Applying guidelines for system modelling in Event-B

Olszewska, Marta; Shokri-Manninen, Fatima; Edmunds, Andrew; Walden, Marina

Published in:
International Journal of Critical Computer-Based Systems

DOI:
10.1504/IJCCBS.2020.108668

Publicerad: 01/01/2020

Document Version
(Referentgranskad version om publikationen är vetenskaplig)

Document License
Publisher rights policy

Applying guidelines for system modelling in Event-B - A Systematic Literature Review

Marta Olszewska
Faculty of Science and Engineering,
Åbo Akademi University,
Turku, Finland
E-mail:Marta.Plaska@abo.fi

Fatima Shokri-Manninen
Faculty of Science and Engineering,
Åbo Akademi University,
Turku, Finland
E-mail:Fatemeh.Shokri@abo.fi

Andrew Edmunds
Faculty of Science and Engineering,
Åbo Akademi University,
Turku, Finland
E-mail:andy.ed.edmunds@gmail.com

Marina Waldén
Faculty of Science and Engineering,
Åbo Akademi University,
Turku, Finland
E-mail:Marina.Walden@abo.fi

Abstract: Developing safety-critical systems is an intricate task since it involves the application of well-established and rigorous methods, supported by good practices. The modelling is merely a part of this undertaking. However, it plays a significant role in the description of the system, how it will behave and what properties it will have. Formal methods, for instance Event-B, are utilised in such cases to assure that the system is correct-by-construction and functions as required. In this work we use a literature review method to collect a body of knowledge that would support the Event-B practitioners with modelling guidelines. We first define the domains in which the guidelines fall and divide them into two categories: beginners and advanced. Then we provide a collection of guidelines as reinforcements for domain-specific applications of Event-B.

Keywords: Event-B, Formal modelling, Guidelines, Lessons learned, Practitioners.

1 Introduction

Development of safety-critical or mixed-criticality systems requires a high degree of rigour of the methods in use. For the assurance of their quality, herein correctness, as well as for the standardization purposes, formal methods are employed (Olszewska, 2011). Formal methods are mathematical techniques for specification, development and verification of software and hardware systems. They can be a foundation for describing complex systems, aid in reasoning about systems and support system development.

Systems are increasingly dependent on software components and the complexity of components (and systems in general) is growing. Formal methods facilitate control of the complexity and providea a correct-by-construction system development, so that the system functions as required (Almeida et al., 2011). Moreover, maintaining the reliability in software-intensive systems, including embedded systems, becomes more difficult by the time. Modelling and proving such systems in a rigorous manner helps to describe their properties accurately, and thus gives confidence in how the system behaves. This is vital in cases where human losses or large financial gains are at stake, e.g. in space or transportation domains (Bowen and Hinchey, 2012).

Modelling in Event-B provides many benefits, such as obtaining a correct-by-construction system, control over the specified requirements, as well as sound management of the early stage development process (Abrial, 2010). However, it is considered as a challenging and time-consuming task, in particular for the beginners, since it involves a steep

learning curve. The challenges with creating the initial model in Event-B and then refining it according to certain formal rules are balanced out by the certainty that the model is proven correct according to given requirements and specified assumptions, once the skill is mastered (Su et al., 2014). In order to build the model in an efficient and sustainable way, a feasible strategy is needed to help with the process all the way. Thus, providing a collection of guidelines to facilitate the early-stage development would improve the experience of modelling. Currently, this experience depends vastly on the level of proficiency of modellers (Abrial, 2010). In this paper we mainly focus on the Event-B formal method and supplement it with some material regarding the B-Method (for historical reasons of Event-B being a descendant of the B-Method). We use the systematic literature review method (Kitchenham and Charters, 2007; Kitchenham et al., 2009) to obtain a body of knowledge that would serve to improve the Event-B modelling experience. The guidelines we collect are meant for Event-B users, who have barely begun to apply this particular formal method, as well as experienced developers, who can use the material as a quick-roadmap on how to manoeuvre in the Event-B ecosystem. To our knowledge, there exist guidelines for managing the Rodin tool (Jastram and Butler, 2014), however, the information on strategies and best practices for modelling in Event-B is very much spread over the literature. The goal of this paper is to show the breadth of the topic and possibilities, pointing to the literature, which can be exploited in-depth, when needed.

This paper, apart from being a literature study, is based on the results from a 4-year project ADVICeS (Adaptive Integrated Formal Design of Safety-Critical Systems) funded by Academy of Finland (ADVICeS Project, 2013-2017). The aim of the project was to make the formal design process for developing complex systems more efficient, flexible and maintainable. A combination of an adaptive design framework, based on concepts of an agile methodology, with formal methods (in particular Event-B) was to augment the development flexibility and give faster response to changes in requirements and design decisions. As a consequence, this aids to achieve a feasible formal development process that enhances maintainability. As an integral part of the development process, metrics and quality measurements specific to the context will provide additional guidelines for the modelling process. Furthermore, they will enable the assessment of the suitability of the proposed hybrid method.

This paper is structured as follows. In Section 2 we present the Event-B method and modelling language and motivate why it was chosen in our work. Next, in Section 3, follows the description of the context of our study, including the research questions. Section 4 presents the review method including a quality assessment. Sections 5 and 6 answer the research questions. We continue to discuss our findings and validity issues in Section 7 and conclude in Section 8 with some general remarks.

2 Background – Event-B

Development of safety-critical systems requires a high degree of rigour to be able to prove that the system will behave as expected. Correctness by construction is obtained with the use of formal methods, which are the mathematical means to describe the system to be developed together with its properties. There are various formal methods that can be utilised, depending on the application domain or the type of the modelled system. We chose to focus on Event-B (Abrial, 2010), as it is not only investigated in the project ADVICeS (ADVICeS Project, 2013-2017), but also because we have long experience with using this method for development of safety-critical systems, including distributed and reactive systems, and it has gained an industrial interest (ENABLE S3 Project, 2016-2019).

Event-B is derived from the B-Method, a formal language for specifying software systems, which covers all stages of development from requirements (specification) and design (refinement) to implementation and code generation. The B-Method uses abstract machines that include states and operations, whereas in Event-B the operations are substituted by events that have no explicit parameters. At most one event is executed non-deterministically at a time. Event-B was designed to allow development of distributed systems and is used for the development of complete systems, encapsulating hardware, software and environment. The refinement in Event-B is more flexible when compared to the classical B-method, since adding new events, merging or splitting events are allowed. Moreover, model decomposition has been introduced in Event-B to reduce the complexity of modelling big systems by dividing them into smaller submodels (Abrial, 2010).

Event-B uses refinements (Back, 1990; Back and von Wright, 1998), which enables the system to be created in a stepwise fashion, starting from an abstract model, where each refinement step must shown to preserve the correctness of previous steps (Back, 1978; Back and Sere, 1994). If model A is refined by model B, then all behaviour present in B should be present in A as well. The developer defines his or her own abstract model and refinements and then uses proofs to check the correctness of the refinement. The refinement approach can be perceived as a rigorous development process which clarifies the system specification and manages the complexity of specification. Since the refined models are tightly dependent on the more abstract models, an inaccuracy or error in one level can cause an inconsistency in the whole model. Therefore, tools for maintaining the refinement process are crucial for the system development.

Event-B is well supported by the Rodin platform (RODIN Project, 2004-2007; Rodin Platform, 2006), which is an Eclipse-based, open-source, rich client platform, extendable with plug-ins. A diversity of plug-ins is available to provide a more flexible and adaptable environment for Event-B development. For instance, ProB (Leuschel and Butler, 2008) and the model decomposition plug-in (Said et al., 2015), include both a separate animator and an Eclipse plug-in for animating and model checking of B and Event-B models. Property violations in the model can be detected

by finding counterexamples. Furthermore, ProB can be used in automated refinement checking. The refinement error detection capability in ProB discovers inconsistent behaviours. A model decomposition plug-in (Butler, 2009) is available for breaking a model into sub-models in Event-B. Two approaches are available: shared variable decomposition and shared event decomposition (Hoang et al., 2011). The user can optimise the decomposition style, depending on the user preference and the system.

However, not only is Event-B well supported by the tool, but also by the community which extends the Event-B ecosystem. The Event-B ecosystem can be understood as co-operative of developers, software organisations, and third parties that share a common interest in the development of the Event-B language, software that supports it (Rodin platform with plug-in extensions), as well as advertising its application. Event-B is a technology that is developed and co-evolved by a community based on shared information and resources to fit contemporary development processes and practices, and at the same time scaling up to the needs of industry. In fact, Event-B has been gaining recognition in an industrial setting. Recently, it was one of the methods used in the European project ENABLE S3 (ENABLE S3 Project, 2016-2019), which was an industry-driven undertaking that aimed to provide more advanced and efficient methods to commercialise highly automated cyber physical systems (ACPS).

