafter, MBEBOK), with the goals of characterizing the contents of the MBE discipline, promoting a consistent view of it worldwide, clarifying its scope with regard to other SE disciplines, and defining a foundation for the development of MBE curricula. As part of this effort, a set of topics was identified and presented at the MODELS 2018 Educators' Symposium [7], where we received useful feedback not only about the topics themselves, but also on how to potentially implement the MBEBOK.

After the symposium, we decided to validate the revised list of topics with the MBE community by means of conducting a survey to obtain a clear picture about:

(a) the importance that the community gives to each topic in education and certification of model-based software engineers;
(b) the coverage of these topics in current SE/MBE curricula, at both Bachelor and Master levels.

The response from the community was significant (101 answers), and the results of the survey provide an interesting snapshot of the current situation. This paper presents these results and formulates a proposal of the topics that should form an MBEBOK. The long-term goal for the MBEBOK is to significantly help consolidate the field of MBE as well as to improve the way it is currently taught and practiced. Our proposal also aims at providing the basis upon which an extension of the *Guide to the Software Engineering BoK* (SWEBOK) can be developed to entail all aspects related to MBE.

The paper is structured in Sect. 5. After this introduction, Sect. 2 discusses what a Body of Knowledge is and briefly presents some of them related to Software, in particular the SWEBOK. Then, Sect. 3 identifies the basic topics that should form part of contents of the MBEBOK, as well as the methodology we have followed to develop it. Finally, Sect. 4 discusses some issues related to the proposal, and Sect. 5 concludes the paper and outlines the next steps.

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 and global view of the discipline;
- 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, more specifically:

- A terminology that defines the set of main concepts of the discipline, as used by their practitioners; this 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 possess.

A BoK should always be descriptive, but not prescriptive: Intentionally, it should not impose any particular method or tool, nor specific practices.

With respect to what should be considered as "generally recognized" or as a "good practice" of a discipline, the *Project Management BoK* (PMBOK) [16] offers precise definitions. First, "generally recognized" means that the knowledge and practices described are generally applicable to multiple kinds of diversified projects in various situations, and there is consensus about their value and usefulness. In turn, "good practice" means that there is general agreement that the application of skills, tools, and techniques can enhance the chances of success over a wide range of projects.

[1] Note that although an apparently more natural acronym for Model-Based Software Engineering would have been "MBSE," we opted instead for MBE to mitigate potential misunderstandings with other initiatives, especially from industrial research and practice. A glaring example is the International Council on Systems Engineering (INCOSE) initiative, which has defined "MBSE" as the standard acronym for Model-Based Systems Engineering widely used in industry [11], as also reported by the Systems Engineering Body of Knowledge (SEBOK) [1].

It does not mean, however, that the precise knowledge or practice should always be applied to any project; the organization and/or project management team should ultimately be 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), while others are evolving (e.g., SLEBOK):

- Software Engineering BoK (SWEBOK) [4]
- Enterprise Architecture BoK (EABOK) [10]
- Business Analysis BoK (BABOK) [12]
- Systems Engineering BoK (SEBOK) [1]
- Data Management BoK (DMBOK) [9]
- Project Management BoK (PMBOK) [16]
- Automation BoK (ABOK) [17]
- Software Language Engineering BoK (SLEBOK) [19]

2.2 The Software Engineering BoK (SWEBOK)

In 2004, the IEEE Computer Society established for the first time a baseline for the BoK of the field of software engineering. It was the outcome of a joint effort with ACM, whose mission was "to establish the appropriate set(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 global 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 in 2014, using the same editing process, giving birth in 2014 to the current version (v3) of the SWEBOK [4]. The SWEBOK has been adopted by ISO and IEC as ISO/IEC TR 19759:2005.[2]

It should be noted that the SWEBOK does not aim at describing the entire BoK for software engineering. Instead, it provides a "guide" to the existing BoK that has been developed since the start of the discipline—generally agreed to have happened in 1964, in the NATO conference held in Germany to discuss, for the first time, the software crisis.

[2] https://www.iso.org/standard/33897.html.

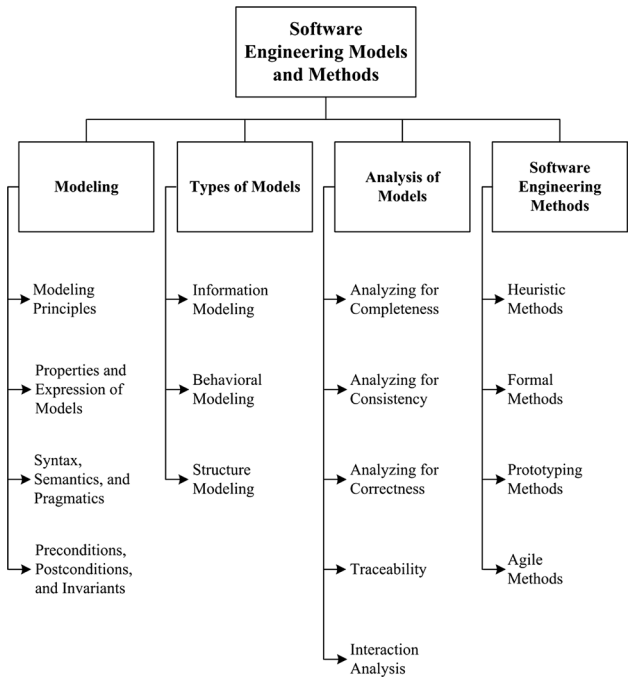


Fig. 1 Breakdown of Topics for the Software Engineering Models and Methods KA in SWEBOK (from [4])

SWEBOK v3.0 defines 15 *Knowledge Areas* (KA), plus Appendix that lists the International Standards supporting the SWEBOK.

One complete KA of the SWEBOK is devoted to *Software Engineering Models and Methods* (Chapter 9). 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. The use of 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):

- **Modeling**: It discusses the general practice of modeling, and presents:
  - The basic modeling concepts and principles
  - Properties (such as completeness, consistency, and correctness) and expression of models (as typed and attributed elements 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**: It 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 (a.k.a. conceptual models)
  - Behavior models (state machines, control-flow models, dataflow models)
  - Structure models (e.g., UML class, component, object, deployment, and packaging diagrams)
- **Analysis of models**: It presents some of the common analysis techniques used in modeling to verify completeness, consistency, correctness, traceability, and interaction analysis.
- **Software Engineering Methods**: It 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 aims to apply to any SE discipline, not only to MBE.