3 Context and research questions

The goal of this paper is to review existing approaches and give recommendations for the use of Event-B in modelling of systems to create a best-practices roadmap for practitioners. From this goal, we derive the following two research questions:

- **RQ1:** How to effectively support the developers using Event-B both at the beginners and advanced levels?

The main goal of RQ1 is to first categorise the Event-B modelling guidelines and then to create guidelines within these categories to be able to support the Event-B practitioners regardless of their familiarity and skill in the method. We aim at dividing the guidelines for the beginners and advanced levels, where the beginners-level is merely a subset of the guidelines anticipated for the more advanced modellers. Our reasoning is to provide the beginners with the get go knowledge and not overwhelm them with modelling nuances, i.e, to (i) introduce them to Event-B modelling in a stepwise manner (steep learning curve of formal methods) and to (ii) sustain their motivation.

- **RQ2:** How can modellers in different modelling areas take advantage of the guidelines proposed in this paper?

Research question RQ2 is meant to tackle the issue of deepening the knowledge of the modelling process with respect to the modelling area, the application domain or the type of the modelled system. This approach uses the findings of RQ1 (categorization of guidelines) to provide more detailed knowledge with respect to the chosen case study. We demonstrate how modellers can use the proposed guidelines in each step. While RQ1 gives a horizontal knowledge on the modelling with Event-B, RQ2 presents a vertical, step by step approach and shows the issues that need to be addressed when modelling and proving a system.

4 Review method

This review has been undertaken according to the guidelines proposed by Kitchenham (Kitchenham and Charters, 2007; Kitchenham et al., 2009) for a systematic literature review which aims at analayzing all the relevant studies in a way that is unbiased. We followed the process for conducting systematic literature reviews: 1) Research questions, 2) Search process, 3) Inclusion and exclusion criteria, 4) Quality assessment, 5) Data extraction, 6) Data synthesis.

4.1 Search process

We used the following search engines for our reviews: ACM Digital Library, Google Scholar, IEEE Xplore, Elsevier ScienceDirect, and SpringerLink. For the search, we used the following terms: “Event-B ”, “Formal methods”, “Agile”, “industrial case study”, “guidelines”, “lessons learned” as the aim was to extract existing methods and guidelines for formal modeling in Event-B. To help modellers, we also searched for experience reports and case studies and then draw conclusions from their experiences and/or lessons learned. Therefore, we created five search strings:
1. “Event-B” AND “Formal methods”
2. “Event-B” AND “Agile”
3. “Event-B” AND “Industrial case study”
4. “Event-B” AND “Guidelines”
5. “Event-B” AND “Lessons learned”

Figure 1 shows the process of study selection and the number of papers identified at each step. When searching for papers according to the above search strings in the electronic databases and conference proceedings (step 1), we found about 300 studies. After that, we did the practical screening for excluding papers based on the title and abstract in case of being irrelevant (step 2) and ended up with almost 150 papers. For refining our studies, we applied the inclusion and exclusion criteria (step 3).

To guarantee a comprehensive extraction of data for covering the most important studies related to our research questions, we also searched for literature via a technique called “snowballing” (Wohlin, 2014) where we identify new papers from the reference lists or citations, either backward or forward, based on the papers being examined. Moreover, we contacted the most active researchers in the area. Furthermore, for including more papers on the topic, we searched in specific journals and conferences, for example the ABZ conference (step 4). Table 1 outlines primary studies for conducting the review. For example, studies [S26, S29 and S30] found by the reference list or citations while studies

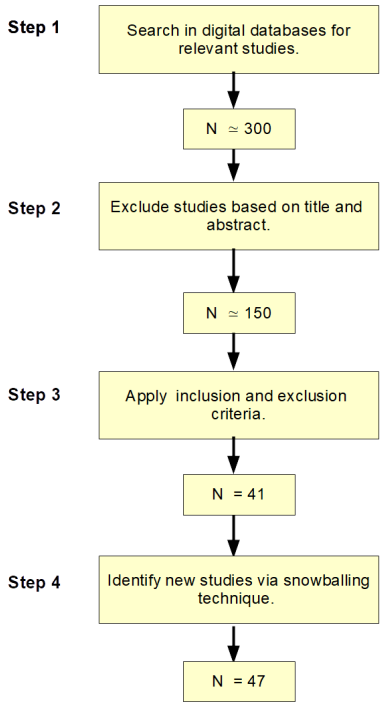

Figure 1 Overview of the study selection process

[S7, S37 and S40] are collected by contacting the active researchers in Event-B.

4.2 Inclusion and exclusion criteria

In step 3, of Figure 1, we applied inclusion and exclusion criteria and the number of papers dropped to 41. The main criterion for the choice of papers was introducing a new method or plug-in for the Event-B platform and also the practicality of the suggested guidelines. The requirement for the papers that were included in this work was that they should be supported by a (preferably industrial) case study or an example. Moreover, we included the work that was industry-driven, either via international projects or executed in the R&D site of an organisation. In cases of surveys, a strong justification of usability of the suggested approach sufficed. Yet another criterion was based on the approaches already well-developed and verified. Moreover, we included all studies that indicated some form of experience with Event-B development.

Studies were excluded if their focus, or main focus, was not modelling in Event-B. Another criterion for exclusion was being non-peer reviewed studies. We also excluded duplicated and purely theoretical studies. Concerning the new version of the Rodin tool, we also exclude the studies and works not maintained anymore. There was a team of four experts involved in choosing the literature, among which there was a senior researcher, two postdoctoral researchers and one PhD student. The preliminary selection was executed within the group of early-career researchers and then iterated with the senior researcher for approval.

4.3 Quality assessment

We have defined a list of questions to assess the reliability and usability of the study. The Quality assessment (QA) questions are as follows:

- **QA1:** Are all model construction methods fully defined concerning tools and methods used?
- **QA2:** Does the paper introduce a new method and plug-in for Event-B?
- **QA3:** Are there adequate guidelines for formal modelling of the case study?
- **QA4:** Does the paper contain a “lessons learned” report (based on expert opinion)?

Each of the 47 studies that remained after step 4 was assessed by the authors according to the above four QAs. These four criteria can be considered as a measurement for providing confidence that the findings in a particular study can make a valuable contribution to the review. The grading of each of the four criteria was done on a (“yes”,“partly ” and “no”) scale. The questions were scored as follows:

- **QA1:** Y (yes), all model construction methods are explicitly defined in the study; P (Partly), the model construction methods are only implicit; N (no), the model construction methods are not defined.
- **QA2:** Y (yes), the paper introduces a new method and plug-in for Event-B; P (Partly), the paper introduces either a method or a plug-in for Event-B; N (no), the paper does not introduce a method nor a plug-in for Event-B.
- **QA3:** Y (yes), there are adequate guidelines for formal modelling of the case study; P (Partly), the guidelines for formal modelling of the case study are implicit; N (no), no guidelines are defined.
- **QA4:** Y (yes), the paper contains a “lessons learned” report; P (Partly), lessons learned are mentioned implicitly in the conclusions; N (no), the paper does not mention “lessons learned”.

The scoring procedure was Y (yes) = 1, P (partly) = 0.5, N (no) = 0. The grading process has been performed by the authors of this paper. When there was a disagreement, it was resolved by discussion. The results of the quality assessment are shown in Table 1.

The results of the quality analysis show that most of the studies scored 2 or more and only three studies scored less than 2 ([S26], [S27] and [S34]). They are concerned with the tool support and considered valuable for this paper, in spite of their low score. Three studies reached almost full score, 3.5 out of 4 ([S8],[S19] and [S47]).