In addition to these topics, the SWEBOK lists the International Standards related to "Software Engineering Models and Methods" in its Annex B. There are three groups of standards, depending on their scope: *modeling notations* (such as IDEF, UML, OCL, or KDM); *tools* (IEEE Stds. 14102, 14471, and 1175, which apply to all CASE tools and their interoperability); and the *environments* of the systems (such as ISO/IEC 26515, on developing information in agile environments, and 15940 on Software Engineering environment services). One additional standard about terminology (ISO/IEC 24765) is common to all Knowledge Areas.

By looking at the concepts and related standards, we see that the coverage of MBE concepts and mechanisms is generally appropriate, but some essential concepts of MBE that should be part of the education of MBE practitioners—such as model transformations, executable models, or code generation, for instance—have not been included. Furthermore, although the SWEBOK provides definitions for some MBE concepts, not all of them are precisely defined, and in some cases the given definitions miss 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 MBE terms is another goal of the MBEBOK.

2.3 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., languages that may ultimately be implemented on a machine. SLEBOK is an ongoing initiative to capture a BoK for SLE. A Dagstuhl seminar [8] was held to capture a preliminary set of artifacts, definitions, methods, best practices, open challenges, etc. The intent was for these to be consolidated into the SLEBOK, which is currently evolving. While 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 the fact that it is very open, and anyone can contribute to the revision process via its Git repository [19].

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. Although these relationships are still evolving, due to the relative immaturity of both BoKs, 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 [7].
- SLEBOK does mention the notion of static semantics, but does not elaborate on language semantics, whereas MBEBOK explicitly captures model semantics as a key concept.
- SLEBOK and MBEBOK treat abstract and concrete syntax differently; while these are first-class concepts in MBEBOK, their notions are distributed across multiple concepts in SLEBOK.

3 Topics for an MBEBOK

3.1 Developing the proposed list of topics

The list of topics proposed for the contents of the MBEBOK is shown in Figs. 2 and 3. This list was produced in two stages. An initial proposal was presented at the MODELS Educators' Symposium in October 2018 and discussed during the event with the audience. A poster with the proposal was also prepared and displayed during the conference, and the participants were asked to comment on its contents. Apart from the suggestions made directly to the poster presenters, sticky notes and pens were available for attendees to annotate the poster with comments at any time during the three days of the conference. A shared document was also available online after the conference so that any interested person could comment on it. We received numerous interesting and valuable comments and suggestions and refined the list of topics in November 2018.

The next step was to reach the wider modeling audience to get a feedback from the interested community. For this,

	Importance		BSc Coverage		MSc Coverage		Importance minus:		
	Avg	Std.dev	Avg	Std.dev	Avg	Std.dev	BSc cov	MSc cov	
1. Model Foundations									
1.1. Syntax	3.73	*0.61*	2.53	*0.94*	3.04	*1.04*	1.21	0.69	*Basic*
* Abstract syntax. Metamodels	3.71	*0.63*	2.20	*1.00*	3.02	*1.07*	1.51	0.69	*Basic*
* Concrete syntax	3.57	*0.69*	2.56	*0.96*	2.93	*1.07*	1.02	0.65	*Basic*
1.2. Semantics	3.72	*0.60*	2.26	*0.91*	2.82	*1.07*	1.46	0.90	*Basic*
* Structural	3.66	*0.69*	2.33	*0.93*	2.77	*1.10*	1.34	0.89	*Basic*
* Behavioral (discrete vs. continuous)	3.54	*0.76*	2.05	*0.86*	2.48	*1.13*	1.49	1.06	*Basic*
1.3. Purpose/intent	3.73	*0.54*	2.28	*0.92*	2.87	*1.04*	1.45	0.86	*Basic*
* Modeling principles	3.76	*0.55*	2.37	*0.96*	2.88	*1.03*	1.39	0.88	*Basic*
* Exemplar purposes: such as Metamodeling or model transformation definition	3.41	*0.79*	1.85	*0.93*	2.70	*1.06*	1.55	0.70	*Intermediate*
2. Model Quality									
2.1. Completeness	3.24	*0.85*	2.05	*0.92*	2.60	*1.07*	1.19	0.64	*Intermediate*
2.2. Consistency	3.77	*0.55*	2.19	*0.91*	2.82	*1.12*	1.58	0.95	*Basic*
2.3. Correctness	3.64	*0.59*	2.30	*0.97*	2.74	*1.13*	1.35	0.90	*Basic*
2.4. Comprehensibility	3.66	*0.57*	2.15	*0.98*	2.56	*1.09*	1.52	1.10	*Basic*
2.5. Confinement (= fitness for purpose)	3.57	*0.64*	1.98	*0.83*	2.48	*1.07*	1.59	1.09	*Basic*
2.6. Changeability	3.22	*0.83*	1.65	*0.77*	2.26	*1.07*	1.57	0.96	*Intermediate*
3. Analysis									
3.1. Structural model analysis	3.46	*0.71*	1.94	*0.93*	2.41	*1.04*	1.52	1.05	*Basic*
* Invariant checking	3.24	*0.89*	1.83	*0.93*	2.42	*1.09*	1.42	0.82	*Intermediate*
* Instance generation	3.22	*0.92*	1.72	*0.90*	2.22	*1.10*	1.51	1.00	*Intermediate*
* Metrics calculation	2.90	*0.83*	1.50	*0.71*	2.03	*1.00*	1.40	0.87	*Advanced*
* Smells detection	2.91	*0.89*	1.49	*0.74*	1.89	*0.98*	1.41	1.02	*Advanced*
3.2. Behavioral model analysis	3.37	*0.77*	1.69	*0.75*	2.22	*0.98*	1.68	1.15	*Intermediate*
* Pre-postcondition checking	3.28	*0.87*	1.68	*0.81*	2.21	*1.06*	1.60	1.07	*Intermediate*
* Simulation	3.29	*0.82*	1.57	*0.74*	2.13	*1.05*	1.73	1.16	*Intermediate*
* Reachability analysis	2.96	*0.86*	1.43	*0.71*	1.96	*1.05*	1.52	1.00	*Advanced*
** Temporal model checking*	*2.77*	*0.97*	*1.35*	*0.58*	*2.07*	*1.08*	*1.42*	*0.70*	--
* Performance	2.96	*0.95*	1.38	*0.61*	1.97	*1.03*	1.58	0.99	*Advanced*
3.3. Model transformation analysis	3.24	*0.94*	1.41	*0.71*	2.15	*1.05*	1.83	1.09	*Intermediate*
* MT Correctness (of transformed models, in syntax and semantics)	3.30	*0.88*	1.35	*0.67*	2.10	*1.02*	1.95	1.20	*Intermediate*
* MT Completeness	3.00	*0.94*	1.32	*0.61*	1.98	*1.09*	1.68	1.02	*Intermediate*
* MT Functional behavior (termination, confluence)	2.93	*0.90*	1.30	*0.58*	1.99	*1.04*	1.63	0.94	*Advanced*
** MT Performance*	*2.78*	*0.93*	*1.30*	*0.58*	*1.86*	*1.05*	*1.49*	*0.93*	--
4. Modeling Languages									
4.1. Language definition	3.68	*0.60*	2.27	*0.91*	2.94	*1.10*	1.42	0.75	*Basic*
* Metamodels	3.63	*0.73*	2.03	*1.02*	2.99	*1.14*	1.60	0.64	*Basic*
* Grammars	3.49	*0.74*	2.25	*1.03*	2.73	*1.16*	1.24	0.76	*Basic*
* Semantics (by e.g. Abstract State Machines or model transformations)	3.35	*0.81*	1.99	*0.84*	2.70	*1.11*	1.36	0.65	*Intermediate*
4.2. Types of modeling languages	3.67	*0.58*	2.40	*0.89*	2.89	*1.07*	1.28	0.78	*Basic*
* General purpose (GPL): UML+OCL, SysML	3.67	*0.63*	2.70	*0.89*	2.94	*1.03*	0.97	0.74	*Basic*
* Domain-specific (DSL): UML Profiles, Language Workbenches, ADLs, ...	3.57	*0.65*	1.76	*0.90*	2.80	*1.10*	1.81	0.77	*Basic*
4.3. Multiview Modeling	3.22	*0.79*	1.62	*0.77*	2.22	*1.14*	1.60	1.01	*Intermediate*
* Model viewpoints and views	3.24	*0.79*	1.65	*0.78*	2.20	*1.13*	1.59	1.04	*Intermediate*
* Correspondences among views	3.17	*0.85*	1.52	*0.76*	2.10	*1.10*	1.65	1.07	*Intermediate*
* Viewpoint consistency	3.14	*0.84*	1.47	*0.76*	2.01	*1.10*	1.67	1.13	*Intermediate*
* Viewpoint integration	3.11	*0.85*	1.41	*0.71*	1.97	*1.08*	1.70	1.15	*Intermediate*

Fig. 2 List of proposed topics for the MBEBOK and results from the survey (Sections 1–4)

we decided to conduct a survey study. A questionnaire was developed to sound out the opinion of the community about three main aspects: the topics included in the list, the importance assigned to each of them, and the coverage of the topics in the courses taught at the respondents' institutions, at both Bachelor and Master levels. The outcomes are summarized in the following sections and depicted in Figs. 2, 3, and 4.

3.2 List of topics

The proposed list contains the topics that were considered to be relevant for the MBEBOK, i.e., that should be part of the knowledge that any MBE practitioner should possess. The list proposed in the survey, which is shown in the first column of Figs. 2 and 3, is structured into the following nine sections.