Table 1 Quality assessment

Study	Author	QA1	QA2	QA3	QA4	Total score
S1	(Jastram and Butler, 2014)	1	0.5	1	0.5	3
S2	(Abrial, 2010)	1	0.5	1	0.5	3
S3	(Siqueira et al., 2017)	1	0.5	1	0	2.5
S4	(Hallerstede et al., 2014)	1	1	0.5	0	2.5
S5	(Snook and Butler, 2008)	1	1	0	0.5	2.5
S6	(Kobayashi et al., 2014)	0.5	0.5	0.5	0.5	2
S7	(Fathabadi et al., 2018)	1	1	0.5	0	2.5
S8	(Hoang et al., 2013)	1	1	1	0.5	3.5
S9	(Butler and Maamria, 2013)	1	1	0	0.5	2.5
S10	(Olszewska et al., 2016)	0.5	0.5	0	1	2
S11	(Fotso et al., 2018)	1	0.5	0	0.5	2
S12	(Said et al., 2015)	1	1	0.5	0.5	3
S13	(Rodríguez, 2013)	1	0.5	1	0.5	3
S14	(Edmunds et al., 2016)	1	0.5	0	0.5	2
S15	(Prokhorova and Troubitsyna, 2012)	1	0.5	0.5	0.5	2.5
S16	(Lopatkin et al., 2011)	1	0.5	0.5	0	2
S17	(Cofer et al., 2012)	1	0.5	0.5	0	2
S18	(Ponsard and Devroey, 2011)	1	1	0	0	2
S19	(Butler, 2009)	1	1	1	0.5	3.5
S20	(Hoang et al., 2011)	1	0	1	0.5	2.5
S21	(Fathabadi et al., 2012)	1	1	0.5	0	2.5
S22	(Edmunds et al., 2016)	1	0.5	0.5	0.5	2.5
S23	(Silva and Butler, 2009)	1	1	0.5	0	2.5
S24	(Iliasov et al., 2010)	1	1	0	0	2
S25	(Hoang et al., 2017)	1	1	1	0	3
S26	(Servat, 2007)	0	1	0	0.5	1.5
S27	(Ladenberger et al., 2009)	0	1	0	0	1
S28	(Ostroumov and Waldén, 2017)	1	0.5	0.5	0	2
S29	(Leuschel and Butler, 2008)	1	1	0.5	0	2.5
S30	(Yang et al., 2013)	1	1	0	0	2
S31	(Savicks et al., 2014)	1	1	0.5	0	2.5
S32	(Sato and Ishikawa, 2015)	1	0.5	1	0	2.5
S33	(Déharbe et al., 2014)	1	1	0.5	0.5	3
S34	(Dinca et al., 2012)	0.5	1	0	0	1.5
S35	(Lanoix, 2008)	1	0	1	0.5	2.5
S36	(Yeganefard et al., 2010)	1	0	1	0.5	2.5
S37	(Hoang et al., 2016)	1	0	1	0.5	2.5
S38	(Cansell et al., 2007)	1	0.5	1	0	2.5
S39	(Rezazadeh and Butler, 2005)	1	0	1	0	2
S40	(Snook et al., 2017)	1	0	1	0.5	2.5
S41	(Mashkoor and Jacquot, 2011)	1	0	1	0.5	2.5
S42	(Su and Abrial, 2017)	1	0.5	1	0.5	3
S43	(Comptier et al., 2019)	1	0.5	0.5	0.5	2.5
S44	(Eschbach, 2019)	1	0	0.5	0.5	2
S45	(Dieumegard et al., 2017)	1	0	0.5	1	2.5
S46	(Mammar and Laleau, 2017)	1	0	1	1	3
S47	(Méry et al., 2015)	1	0.5	1	1	3.5

4.4 Data extraction

During the data extraction stage, data was extracted from each of the 47 primary studies included in this systematic review and was performed by all of the authors of this paper. The data extraction divided the studies into three categories; the first category of papers consists of studies introducing tools and methods for formal modelling, the second category includes papers that propose guidelines for modelling, and the last category covers experience reports from modelling in Event-B.

We collected the data for the articles that were published until the end of September 2019. We found relevant work in books, journals, conferences, and workshops with a referee process. Table 2 shows the number of publications found for each venue. The reviewed empirical articles consist of 3 books, 9 journal articles, 31 conference articles and 4 workshops articles with the earliest article from year 2005 and the latest from 2019.

Table 2 Distribution of studies according to the publication venue.

Publication source	Type	Number
CreateSpace Independent Publishing Platform	Book	1
Cambridge University Press	Book	1
PhD Thesis	Book	1
Theories of Programming and Formal Methods	Journal	1
Software & Systems Modeling	Journal	2
Science of Computer Programming	Journal	2
Journal of Systems Architecture	Journal	1
Software Tools for Technology Transfer	Journal	3
Abstract State Machines, Alloy, B, TLA, VDM, and Z (ABZ)	Conference	7
Product-Focused Software Process Improvement (PROFES)	Conference	1
Advances and Trends in Software Engineering (SOFTENG)	Conference	2
NASA Formal Methods Symposium (NFM)	Conference	4
INFormatique des ORganisations et Systèmes d'Information et de Décision (INFORSID)	Conference	1
Integrated Formal Methods (iFM)	Conference	2
Software Engineering and Formal Methods (SEFM)	Conference	1
International Conference on Engineering of Complex Computer Systems (ICECCS)	Conference	1
Asia-Pacific Software Engineering Conference (APSEC)	Conference	1
International Conference on Formal Engineering Methods (ICFEM)	Conference	1
International Conference of B Users	Conference	1
International Conference of B and Z Users (ZB)	Conference	1
IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE)	Conference	1
The IASTED International Conference on Software Engineering (SE)	Conference	1
High-Assurance Systems Engineering (HASE)	Conference	1
Reliability, Safety, and Security of Railway Systems (RSSRail)	Conference	1
International Symposium on Formal Methods (FM)	Conference	2
International Conference on Model and Data Engineering (MEDI)	Conference	1
International Symposium on High-Assurance Systems Engineering (HASE)	Conference	1
Software Engineering for Resilient Systems (SERENE)	Workshop	1
Automated Verification of Critical Systems (AVoCS)	Workshop	1
Formal Methods in Software Engineering (FME)	Workshop	1
Formal Methods for Industrial Critical Systems (FMICS)	Workshop	1

4.5 Data synthesis

Data synthesis includes assembling and organising the results of the included primary studies. We have selected a descriptive synthesis and display the information extracted from the primary studies in figures and tables. For research question RQ1, we have collected literature for drawing Figuers 2,3 and 4 containing new methods and/or plug-ins for modelling in Event-B according to the two first search strings; 1. "Event-B" AND "Formal methods" 2. "Event-B" AND "Agile". We took into consideration the two experience levels - Beginner and Advanced. The division into modeling levels is the contribution of the authors as from a long modelling experience where two of the authors are from verification and validation and the other two are from modelling development, agile and measurement. The snowballing technique was used at this stage as we contacted a researcher who is active in developing the Rodin plug-ins and methods in Event-B for the current status of the plug-ins. Tables 3 and Tables 4 (in Section 5.3) shows the result of the collected literature for particular methods or Event-B plug-ins with references to publications and their classification according to Fig. 2, 3 and 4.

For research question RQ2, we searched studies based on two seach strings 3; "Event-B" AND "Industrial case study" and 4. "Event-B" AND "Guidelines" to extract the guidelines offered for modelling in Event-B in terms of industrial case studies in different domains. Our objective was to provide self-contained packages in the form of research paper reference concentrating on the application context

or domain. We created a table with a list of 15 entries (publications) that can be used as a handbook for modelling in Event-B depending on the application domain (see Table 5 in Section 6.1).

As the goal of this study was to provide an as extensive as possible body of knowledge for developers, we dedicated Section 6.2 to lessons learned which is based on experience reports of Event-B modellers found via the last search string 5. “Event-B” AND “Lessons learned”.

5 Guidelines

The collection of guidelines is meant to facilitate the modelling process, either by providing the knowledge of the tools to be used or the literature that is available for a certain problem or application domain. In our work we focussed both on (i) building the correct models and providing guidelines for these (product-oriented guidelines), and (ii) the modelling process itself. This includes iterative evolutionary methods (agile methods) as well as techniques that support efficient, continuous development, integration and deployment (reuse and modularisation).

In order to be able to effectively support developers using formal methods, in particular Event-B and the B-Method, we divided them into the ones that are meant for beginners and the ones that target more experienced modellers. We planned the support in the most practical way by dividing the guidelines into categories, which are represented as a tree-like structure. The leaves of the tree represent particular methods or Event-B plug-ins that can enable the fulfilment of certain modelling demands or facilitate the modelling process.

In this section, we address RQ1, i.e., ”How to effectively support the developers using Event-B both at the beginner and advanced levels”. We divide the guidelines with respect to the level of experience of potential users, beginners and more advanced modellers. The purpose of the division is to motivate newcomers. Since modelling in Event-B can be challenging for someone who has just started learning it, we have collected studies that introduce guidelines for lightweight modelling using proved patterns and animations for validation rather than proofs. Furthermore, we cannot expect beginners to model the system completely and then verify the model. For this reason, we do not include the V&V techniques at the beginners level. Once modellers learn more regarding modelling Event-B, they can move from the beginners level to the advanced level.

The classification of the guidelines have crystalized from the collected literature, as well as from the long (more than fifteen years) modelling experience of the authors. To show only the most vital information in the plethora of literature on the subject, the guidelines for beginners are a limited subset of those for the more experienced modellers. They are enriched with the sources for knowledge acquisition that enable a smooth start with modelling using Event-B. In the following subsections we describe the guidelines for the two target groups.