	Importance		BSc Coverage		MSc Coverage		Importance minus:		
	Avg	Std.dev	Avg	Std.dev	Avg	Std.dev	BSc cov	MSc cov	
5. Model Representation									
5.1. Concrete syntax	3.53	*0.73*	2.49	*1.06*	2.87	*1.05*	1.04	0.66	*Basic*
5.2 Physics of notations	2.80	*1.06*	1.56	*0.92*	1.97	*1.07*	1.24	0.84	--
5.3. Layout	2.99	*0.94*	1.63	*0.83*	2.10	*1.05*	1.36	0.89	*Advanced*
5.4. Textual vs visual models	3.30	*0.81*	1.82	*0.86*	2.49	*1.11*	1.48	0.81	*Intermediate*
5.5. Animation	2.46	*0.96*	1.33	*0.67*	1.72	*1.01*	1.12	0.74	--
6. Model Maintenance and Evolution									
6.1. Model operations (diff, merge, refactoring)	3.40	*0.80*	1.35	*0.73*	1.95	*1.08*	2.04	1.45	*Intermediate*
6.2. Model Versioning	3.45	*0.76*	1.31	*0.64*	1.86	*1.04*	2.13	1.59	*Intermediate*
6.3. Model Migration	3.22	*0.85*	1.22	*0.50*	1.75	*0.97*	2.00	1.47	*Intermediate*
7. Model Execution									--
7.1. Model simulation	3.30	*0.85*	1.49	*0.78*	1.95	*1.01*	1.81	1.36	*Intermediate*
* Model co-simulation (simulation of a hybrid model)	2.91	*0.95*	1.25	*0.62*	1.63	*0.95*	1.66	1.27	*Advanced*
7.2. Execution strategies (sequential vs. parallel)	2.84	*0.95*	1.31	*0.64*	1.72	*1.00*	1.52	1.11	*Advanced*
7.3. Model debugging and testing	3.37	*0.82*	1.41	*0.78*	1.87	*1.01*	1.97	1.51	*Intermediate*
8. Model Transformations									
8.1. Model transformation languages	3.46	*0.84*	1.52	*0.79*	2.56	*1.07*	1.93	0.90	*Basic*
* Syntax	3.33	*0.93*	1.50	*0.80*	2.59	*1.10*	1.83	0.74	*Intermediate*
* Semantics	3.43	*0.85*	1.39	*0.74*	2.48	*1.11*	2.03	0.94	*Intermediate*
8.2. Model transformation types	3.39	*0.86*	1.47	*0.74*	2.47	*1.09*	1.92	0.91	*Intermediate*
* Text-to-Model, M2M, M2T	3.47	*0.83*	1.53	*0.82*	2.60	*1.12*	1.94	0.87	*Basic*
* Exogenous vs. endogenous	3.24	*0.91*	1.29	*0.64*	2.28	*1.08*	1.95	0.95	*Intermediate*
* In-place vs. out-place	3.14	*0.97*	1.31	*0.72*	2.24	*1.11*	1.83	0.90	*Intermediate*
* Horizontal vs. vertical	3.12	*0.99*	1.29	*0.67*	2.11	*1.09*	1.82	1.00	*Intermediate*
* Uni-directional vs. bidirectional	3.20	*0.91*	1.25	*0.60*	2.06	*1.08*	1.95	1.14	*Intermediate*
* Syntactical vs. semantic	3.09	*0.99*	1.25	*0.60*	1.93	*1.04*	1.84	1.16	*Intermediate*
* Paradigm (declarative vs. operational)	3.22	*0.88*	1.30	*0.67*	2.14	*1.13*	1.92	1.08	*Intermediate*
8.3. Model transformation applications	3.49	*0.77*	1.45	*0.71*	2.41	*1.08*	2.04	1.08	*Basic*
* Model translation (synthesis, code generation, reverse engineering, migration, optimization, refactoring, refinement, adaptation) [10]	3.60	*0.73*	1.57	*0.80*	2.50	*1.10*	2.03	1.10	*Basic*
* Model merge	3.15	*0.89*	1.23	*0.55*	1.85	*0.99*	1.92	1.30	*Intermediate*
* Model differencing	3.15	*0.93*	1.26	*0.57*	1.82	*1.03*	1.89	1.33	*Intermediate*
* Model weaving	2.97	*0.99*	1.20	*0.52*	1.71	*0.96*	1.77	1.26	*Advanced*
* Model synchronization	3.07	*0.92*	1.20	*0.52*	1.72	*0.87*	1.87	1.35	*Intermediate*
* Model interpretation (incl. execution)	3.24	*0.88*	1.30	*0.66*	2.02	*1.12*	1.94	1.22	*Intermediate*
9. Further topics									
9.1. How MBSE is used in application domains	3.58	*0.68*	1.58	*0.78*	2.10	*1.04*	2.00	1.48	*Basic*
* Automotive	3.28	*0.83*	1.42	*0.71*	1.87	*0.96*	1.86	1.41	*Intermediate*
* Cyber physical systems	3.31	*0.79*	1.43	*0.76*	1.93	*1.02*	1.87	1.37	*Intermediate*
* Industry 4.0	3.28	*0.83*	1.32	*0.67*	1.76	*0.94*	1.96	1.52	*Intermediate*
* Banking systems (e.g. modernization)	2.96	*0.94*	1.34	*0.70*	1.68	*0.91*	1.62	1.27	*Advanced*
9.2. Advanced topics	2.69	*0.92*	1.22	*0.55*	1.57	*0.89*	1.47	1.12	--
* *Streaming model transformations*	2.42	*1.03*	1.14	*0.46*	1.38	*0.77*	1.28	1.05	--
* *Incremental transformations*	2.60	*0.99*	1.16	*0.52*	1.54	*0.87*	1.44	1.06	--
* *Uncertainty in modeling*	2.73	*1.01*	1.15	*0.47*	1.49	*0.89*	1.58	1.24	--
9.3. Application scenarios of MBSE	3.30	*0.78*	1.44	*0.80*	1.98	*1.02*	1.86	1.32	*Intermediate*
* Model based testing	3.28	*0.77*	1.46	*0.76*	2.03	*1.05*	1.81	1.24	*Intermediate*
* Model-based modernization	2.99	*0.87*	1.22	*0.59*	1.75	*0.95*	1.77	1.24	*Advanced*
9.4. Engineering Best Practices	3.52	*0.73*	1.72	*0.93*	2.22	*1.12*	1.80	1.30	*Basic*
* How to use models to represent information systems	3.56	*0.71*	1.82	*0.99*	2.28	*1.16*	1.74	1.28	*Basic*
* How to use models to represent physical systems	3.40	*0.79*	1.52	*0.86*	1.98	*1.05*	1.88	1.42	*Intermediate*

Fig. 3 List of proposed topics for the MBEBOK and results from the survey (Sections 5–9)

1. **Model Foundations**: It covers the basic modeling concepts and practices, including *syntax* (e.g., abstract vs. concrete), *semantics* (structural, behavioral, informational), and *purposes/intents* of models (namely modeling principles and exemplar purposes such as metamodeling or model transformation definition).
2. **Model Quality**: This section deals with quality aspects of models, including *completeness*, *consistency*, *correctness*, *comprehensibility*, *confinement* (i.e., fitness for purpose) and *changeability*.
3. **Analysis**: This section is organized in three main subsections: *structural model analysis* (consistency checking, instance generation, metrics and bad smell detection),

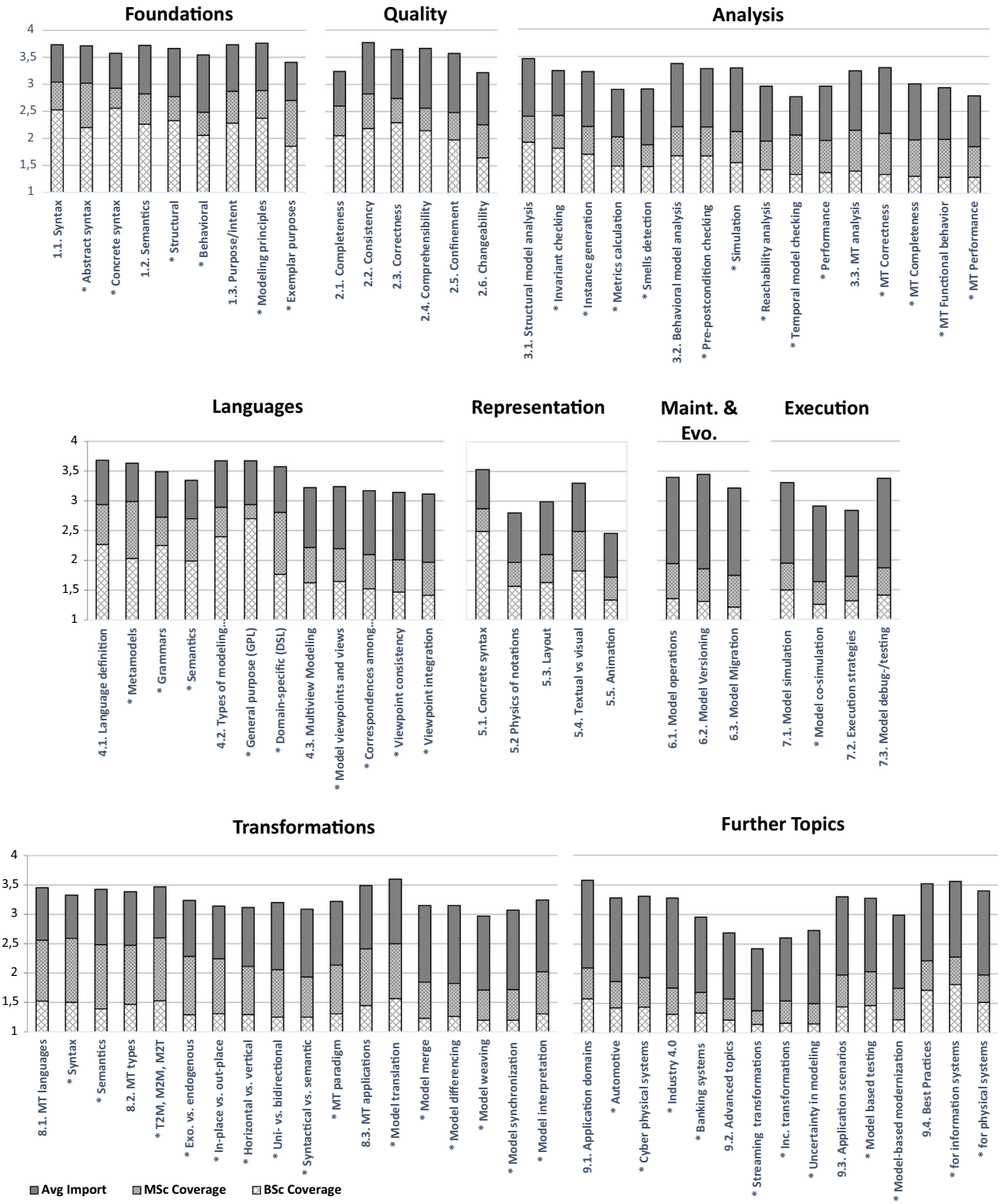


Fig. 4 Chart with the results of the survey

behavioral model analysis (pre-/postcondition checking, simulation, performance analysis, reachability analysis, temporal model checking), and *model transformation analysis* (correctness, termination, etc.).

4. **Modeling Languages**: This section deals with *language definition* (metamodels, grammars, semantics), *types of modeling languages* (general purpose, domain-specific), and *multiview modeling* (model viewpoints and views, correspondences among views, viewpoint consistency, and viewpoint integration).
5. The **Model Representation** section covers *concrete syntax*, the *physics of notations*, *layouts*, the dichotomy between *textual and visual* models, as well as *animation*.
6. **Model Maintenance and Evolution** is concerned with *model operations* (diff, merge, refactoring), model *versioning*, and model *migration*.
7. **Model Execution** deals with *model simulation* and co-simulation, *execution strategies* (sequential vs. parallel), and model *debugging and testing*.
8. **Model Transformations** are covered in this section. It includes *model transformation languages* (syntax and semantics), *model transformation types* [14] (text-to-model, model-to-model, model-to-text, exogenous vs. endogenous, in-place vs. out-place, horizontal vs. vertical, unidirectional vs. bidirectional, etc.), and *model transformation applications*, such as model translation (synthesis, code generation, reverse engineering, migration, optimization, refactoring, refinement, adaptation) [13], model merge, differencing, weaving, synchronization, etc.
9. **Further topics** includes how MBE is used in *application domains* (such as Automotive, Cyber physical systems, Industry 4.0, Banking systems (e.g., modernization), etc.), *advanced topics* (streaming model transformations, incremental transformations, uncertainty in modeling), some *application scenarios of MBE* (model-based testing and model-based modernization) and some *engineering best practices*, namely how to use models to represent information applications or physical systems.

3.3 The survey

The survey[3] was conducted in December 2018. An invitation to participate was sent to all major Software Engineering and Modeling distribution lists, with a questionnaire where participants were asked to grade their perceived importance for each topic and whether the topic was covered in any of the courses taught at their institution. The detailed instructions given to the participants were the following:

"By assigning importance to each topic, you are defining the minimum set of concepts that an average MBS engineer should know, according to your own view of what an 'average' model-based software engineer is. In the questions below, 'Not important at all' *means that this topic should not be part of the basic curriculum.* "Covered" *means that the topic is already covered in any of the courses taught at your institution, at Bachelor or Master level, or both. Respond indicating the level of coverage of the topic in the course(s)."*

A 4-point Likert scale was used to record the importance assigned to each topic: (1) "Not at all important," (2) "Slightly important," (3) "Moderately important," and (4) "Important." An additional option "No opinion" was also included. Similarly, coverage was captured by a 4-value Likert scale: (1) "No,' (2) "Slightly," (3) "Enough," and (4) "Well," with an additional value "Do not know" to improve the reliability of the responses.

A total of 101 responses were recorded, from 23 different countries, distributed across continents as follows: Europe 85%, America 11%, Oceania 2%, Asia 1%, and Africa 1%. The results of the survey, including the raw responses from the participants, are available at http://bit.ly/MBEBOK-2018SurveyResults.