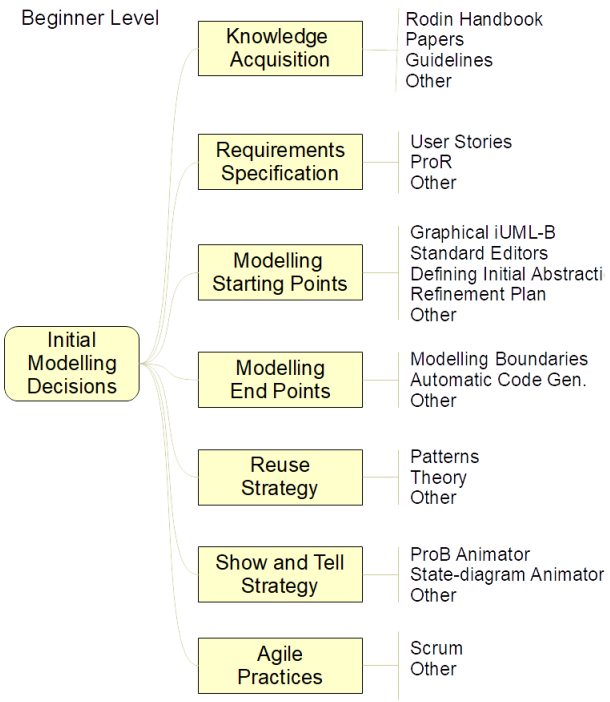

Figure 2 Collections of guidelines for the beginners level

5.1 Beginner support

In Figure 2 we have divided the type of guidelines into categories with respect to the modelling challenges, such as knowledge acquisition, requirements specification, modelling starting- and end-points, reuse strategy, show and tell strategy, as well as agile practices. In this section we will describe them in detail, focusing on the means to tackle them when modelling. In each category we also included a “disclaimer” stating “Other”, which means that there might be another source of guidelines regarded as more suitable for the beginner, depending on the case.

Knowledge acquisition, which is in particular needed at the beginning of the modelling journey with Event-B, has multiple sources, starting from the Rodin Handbook (Jastram and Butler, 2014) and Rodin (*Event-B*, 2001). Moreover, there is a noteworthy literature source available coming from three European projects RODIN (RODIN Project, 2004-2007) and DEPLOY (DEPLOY Project, 2008-2012), ADVANCE (ADVANCE Project, 2011-2014), as well as from the Academy of Finland funded project ADVICeS (ADVICeS Project, 2013-2017). Furthermore, the guidelines on refinement and building the models can be found in the literature by the founder(s) of the method, e.g., (Abrial, 2010).

The requirements specification, which is the baseline for the formal modelling process, can be expressed in various ways from organized, formal listings, which is desirable in formal modelling, to descriptions given in natural language. The most common way of requirement description seems to be the collection of user stories – an informal depiction of a feature (or more features) of a software system, usually

provided in natural language (Siqueira et al., 2017; Cohn, 2004).

In the requirements specification category, we also included the tool support for requirements editing, called ProR (Hallerstede et al., 2014; *ProR tool*, 2014), which can be used. The requirements can be elicited and processed with the support of the ProR tool. They are presented in a table format, in which the relations between the requirements are enabled as annotated links.

There are many modelling starting points to consider. For example, there is begin with the choice of editor. They range from text-based editors, like standard Rodin or Camille text editors, the Event-B editor, text editor or tree-like EMF editor, to graphical ones like iUML-B (and the older version UML-B) (Said et al., 2015; Snook and Butler, 2008). After choosing the environment for modelling, the initial structure of the system (level of abstraction) needs to be identified. Finally, the refinement plan needs to be decided, meaning that the sequence of introducing and detailing the features or requirements should be prearranged. The plan should take into consideration how easy it will be to model and prove the system which later on, can be based on experience, but for a beginner should be based on some already existing guidelines, like in Kobayashi et al. (Kobayashi et al., 2014), Sato et al. (Sato and Ishikawa, 2015) or Rodriguez (Rodriguez, 2013).

As modelling end-points we understand the "end-game" for the model, i.e., the last refinement step or the stage of the model when the modelling process is considered to be complete. It can either be the case that in the next step code will be generated, or that the model is ready in terms of what is to be implemented and what is to remain as assumed. In the case of the former, modellers are able to generate executable C/C++ code from Event-B formal models by using the PRiME code generation plug-in (Code Generation Activity, 2018; Dalvandi et al., 2018).

The latter case refers to the process of deciding what parts of the system are being modelled explicitly and which the abstractions of the environment are. For example, if the purpose of a model is primarily the correctness of the specification of software components, all external device drivers and sensors may be considered to be abstractions of the environment. If building a model involves pre-existing modelling components, then the component interfaces form part of the boundary (Edmunds et al., 2016).

The system under development is more likely to be implemented by a development team than by one particular developer. It is beneficial for the team members to understand the 'scope' of the model. Invariably the abstraction of the environment contains assumptions that are out of the developer's control. It would, therefore, be important to consider all the assumptions, and to consider how they might affect the integrity of the model.

Since reuse is considered to be good practice in coding and modelling, it is also encouraged in Event-B. It can be achieved for instance with the application of patterns or the so-called theories enabled via the Theory plug-in for reuse of proofs. The former is not only tool supported, but also some guidelines are given for the general purpose (Hoang et al., 2013) or introducing timing patterns in Event-B (Cansell et al., 2007). The latter extend the Event-B language and the proving infrastructure in a fashion familiar to Rodin users. The Theory extension provides a way to add new data types, operators, inference and rewrite rules, as well as code generation translation rules (Butler and Maamria, 2013). There is also an extension of the theory of Real Numbers for embracing discrete and continuous functionalities of hybrid systems in Event-B where all the relevant definitions, theorems and proof rules related to continuous functions are defined (Dupont et al., 2018).

When modelling, it is beneficial to be able to visualise the progress of the development and the current state of the model. The Event-B ecosystem also provides the developers with the "show and tell" tool support. The ProB Animator (constraint solver and model checker) (Leuschel and Butler, 2003) allows fully automatic animation of specifications, and can be used to systematically check a specification for a wide range of errors. Moreover it can be used for deadlock checking and test-case generation. ProB was originally developed for the B-Method and is now extended to also support Event-B (Leuschel and Butler, 2008; ProB, 2018). It can be installed with Rodin, where it comes with BMotionStudio (Ladenberger et al., 2009; BMotionWeb, 2018) to easily generate domain specific graphical visualizations. On the other hand, the State-diagram Animator (Said et al., 2015; Snook and Butler, 2008) provides visual animation of UML-B state-machine diagrams. It "executes" the model so that the modeller can check that the state changes as expected and that the correct events are enabled for the next animation step, and thus validates the model. Multiple diagrams can be animated simultaneously so that the behaviour of refinements and/or nested statemachines can be explored. Both strategies visually inform the modeller if the constructed model is as intended.

Creating a correct-by-construction formal model has been supported by methodologies like refinement (Back, 1978; Back and von Wright, 1998; Back, 1990), tools (the Rodin platform and the associated plug-ins), and best practices, which we mentioned earlier. The experience in formal modelling can be further improved by tailoring the modelling process to the needs of the modeller himself. This can be enabled by employing agile practices, which provide a flexible and adaptive development process that is project-specific. Amongst the plethora of agile practices and processes, we focus here on Scrum (Olszewska et al., 2016; Wolff, 2012). The formal approach supports an iterative development of models, while the agile one adapts Scrum-based management of product development to the modelling phase in terms of providing time frames for the modelling and prioritizing the requirements.

5.2 Advanced support

The developer who is more experienced with formal development is also more demanding when it comes to guidelines and might have more in-depth questions regarding development issues. Hence, the collections of guidelines for the advanced level are more extensive than those at

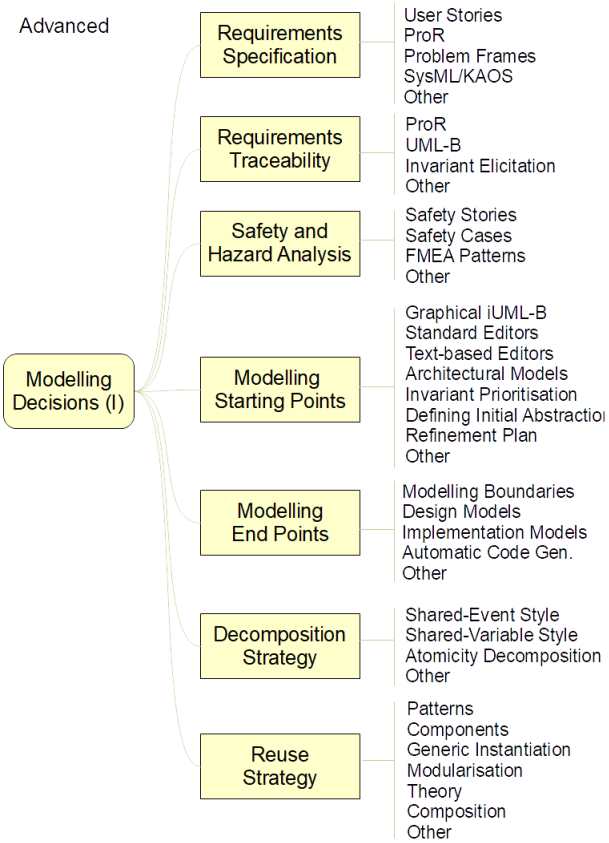

Figure 3 Collections of guidelines for the advanced level (part 1)

the beginners level and consider more alternatives for each category as can be seen in Figures 3 and 4.

We widen the means to tackle requirements specification with Problem Frames (Jackson, 2001), which is an approach to software requirements analysis to be used when gathering requirements and creating specifications for computer software. A clear separation of the requirements, the environment and the system to be built can be achieved. Moreover, it is possible to define the requirements and identify the missing ones more precisely (Romanovsky and Thomas, 2013). This approach was acknowledged as one facilitating the early stage development and requirement elicitation in the automotive domain for developing a cruise control system (Gmehlich et al., 2011).

For representing functional and non-functional requirements, SysML/KAOS is an ideal option (Mammar and Laleau, 2016; Gnaho et al., 2013). The requirements specification is created using the goal-oriented methodology KAOS while the modelling process is performed within SysML, which is a UML profile (Ahmad et al., 2015). Translation between SysML/KAOS domain models and B system specifications has been formally validated in Event-B (Fotso et al., 2018).

Yet another approach to handling requirements, by following their evolution throughout the modelling process and over refinement steps, is requirements traceability. The control over requirements is available for instance via the ProR (Hallerstede et al., 2014) and UML-B (Said et al., 2015) tools. Moreover, Event-B, and formal modelling in general, is considered to be useful for requirements engineering, in particular for invariant elicitation (Rodriguez, 2013).

Since formal methods are most useful when applied to the safety-critical domain, often a safety and hazard analysis is needed in order to model the right properties in the right way. With Event-B it is possible to do this in a variety of ways, such as safety cases (Prokhorova and Troubitsyna, 2012; Prokhorova et al., 2015), safety stories (Edmunds et al., 2016; Ricketts, 2015) or FMEA (Failure Modes and Effects Analysis) patterns (Lopatkin et al., 2011; Prokhorova, 2015). Safety cases are used for instance in the certification process to justify why a system is safe and if the systems properly tackles safety requirements defined in a system requirement specification. Safety stories, on the other hand, are given in an easy to understand narrative, which can be modelled formally. The latter category of guidelines is FMEA patterns. They form the risk identification and are most commonly used in combination with Event-B for rigorously examining the causes and the consequences of hazards.

When compared to the Beginners level, modelling starting points have been extended with more editors, e.g., the text-based editors, which may be appropriate for those who are familiar with the Event-B notation and for whom it might be more efficient to directly code the model. As a modelling starting point, one can consider creating architectural models, which are high level representations of a system from the viewpoint of software architecture (Cofer et al., 2012). Furthermore, invariant prioritization is needed to know which properties of the system are important (and which are more important than others) and when they should be introduced to the model (at what levels of the refinement chain) (Rodriguez, 2013).

Modelling, as a stage of development and an activity, may have an “end-point”, when it is considered to be complete either by the developer or by the requirements for a project. The developer might for example decide that it is sufficient to know the properties of the system and that they are defined correctly, while project requirements need to be fulfilled for instance for the standardization purposes. Modelling endpoints, when framed within the model-driven engineering philosophy, can also touch upon requirements engineering resulting in a concept called the design model (Ponsard and Devroey, 2011). The implementation models, on the other hand, are the models to be obtained when planning the code generation from Event-B. The implementation models have certain assumptions about the environment, which may not be universally valid. This means that the properties of the implementation model are proven for the given case, which may suffice for some systems in their “typical” mode and be an adequate rationale in a certification process (Fathabadi et al., 2018).

The refinement of the system and how it evolves depends on the decomposition strategy which is chosen. The decomposition of the Event-B model is particularly important if the model is of significant size and more than one person is involved in modelling. It provides a mechanism for splitting a large model into several sub-models. The shared event style (Butler, 2009; Hoang et al., 2011) and shared variable style (Hoang et al., 2011) are the two main decomposition

styles supported by the Rodin platform. In the former, a set of events are synchronised and shared by sub-components (synchronization and communication via shared parameters), while in the latter part of the information (variables) is shared among sub-components as in the rely/guarantee approach. The atomicity decomposition (Fathabadi et al., 2012) is yet another technique to help with the structuring of the refinement-based development of complex systems in Event-B. It enhances Event-B refinement with the graphical notation that denotes the relationships between the abstract and concrete (new) events explicitly. In this approach, modelling typically begins with a single atomic event of the system which is split to two or more sub-events in the next refinement step. As a complementary approach to decomposition, there is composition which is used in the bottom-up approach or modularity and reuse scenarios. Such a composition mechanism for refinement based methods has been presented in (Hoang et al., 2017), where both top-down and bottom-up approaches can be used to combine existing models. The work focuses on providing mechanisms for proving the correctness of machine inclusion. The modeller can use CamilleX for this purpose which supports machine inclusion.

While for the beginners level we described patterns and theories which contribute to the model reuse in the Event-B modelling, for the advanced level we enhance it with a concept of components, generic instantiation and modularization. Modelling with the use of components (Edmunds et al., 2016; Ostroumov and Waldén, 2017) is quite an intuitive way to tackle reuse and scalability. Each component is formally developed and proved correct, at the same time supported by the compositionality mechanism. It can be treated as a higher-level placeholder for a "real" component to be developed. Generic instantiation, on the other hand, is meant to instantiate generic models and extend the instantiation to a chain of refinements and is supported by a Rodin plug-in. Sufficient proof obligations are defined to ensure that the proofs associated with a generic development remain valid in an instantiated development, so that there is no need for re-proving (Silva and Butler, 2009). Furthermore, modularization (Iliasov et al., 2010; Hoang et al., 2011) defines a set of interfaces that are shared and accessed by different components. Interfaces provide callable operations and guarantee that these operations can deliver a result for any given circumstance. The implementation of an operation should guarantee that the promises are fulfilled for any given circumstance (Hoang et al., 2011).

For discussing the model among team members or with customers there are a number of show and tell strategies that can support the modeller with his work except for the ProB Animator (Leuschel and Butler, 2008) and State-Diagram Animator (Said et al., 2015; Snook and Butler, 2008). For instance, the B-Motion Studio, also referred to as BMotionWeb (BMotionWeb, 2018), is a tool built on top of ProB for creating interactive visualisations of Event-B models (Ladenberger et al., 2009). By using Brama, an animator that is an Eclipse-based plug-in, it is also possible for modellers to discover problems in a specification. Modellers create B models with, for example, the Rodin

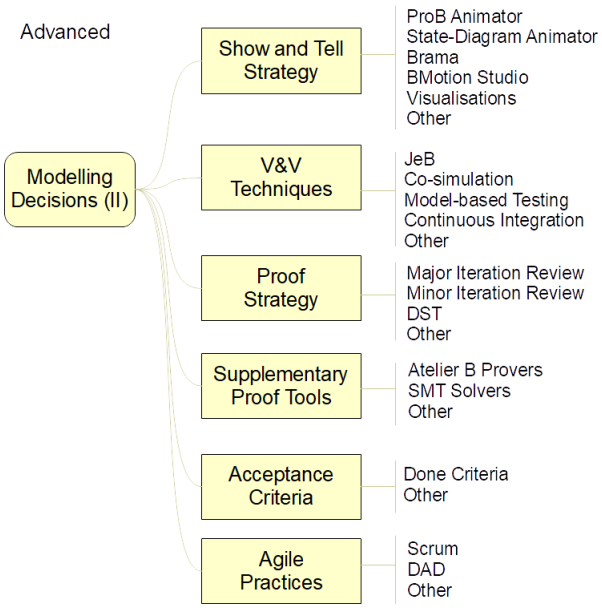

Figure 4 Collections of guidelines for the advanced level (part 2)

platform on which they use the Brama animation tool to test these models (Servat, 2007; Brama, 2011).

Finally, there are other ways of visualizing models that can facilitate the Event-B development, e.g., via a library of visual components (Ostroumov and Waldén, 2017), or illustrating the refinement process itself (Pląska et al., 2007).