Figures 2 and 3 display the results in tabular form. Each row represents a specific topic, with the average score and standard deviation of the assigned importance and coverage at BSc and MSc levels. The last two columns show the difference between the importance assigned to a topic and how it is covered in the curricula of the institution, in order to identify possible decompensations. Cells shaded in green color highlight the highest scores, while red-shaded cells identify the lowest. The highest standard deviations are highlighted in yellow; they identify the topics with less consensus.

For clarity, Fig. 4 shows the results of the survey graphically. The X-axis lists the topics (they are the same as those listed in Figs. 2 and 3, respectively), while the Y-axis plots the resulting scores of the answers: topic importance and topic coverage at MSc and BSc levels.

3.4 Main findings

The results of the survey provided a very interesting feedback about our proposed list of topics.

(a) *Topics with insufficient support.* Some of the proposed topics received little support:

- Topics marked as **Not important at all** (a significant number of respondents indicated that the topic should not be included in the MBEBOK):
  - More than 15%: Animation (5.5), Streaming Transformations (9.2.1), Incremental Transformations (9.2.2)
  - Between 10 and 15%: Physics of notations (5.2), Further topics (9.2), and Uncertainty in Modeling (9.2.3)

[3] https://encuestas.uma.es/27511/lang-en.

- **Low score in Importance** (this indicates lack of support):
  - Score below 2.5: Animation (5.5), Streaming Transformations (9.2.1)
  - Score below 2.75: Further topics (9.2), Incremental Transformations (9.2.2), Uncertainty (9.2.3)
- **No Opinion** (this normally indicates lack of sufficient knowledge about the topic):
  - Between 10% and 15%: Streaming Transformations (9.2.1);
  - More than 15%: Physics of notations (5.2)

(b) *Topics with low coverage.* Interestingly, there is a clear correlation between importance and coverage: In general, more important topics show higher coverage in existing curricula (the Pearson coefficient for the correlation between the assigned importance and the coverage at BSc level is 0.73 and 0.82 for the coverage at MSc level). There are significant exceptions, such as *model maintenance* and *execution*; despite being considered fairly important, their coverage is rather low.
(c) *Relative importance of subtopics.* There are some topics—namely 3.1 (*structural model analysis*), 8.1 (*model transformation languages*), and 9.1 (*MBE application domains*)—classified as `Basic` due to their importance, but the selected subtopics are, however, classified as `Intermediate` or even `Advanced`. This means that the topic as a whole is considered essential, but the subtopics themselves are slightly less relevant.
(d) *Teaching metamodels and grammars.* There seems to be a discrepancy in the community about when to teach metamodels and grammars, whether at BSc or MSc levels. This issue would require further analyses.

3.5 Proposal

After analyzing the results of the survey, our proposal is summarized in the last column of the tables shown in Figs. 2 and 3, where we have classified each topic as: `Basic`, `Intermediate`, or `Advanced`. Topics marked with two hyphens (`--`) received little support, and hence, they have been excluded from the final proposal.

Topics were classified as `Basic` if their average *importance* score was above 3.45, and `Intermediate` if importance was above 3.0, which represent, respectively, an average of 86% and 66% of the max score in the Likert scale [1–4]. `Advanced` topics were those that scored at least 2.8 in importance (60%). We decided to discard those topics that did not reach a score of 2.8 in importance or did not get sufficient support, as discussed above. This classification is also consistent with the expected coverage of each topic at BSc and MSc levels. `Basic` topics should be taught at BSc level. `Basic` and `Intermediate` topics should be covered by all BSc and MSc courses on MBE. This approach would provide the required homogeneity and interoperability between the foundational contents of separate MBE curricula.

`Advanced` topics could be either covered at MSc level or as part of specialized courses, depending on the educational institution. In this way, each institution could define specialized offerings depending on the knowledge and experience of its research groups. However, they all will be able to share the same core concepts, use the same terminology, and have a common background in the basic MBE courses.

Of course, additional MBE topics (e.g., those not included in the list) could also be taught depending on the experience, industrial demands, and specialized offerings of each institution.

4 Discussion

4.1 Integration with the SWEBOK

As mentioned earlier, the SWEBOK already provides good coverage of some of the concepts and practices of MBE. However, it misses out some key elements that should be part of the knowledge and skills of any model-based software engineer.

At the beginning of this effort, we considered several options to integrate our extension with the contents of the SWEBOK, from more conservative to more disruptive: (a) adding paragraphs to the existing text of Chapter 9 and new subsections if needed, but respecting its current structure; (b) replacing the contents of the entire Chapter 9; and (c) creating a complete BoK, separating from SWEBOK (this is the strategy that other BoKs, e.g., SLEBOK, adopted).

Based on the suggestions received at EduSymp'18, our proposal is to follow the same strategy as other extensions to BoKs, which are delivered as Appendices to them. Such a strategy is not intrusive nor disruptive and enables both the SWEBOK and the MBE-annex to separately evolve, but still maintaining consistency and minimizing redundancy.

4.2 Integration with the SLEBOK

The design, development, and maintenance of modeling languages could be considered as very important for any MBE practitioner. In fact, a number of topics listed in Sect. 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 MBE topics adequately. For example, 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 that they are treated consistently in both guides. This is not a trivial issue, given the current status of the SLEBOK. However, the fact that it 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 permits complementing the contents of both BoKs in a successful manner.

4.3 Relationships with other BoKs

Although the closest connections of the MBEBOK are with the SWEBOK and the SLEBOK, overlaps with the other BoKs listed in Sect. 2.1 also exist. The identification and analysis of common and related topics with other BoKs is essential in order to maintain consistency and interoperability with the rest of the engineering disciplines.

4.4 Field studies and other related initiatives

In parallel with the development of the basic contents for the MBEBOK, a number of ongoing initiatives are surveying different aspects of how models are currently taught, for example, the experiences of instructors when teaching modeling and MBE topics [6] or the experiences of students with software modeling tools [2]. These initiatives nicely complement our present proposal and could be of significant value for the development of the Guide to the MBEBOK, e.g., to define the MBE skills and practices required to use the concepts identified in this work, and the kinds of tools needed by MBE practitioners and students.