On the advanced level, there are broader verification and validation practices (V&V techniques) offered, such as JeB, co-simulation and model based testing. For the purpose of validating a model, modellers can use the JeB simulator. This Java-based framework helps users to generate and execute simulations of Event-B models. By validation of the formal models, the adequacy of the requirements during the development process can be evaluated (Yang et al., 2013; Mashkoor et al., 2017; Jacquot and Mashkoor, 2018). Co-simulation can be utilized as the development progresses to simulate a continuous model of parts of the system with a continuous representation of the environment. For example, for simulation of cyber-physical systems multi-simulation tools can be used, such as that based on the Functional Mock-up Interface (FMI) (Savicks et al., 2014). This approach facilitates modelling, by using a combination of continuous and discrete models, which can offer more confidence that non-functional requirements will be satisfied. Model-based testing is supported in Event-B by a tool that generates test-cases from the Event-B models based on ProB (Dinca et al., 2012). It can be used as a follow up after code generation, as the code generators may not be certified and thus do not provide an appropriate level of confidence. Moreover, the test can be further used as part of a continuous integration process for faster delivery and continuous demonstrations of systems (Edmunds et al., 2016).

When building a formal model, not only are modelling strategies needed to guide the developer, but also proving strategies. The major and minor iteration review concepts stand for the necessity of proofs to be discharged (Edmunds

et al., 2016). In the case of minor review not all proofs need to be discharged. It only needs to be argued on a general level that the model is correct. However, a major review requires the complete discharge of proofs. It is motivated by the fact that the major revision is meant for a "release" or proper deployment of the model, e.g., for building a safety case. Yet another facilitator for proving is the Dividing Strategy Tree (DST) (Sato and Ishikawa, 2015), which helps to design how to prove the safety requirement in a natural language separately from the Event-B formalism. DST has a hierarchical tree-structure of specification descriptions similar to Fault Tree and thus is supposed to make refinement more accessible and motivating for developers in learning Event-B.

Hand in hand with proving strategies, there are supplementary proof tools which are meant to support the modeller in model creation. Examples of such proof tools are SMT solvers (plug-in to Rodin) (Déharbe et al., 2014) and Atelier B Provers plug-in for the Rodin tool (Jastram and Butler, 2014; *Atelier B*, 2016).

When working with a formal model, one needs to define the acceptance criteria for the model. They can either be predefined in the project or decided upon by the modeller himself. Utilising the agile philosophy of the "done" status for the implemented and well-functioning software, the "done criteria" were also defined for formal models, meaning an Event-B model that is "modelled and proven" (Edmunds et al., 2016; Olszewska et al., 2016; Olszewska and Waldén, 2015).

Agile practices, which can facilitate the development process, were enhanced for the advanced level with Disciplined Agile Delivery (DAD) framework (Ambler and Lines, 2012). It is a high-level agile framework, which embraces agile methods in a broader perspective. It can be described as a pick-and-mix approach which, apart from the process of actually building the system, tackles additional process goals, works on the requirements and plans the project, up to delivering a consumable solution (meaning something more than executable code). DAD has been investigated in terms of feasibility for Event-B (Edmunds et al., 2016).

5.3 Result for the first category

For supporting our proposed guidelines, we collected the primary studies related to each category according to Figures 2, 3 and 4. Table 3 and Table 4 show the result of the collected literature referring to either particular modelling techniques or Event-B plug-ins. Each publication is assigned a classification (guidelines category) according to the beginers and advanced levels, presented in the paper in Sections 5.1 and 5.2.

It should be mentioned that for some specific plug-ins we referred to tool support where the primary study was non-existent, e.g. "Atelier B Provers". Therefore, in order to be able to add the meaningful plug-ins to our list of guidelines, we followed a process: we contacted the developer of a plug-in and asked if the plug-in is actively maintained. In this way we excluded the studies related to the plug-ins which are not maintained any more. To the best of our knowledge the plug-ins and methods presented in this paper are supported by the current version of Rodin tools (as of the end of September 2019).

6 Applying guidelines for system modelling

In this section we focus on answering the second research question RQ2, namely: "How can modellers in different modelling areas take advantage of the guidelines proposed in this paper"?

According to the collection of guideline categories for more advanced modellers presented in Section 5.2 (Figures 3 and 4), we will present the road map for system modelling. Since we are focusing on the construction of a complete system here, it requires the modeller to be on advanced level (Dieumegard et al., 2017).

The main advantage of using Event-B is to enable modellers to build models gradually by refinement. Users do not need to implement the whole system at the beginning. When modellers have a deeper understanding of the system requirements they can develop the abstract model as a first step. After that they can extend the model via refinement mechanisms by introducing more details to the system or using a decomposition or reuse strategy. Subsequently, the model can be verified and validated by verification and validation techniques which lead to a complete target model.

Figure 5 shows the whole modelling process with the main steps in blue. The features connected to each main step correspond to the guidelines at advanced level in Figures 3 and 4. In Figure 5 we show Agile Practices as a context for modelling systems in the form of a grey background for the whole modelling process, its steps, iterations and associated methods. Agile practices serve as a catalyst for faster delivery of artefacts, in shorter release cycles, which as a consequence aims to improve quality control and lower the delivery costs.

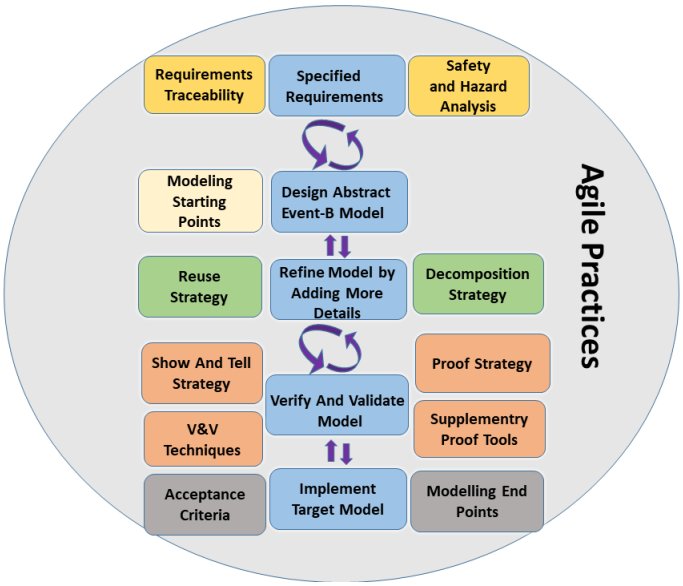

Figure 5 Overview of the modelling process

Table 3 Primary studies regarding methods or Event-B plug-ins for proposed guidelines(part 1)

Study	Author	Paper title	Guidelines category	Figure
S1	(Jastram and Butler, 2014)	Rodin User's Handbook: Covers Rodin V.2.8	Knowledge acquisition (Rodin Handbook) Supplementary Proof Tools (Atelier B Provers)	Figures 2,4
S2	(Abrial, 2010)	Modeling in Event-B: System and Software Engineering	Knowledge Acquisition (Guidelines) Modelling Starting Points (Defining Initial Abstraction)	Figure 2
S3	(Siqueira et al., 2017)	Using BDD and SBVR to refine business goals into an Event-B model: a research idea	Requirement Specification (User Stories)	Figures 2,3
S4	(Hallerstede et al., 2014)	A Method and Tool for Tracing Requirements into Specifications	Requirement Specification (ProR) Requirement Traceability (ProR)	Figures 2,3
S5	(Snook and Butler, 2008)	UML-B and Event-B: an integration of languages and tools	Modelling Starting Points (Graphical iUML-B)	Figures 2,3
S6	(Kobayashi et al., 2014)	Understanding and Planning Event-B Refinement through Primitive Rationales	Modelling Starting Points (Refinement Plan) Requirement Specification (Problem Frames)	Figures 2,3
S7	(Fathabadi et al., 2018)	A model-based framework for software portability and verification in embedded power management systems	Modelling End Points (Implementaion Models, Automatic Code Gen)	Figures 2,3
S8	(Hoang et al., 2013)	Event-B patterns and their tool support	Reuse Strategy (Patterns)	Figures 2,3
S9	(Butler and Maamria, 2013)	Theories of Programming and Formal Methods	Reuse Strategy (Theory)	Figures 2,3
S10	(Olszewska et al., 2016)	Using Scrum to Develop a Formal Model – An Experience Report	Agile Practices (Scrum)	Figures 2,3
S11	(Fotso et al., 2018)	Event-B Expression and Verification of Translation Rules Between SysML/KAOS Domain Models and B System	Requirement Specification (SysML/KAOS)	Figure 3
S12	(Said et al., 2015)	A method of refinement in UML-B	Requirement Traceability (UML-B) Show and Tell Strategy (State-diagram Animator)	Figures 2, 3
S13	(Rodriguez, 2013)	Invariant discovery and refinement plans for formal modelling in Event-B	Requirement Traceability (Invariant Elicitation) Modelling Starting Points (Invariant Prioritization)	Figure 3
S14	(Edmunds et al., 2016)	Using the Event-B Formal Method for Disciplined Agile Delivery of Safety-critical Systems	Safety and Hazard Analysis (Safety stories) Acceptance Criteria (Done Criteria) Agile Practices (DAD) Proof Strategy (Major and Minior Iteration Review) V&V Techniques (Continuous Integration)	Figures 3,4
S15	(Prokhorova and Troubitsyna, 2012)	Linking Modelling in Event-B with Safety Cases	Safety and Hazard Analysis (Safety Cases)	Figure 3
S16	(Lopatkin et al., 2011)	Patterns for representing FMEA in formal specification of control systems	Safety and Hazard Analysis (FMEA Patterns)	Figure 3
S17	(Cofer et al., 2012)	Compositional Verification of Architectural Models	Modelling Starting Points (Architectural Models)	Figure 3
S18	(Ponsard and Devroey, 2011)	Generating High-Level Event-BSystem Models from KAOS Requirements Models	Modelling End Points (Design Model)	Figure 3
S19	(Butler, 2009)	Decomposition Structures for Event-B	Decomposition Strategy (Shared Event Style)	Figure 3
S20	(Hoang et al., 2011)	Event model decomposition	Decomposition Strategy (Shared-Variable Style, Shared-Event Style) Reuse Strategy (Modularisation)	Figure 3
S21	(Fathabadi et al., 2012)	A Systematic Approach to Atomicity Decomposition in Event-B	Decomposition Strategy (Atomicity Decomposition)	Figure 3
S22	(Edmunds et al., 2016)	On Component-Based Reuse for Event-B	Reuse Strategy (Component) Modelling End Points (Modelling Boundaries)	Figures 2,3
S23	(Silva and Butler, 2009)	Supporting Reuse of Event-B Developments through Generic Instantiation	Reuse Strategy (Generic Instantiation)	Figure 3