5 Conclusions and next steps

This paper has presented a proposal for the list of topics that should be covered by the MBEBOK. As such, it aims to characterize the contents and known practices of the MBE discipline, assist universities and other institutions that provide teaching courses on SE to develop their MBE curricula, and identify the core set of concepts that any MBE engineer should know, providing a common and consistent reference terminology. We are convinced that this will significantly help to consolidate the field of MBE, clarify its scope with respect to other SE disciplines, and to improve the way it is currently taught.

Based on this proposal, we now plan to start working on "the Guide to the MBE Body of Knowledge," which further develops these concepts and the practices that support the discipline. Such a document is intended to supplement the contents of the SWEBOK and to be aligned with the current SLEBOK efforts and with the rest of the engineering BoKs.

Our effort will continue in two main directions. First, part of us will be in charge of developing the "Guide to the MBEBOK" as an Annex to the SWEBOK. A smaller group of experts will work on the glossary of terms for the MBEBOK, too. Once initial versions of the Guide and of the Glossary will be ready, we will let them circulate it to a broader audience so that the SE, MBE, and SLE communities have the opportunity to revise them and provide suggestions. We envision the first version of the MBEBOK as an open and collaborative platform (e.g., GitBook) that will be used to (i) disseminate the MBEBOK while writing it and (ii) continue to collect feedback from the community.

In this respect, we would like to use the opportunity to invite educators and researchers of the SoSyM community willing to contribute to this effort. Volunteers ready to participate are invited to write to us, indicating how they want and can help, to the following mail address: TheMBEBOK@gmail.com.

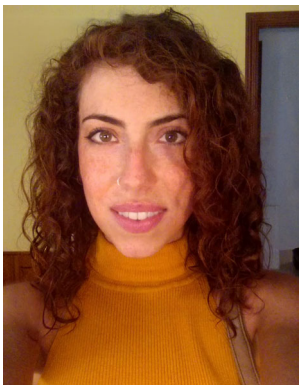
Acknowledgements We would like to thank the MODELS'18 participants, and in particular the EduSymp'18 attendees, for their valuable comments and suggestions on the previous version of this proposal. We are also grateful to all the researchers and educators who participated in the survey. Thanks are due to the Editors of the SoSyM journal for giving us the opportunity to present this initiative.

Funding Open access funding provided by Mälardalen University.

References

1. Adcock, R. (ed.): Guide to the Systems Engineering Body of Knowledge (SEBoK), Version 1.9 (2017). https://www.sebokwiki.org. Accessed 16 July 2019
2. Agner, L.T.W., Lethbridge, T.C., Soares, I.W.: Student experience with software modeling tools. Softw. Syst. Model. (2019). https://doi.org/10.1007/s10270-018-00709-6
3. Bézivin, J., Vallecillo, A., García-Molina, J., Rossi, G.: MDA at the age of seven: past, present and future. UPGRADE **IX**(2), 4–6 (2008)
4. Bourque, P., Fairley, R.E. (eds.): Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 3.0. IEEE Computer Society Press (2014). https://www.computer.org/web/swebok. Accessed 16 July 2019
5. Brambilla, M., Cabot, J., Wimmer, M.: Model Driven Software Engineering in Practice, 2nd edn. Morgan and Claypool, San Rafael (2017)
6. Ciccozzi, F., Famelis, M., Kappel, G., Lambers, L., Mosser, S., Paige, R.F., Pierantonio, A., Rensink, A., Salay, R., Taentzer, G., Vallecillo, A., Wimmer, M.: How do we teach modelling and model-driven engineering?: a survey. In: Proceedings of

Edusymp'18@MODELS, pp. 122–129. ACM (2018). https://doi.org/10.1145/3270112.3270129
7. Ciccozzi, F., Famelis, M., Kappel, G., Lambers, L., Mosser, S., Paige, R.F., Pierantonio, A., Rensink, A., Salay, R., Taentzer, G., Vallecillo, A., Wimmer, M.: Towards a body of knowledge for model-based software engineering. In: Proceedings of Edusymp'18@MODELS, pp. 82–89. ACM (2018). https://doi.org/10.1145/3270112.3270121
8. Combemale, B., Lämmel, R., Wyk, E.V.: SLEBOK: the software language engineering body of knowledge (Dagstuhl Seminar 17342). Dagstuhl Rep. **7**(8), 45–54 (2018). https://doi.org/10.4230/DagRep.7.8.45
9. DAMA International (ed.): Data Management Body of Knowledge (DMBOK), 2nd edn (2017). https://technicspub.com/dmbok/. Accessed 16 July 2019
10. EA community: Enterprise Architecture Body of Knowledge (EABOK) (2015). http://www.eabok.org/. Accessed 16 July 2019
11. Friedenthal, S., Griego, R., Sampson, M.: INCOSE model based systems engineering (MBSE) initiative. In: INCOSE 2007 Symposium, vol. 11 (2007)
12. IIBA (ed.): Business Analysis Body of Knowledge (BABOK), Version 3. International Institute of Business Analysis (IIBA) (2015). http://www.iiba.org/babok-guide.aspx. Accessed 16 July 2019
13. Lúcio, L., Amrani, M., Dingel, J., Lambers, L., Salay, R., Selim, G.M.K., Syriani, E., Wimmer, M.: Model transformation intents and their properties. Softw. Syst. Model. **15**(3), 647–684 (2016). https://doi.org/10.1007/s10270-014-0429-x
14. Mens, T., Gorp, P.V.: A taxonomy of model transformation. Electr. Notes Theor. Comput. Sci. **152**, 125–142 (2006). https://doi.org/10.1016/j.entcs.2005.10.021
15. Oliver, G.R.: Foundations of the Assumed Business Operations and Strategy Body of Knowledge (BOSBOK): An Outline of Shareable Knowledge. Darlington Press, Durham (2012)
16. Project Management Institute (ed.): A guide to the Project Management Body of Knowledge (PMBOK guide), 6th edn. (2017). https://www.pmi.org/pmbok-guide-standards/foundational/pmbok. Accessed 16 July 2019
17. Sands, N.P., Verhappen, I. (eds.): Automation Body of Knowledge (ABOK), 3rd edn. (2018). https://www.isa.org/store/a-guide-to-the-automation-body-of-knowledge,-third-edition/60891879. Accessed 16 July 2019
18. Watson, A.: A brief history of MDA. UPGRADE **IX**(2), 7–11 (2008)
19. Zaytsev, V. (ed.): Software Language Engineering Body of Knowledge (SLEBOK) (2018). http://slebok.github.io/. Accessed 16 July 2019