Table 4 Primary studies regarding methods or Event-B plug-ins for proposed guidelines (part 2)

Study	Author	Paper title	Guidelines category	Fig
S24	(Iliasov et al., 2010)	Supporting reuse in Event B development: modularisation approach	Reuse Strategy (Modularisation)	Figure 3
S25	(Hoang et al., 2017)	A composition mechanism for refinement-based methods	Reuse Strategy (Composition)	Figure 3
S26	(Servat, 2007)	Brama: A new graphic animation tool for B models	Show and Tell Strategy (Brama)	Figure 4
S27	(Ladenberger et al., 2009)	Visualising Event-B Models with B-Motion Studio	Show and Tell Strategy (B-Motion Studio)	Figure 4
S28	(Ostroumov and Waldén, 2017)	Visual Component-Based Development of Formal Models	Show and Tell Strategy (Visuallisations)	Figure 4
S29	(Leuschel and Butler, 2008)	ProB: an automated analysis toolset for the B method	Show and Tell Strategy (ProB Animator)	Figures 2,3
S30	(Yang et al., 2013)	JeB: safe simulation of Event-B models in javascript	V&V Techniques (JeB)	Figure 4
S31	(Savicks et al., 2014)	Co-simulation Environment for Rodin: Landing Gear Case Study	V&V Techniques (Co-simulation)	Figure 4
S32	(Sato and Ishikawa, 2015)	Separation of Considerations in Event-B Refinement toward Industrial Use	Proof Strategy (DST)	Figure 4
S33	(Déharbe et al., 2014)	Integrating SMT solvers in Rodin	Supplementary Proof Tools (SMT Solvers)	Figure 4
S34	(Dinca et al., 2012)	Learn and Test for Event-B – A Rodin Plugin	V&V Techniques (Model-based Testing)	Figure 4

Between the stages *specifying the requirements* and *design of the abstract model* there are circular arrows, since it might take many iterations for constructing a feasible abstract model based on specified requirements. The circular arrows are also applied for the stages *refinement* and *verification* which indicate that the correctness of each refinement step should be verified. In general, down-arrows are used for the development process, while up-arrows are used for traceability.

In the beginning of the modelling process when *specifying the requirements* safety and hazard analysis should also be performed. The *abstract model* can be obtained via the modelling starting point guidelines, while decomposition and reuse strategies can be taken into account when *refining the system*. In the *verification and validation* stage, a modeler can use animation tools in Event-B for animating the model in order to detect errors. With the proving strategy guidelines modellers can be assisted in proving the failed automatic proofs, which indicate how the model should be improved (Krings et al., 2015). At the end of the refinement process the implemented *target model* is reached and can be validated with the acceptance criteria.

When modeling a case study in a particular domain, the user can follow the development guidelines proposed in Figures 3 and 4. Even if different case studies may require different packages as indicated in Table 5, the main steps still remain the same.

6.1 Result for second category

The collection of modelling guidelines, apart from being classified according to the experience of the modeller, can also be classified according to the specifics of a certain domain. The characteristics of the modelled system or the application domain might require some more specific modelling tactics or strategies. For the purpose of our investigation on the system-specific guidelines, we did a systematic literature review and concentrated solely on the published scientific literature (disregarding non-scientific and non-reviewed web content).

We searched for articles based on the two following search strings; “Event-B” AND “Industrial case study”, as well as “Event-B” AND “‘Guidelines” to provide a collection of guidelines. Currently there is no holistic method that guides the developer specifically in the modelling of different domains in Event-B. The collection of modelling guidelines presented in this section are given in form of a table, where the studies are ordered in terms of the type of the modelled system or the application domain. The table displays the paper title with reference, as well as guidelines categories. The categories originate from the guideline categories at beginners and advanced levels presented in Sections 5.1 and 5.2. At times we also specify the detailed guideline classification (a leaf in the guidelines tree), when it is particularly important in the scope of the referenced paper. The papers included needed to be supported by a, preferably industrial, case study or an example and a strong justification of the suggested approach. Yet another criterion was that the presented work should be relatively recent (2005 - 2019) and based on the approaches already well-developed and verified. The results are presented in Table 5.

The domains that are in the scope are mainly the safety-critical ones, like transportation, automotive, banking, spacecraft, aerospace and medical domains. Moreover, guidelines for modelling communication and network protocols are provided, as well as best practices for modelling systems where security is a priority. Among the studies representing the system specific guidelines there are also ones used for enhancements of the development process.

Table 5 System-specific modelling guidelines

Study	Type of system (Domain)	Paper title	Guidelines category
S35	Multi-Agent System / Control System (Transportation)	Event-B Specification of a Situated Multi-Agent System: Study of a Platoon of Vehicles (Lanoix, 2008)	Modelling Starting Points Decomposition Strategy
S32	Multi-Agent System (Network Protocols)	Separation of Considerations in Event-B Refinement toward Industrial Use (Sato and Ishikawa, 2015)	Requirement Specification Modelling Starting Points (Refinement Plan) Proof Strategy (DST)
S36	Control System (Automotive)	Evaluation of a Guideline by Formal Modelling of Cruise Control System in Event-B (Yeganefard et al., 2010)	Modelling Starting Points Decomposition Strategy
S37	Control System (Medical)	Validating the Requirements and Design of a Haemodialysis Machine Using iUML-B, BMotion Studio, and Co-Simulation (Hoang et al., 2016)	Modelling Starting Points Show & Tell Strategy (BMotion Studio) V&V Techniques (Co-simulation)
S28	Control System (Aerospace)	Visual Component-based Development of Formal Models (Ostroumov and Waldén, 2017)	Modelling Starting Points Reuse Strategy (Component) Show & Tell Strategy (Visualisation)
S24	Embedded Software (Spacecraft)	Supporting Reuse in Event-B Development: Modularisation Approach (Iliasov et al., 2010)	Reuse Strategy (Modularisation) Decomposition Strategy (Shared-Variable Style) V&V Techniques
S38	Time Constraint System (Firewire Network Protocol)	Time Constraint Patterns for Event-B Development (Cansell et al., 2007)	Modelling Starting Points Reuse Strategy (Patterns) Proof Strategy
S39	Web-Based Application (E-commerce)	Some Guidelines for Formal Development of Web-Based Applications in B-Method (Rezazadeh and Butler, 2005)	Modelling Starting Points
S40	Network Protocols (VLAN Security)	Analysing Security Protocols Using Refinement in iUML-B (Snook et al., 2017)	Modelling Starting Points Proof Strategy V&V Techniques Show & Tell Strategy (iUML-B, ProB)
S41	Domain Modelling (Transportation)	Guidelines for formal domain modeling in Event-B (Mashkoor and Jacquot, 2011)	Requirement Specification Modelling Starting Points Proof Strategy Show & Tell Strategy Reuse Strategy (Time Patterns)
S25	Components (Transportation)	A composition mechanism for refinement-based methods (Hoang et al., 2017)	Reuse Strategy (Inclusion) Decomposition Strategy (Composition)
S20	A Master Data Updating System (User and Server's Database)	A Survey on Event-B Decomposition (Hoang et al., 2011)	Modelling Starting Points Decomposition Strategy (Shared-Variable Style, Shared-Event Style) Reuse Strategy (Modularisation) V&V Techniques
S42	Aircraft Landing (Aerospace)	Aircraft landing gear system: approaches with Event-B to the modeling of an industrial system (Su and Abrial, 2017)	Requirement Specification (ProR) Modelling Starting Points (Refinement Plan) Supplementary Proof Tools (Atelier B provers and SMT Solvers) Show and Tell Strategy (ProB Animator) Modelling End Points (Implementaion Models, Automatic Code Gen)
S43	Control System (Automotive)	Property-Based Modelling and Validation of a CBTC Zone Controller in Event-B (Comptier et al., 2019)	Modelling Starting Points (Refinement Plan) Show and Tell Strategy (ProB Animator) Supplementary Proof Tools (Atelier B provers)
S44	Control System (Automotive)	Industrial Application of Event-B to a Wayside Train Monitoring System: Formal Conceptual Data Analysis (Eschbach, 2019)	Agile Practices (Scrum) Modelling Starting Points (Refinement Plan) Proof Strategy (Iteration Review)