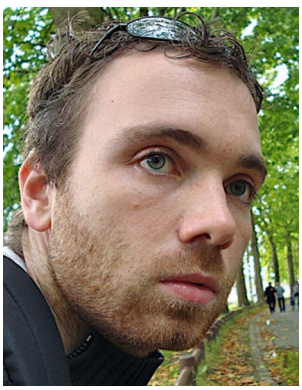
Loli Burgueño is a postdoctoral researcher at the Open University of Catalonia (UOC), in Barcelona (Spain) and CEA List in Paris (France). Her research interests focus on Model-Driven Engineering (MDE). She has worked on the field of testing model transformations, the distribution of very large models and the parallelization of the execution of model transformations, the formalization of Complex-Event-Processing languages, and the modeling of uncertainty in software models for its use in the Industry 4.0. She is also working on the integration of Artificial Intelligence techniques into modeling tools and processes. Further information can be found at https://som-research.uoc.edu/loli-burgueno/.

Federico Ciccozzi is an Associate Professor at Mälardalen University, Sweden. His research focuses on the definition of metamodels and model transformations for several automation aspects in the model-based engineering of component-based software for embedded real-time systems. Moreover, he carries out research on several aspects of teaching and education, with particular focus on distributed education and cultural differences. He has co-authored more than 80 publications in these areas. Further information can be found at http://www.es.mdh.se/staff/266-Federico_Ciccozzi.

Michalis Famelis is an assistant professor at the Department of Computer Science and Operations Research at the Université de Montréal, where he is a member of the GEODES software engineering research team. Prior to joining the Université de Montréal, he was a postdoctoral fellow at the Software Practices Lab at the Department of Computer Science at the University of British Columbia. He received his PhD in computer science in 2016 from the University of Toronto. His research interests include model-driven engineering, formal methods, software analysis and design, and empirical software engineering.

Gerti Kappel is a full professor in the Institute of Information Systems Engineering at TU Wien, heading the Business Informatics Group. Her current research interests include Model Engineering, Web Engineering, and Process Engineering, with a special emphasis on Cyber-Physical Production Systems. For further information, please contact her at gerti@big.tuwien.ac.at or visit https://www.big.tuwien.ac.at/people/gkappel.

Leen Lambers is a senior researcher in the group System Analysis and Modeling at the Hasso Plattner Institute, University of Potsdam in Germany. Her main research area is formal modeling and analysis in software engineering, focusing graph and graph transformation engineering in particular. She served as PC chair/member for several international workshops and conferences related to modeling and analysis in software engineering. She received a PhD for her dissertation "Certifying Rule-Based Models using Graph Transformation" at the Technical University of Berlin in 2009.



Sebastien Mosser is Professor of Software Engineering at Université du Québec à Montréal (UQAM). His research interests cover model-driven engineering, software composition, software development and separation of concerns mechanisms, with a particular focus on domain-specific languages. His teaching is related to software engineering, software design, and modeling. Sébastien has been part of the organization and program committees of several conferences and workshops related to model-driven engineering and software development, including industrial events.

Richard F. Paige is Professor of Software Engineering at McMaster University, Canada, and holds the Chair of Enterprise Systems at the University of York, UK. He has published almost 300 papers on topics related to software engineering, safety and security. He has chaired numerous software engineering conferences and workshops and is on the editorial boards of Software and Systems Modeling, the Journal of Object Technology and Empirical Software Engineering. His research interests are in model management, model-based systems engineering, software processes, agile methods, and safety critical systems.

Alfonso Pierantonio is Associate Professor in Computer Science at the University of L'Aquila (Italy). His research interests include Model-Driven Engineering with a specific emphasis on co-evolution, bidirectionality, and analytics. He has been involved in several national and international projects including H2020 ITN Lowcomote and H2020 Typhon. He has been general chair of STAF 2015, PC chair of ECMFA 2018 and on the organizing, steering, and program committees of many international conferences. He is the editor-in-chief of the Journal of Object Technology (JOT) and a member of the editorial board of the Journal on Software and Systems Modeling (SoSyM).

Arend Rensink has held the chair of Software Modelling, Transformation and Verification at the University of Twente since 2010. His main research interests are the support or software evolution and maintainability using Model-Driven techniques, in particular Graph Transformation.

Rick Salay is a researcher in software modeling with over 40 peer-reviewed papers in the area. He has conducted and led internationally recognized research in modeling on topics including safety assurance modeling, model management, model uncertainty, and model transformations. He regularly participates in program committees for international conferences related to software engineering, modeling and safety and has a very active role in industrial projects on these topics with major companies.

Gabriele Taentzer is professor in software engineering at the Philipps-Universität Marburg. Her research interests include formal foundations and applications of model-driven software engineering, graph transformation, and software quality assurance. Contact her at taentzer@informatik.uni-marburg.de, or visit www.uni-marburg.de/fb12/swt.

Antonio Vallecillo is Professor of Software Engineering at the University of Málaga, Spain, where he leads the Atenea Research Group. His current research interests include Model-based Software Engineering, Open Distributed Processing, and Software Quality. More information about his publications, research projects, and activities can be found at http://www.lcc.uma.es/~av, or contact him at av@lcc.uma.es.

Manuel Wimmer is a full professor leading the Institute of Business Informatics—Software Engineering at the Johannes Kepler University Linz. His current research interests are focused on the foundations and applications of model-driven engineering technologies. Contact him at manuel.wimmer@jku.at, or visit https://www.se.jku.at/manuel-wimmer.