6.2 Lessons learned

In spite of the popularity of Event-B, some modellers decline to use formal methods. Even with the advent of iUML-B which allows modellers to build a model through graphical design, we still notice that developers in industry hesitate to use formal methods for developing safety-critical applications. Abrial mentions in (Abrial, 2018) that the main reason for poor adoption of formal modelling by some industries such as automotive, aeronautics and space industry is the difficulty to integrate existing development cycles with formal methods (Lecomte et al., 2017).

Below we present some reports on the difficulties encountered by modellers applying formal methods. We believe that this feedback would be constructive for the modellers who want to develop a system through a correct by construction methodology. According to the experience reports the following tips are worthy of consideration during the modelling process:

- Due to lack of guidance provided by the tools and inexperience with the formalism, proving is a difficult task. A short training time such as crash courses on the Event-B methodology, formalism and proof techniques are needed (Dieumegard et al., 2017) (S45 in Table 1).
- For interactive proofs, the Rodin plug-ins Atelier B and SMT provers are good candidates to consider. These tools help not only during the verification steps, but also during the model development (Mammar and Laleau, 2017) (S46 in Table 1).
- Having a good understanding of system requirements leads you to correct implementation of a system (Dieumegard et al., 2017).
- It would be good that the refinement strategy is constructed by the person in charge of writing the Event-B model (Dieumegard et al., 2017).
- Making use of explicit types for numerical units (rather than treating them as subsets of N) makes the model easier to understand for the stakeholders and results in clearer proofs from the modelling perspective (Méry et al., 2015) (S47 in Table 1).
- Considering time constraints only later in the design facilitates the proof activities (Mammar and Laleau, 2017).
- Before proceeding to the software design phase, the formal models should be proven correct with respect to their (correctly and unambiguously stated) requirements specifications (Méry et al., 2015).
- A refinement-based modelling method has an important role in requirement traceability and detection of missing and contradicting requirements (Méry et al., 2015).
- Using animation and validation tools such as ProB, B-Motion, and JeB for visualising a model helps stakeholders a lot to understand how requirements are modelled formally and validated (Dieumegard et al., 2017).
- Having discussions with domain experts helps to validate the modelling decisions. Besides, it reveals new, subtle yet important, properties that the model was lacking (Méry et al., 2015).

It is beneficial to investigate in advance what method should be applied for modelling a given system and its features. Clearly, the implementation of the formal method is not a straightforward task. It requires a lot of experience, skills, time and effort to verify and validate the model. Formal methods should mainly be used for the critical parts of a system; Event-B does not have to be the only method for modelling the system. By using lightweight approaches, we can model safety critical parts of a system in a formal manner verifying particularly critical properties (Jackson, 1996), while the rest of the system is modelled with a semi-formal method. This is called mixed criticality approach. Formal methods can also be combined with techniques, which make the process of modelling faster and shortens the time of delivery. In our previous work (Olszewska et al., 2016), we combined Agile methods with Event-B in order to increase the efficiency of modelling.

7 Discussion

The papers in the previous section show that for every case study there are different guidelines with various approaches for modelling, which can be difficult for a developer to choose from. In order to increase the efficiency of modelling, we need to have a good overview of specific guidelines, which would advise the potential modeller on how to obtain comprehensive guidelines whenever needed. The present review is aimed at providing that information and serving as a roadmap for Event-B developers at beginners, as well as advanced levels.

7.1 General findings

Our work shows that there is a considerable body of knowledge that can serve for building the skillset of Event-B developers and for improving their modelling experience. While the practitioners still mostly rely on gut feeling, experience and prototypes, having a roadmap for modelling in the form of a literature review would greatly improve their work, even in cases where they would not use it as a step-by-step guide, but more on a need-be basis.

This review has a number of implications for research and practice. For research, the review shows a clear need for more empirical studies of Event-B developments in order to (i) facilitate the take up of Event-B in industry, (ii) answer to industrial needs, and (iii) provide structured learning material for students. There is a clear indication that formal methods are indispensable in the development of safety-critical and software-intensive systems. However, its use should also fit the continuous integration and deployment

requirements. This opens challenges and possibilities for the Event-B ecosystem. Therefore, the research becomes more problem and industry-driven.

For practitioners, this review shows that many promising studies of the use of Event-B have already been reported. Naturally, for every approach some limitations have been identified in the future work sections of the selected papers. For instance, it is apparent that the learning curve for Event-B is steep and it is difficult to introduce Event-B into large and complex projects without having proper background. Moreover, implementation of the timing in Event-B is not straightforward and needs to be modelled separately.

The continuous integration and delivery concepts, which are included in the way the companies, also the safety-critical ones, work nowadays, are also a part of the research for Event-B. The ideas of having quicker delivery time through iterative development and strong tool support have recently been investigated to step up to the industrial needs, e.g. in the railway domain. Attempts have been made to scale up the proposed approaches, so that they are feasible in "real" industrial-size cases.

To increase the applicability and usefulness of the research for industry and to provide a sufficient number of studies of high quality on subjects related to Event-B development, the researchers should integrate more with industry to identify a common investigation agenda. The research already seems to be more problem-driven due to the collaboration with industry via various European and National projects. Moreover, the technology transfer already exists; however, it needs to be strengthened so that the practitioners are reinforced with up to date knowledge.

It has been emphasized many times that incremental development with small refinement steps, appropriate abstractions at each level and powerful tool support are all vital in Event-B developments.

7.2 Validity of the review

The main strength of this review is the holistic perspective it provides. It embraces modelling guidelines that are generic, as well as application- or domain-specific, both of which are meant for diverse experience levels – from beginners to more advanced modellers. The work contains studies from 2005 until the end of September 2019 and, thus, supports Event-B modellers with a state-of-the-art body of knowledge in the Event-B area. The review attempts to provide a complete view on the modelling with Event-B from the project perspective. It tackles the models and proofs, which can be treated as a product type of artefacts, but it also recognises the modelling and proving activities as parts of the development process, thus providing process-oriented guidelines.

As the Rodin Platform is constantly developing, there is a chance that our review will lack the studies regarding a plug-ins that are under development. Moreover, there is a risk that some plug-ins are not supported anymore. There is also a risk that a certain keyword was omitted in the search, due to the specifics of the vocabulary used in the Event-B area, and thus resulted in refuting some relevant literature. To help us ensure that the process of selection was unbiased, we went over the selection iteratively, first in a group of early-career researchers and then with a more experienced researcher. We used very careful inclusion criteria and eliminated some promising studies, which have not been evaluated thoroughly with a case study or an example, or have not been driven by industry. However, we included "Lessons Learned" and experience report papers to provide more guidelines to the practitioners.

Since some of the chosen works might have lacked sufficiently detailed description of context or the methods were not explained appropriately in terms of their validity (e.g. shallow discussion on validity of the approach included in a paper), there is a possibility that the collected guidelines also encapsulate some inaccuracy and will require fine-tuning when applied in different setting or domain. However, we did our utmost to ensure that the practitioners are provided with an as vast body of knowledge as possible, so that they are able to adapt the guidelines for their needs.

One of the limitations of this review is its scope that is restricted to the Event-B method (with a few exceptions of the B-Method). Such a decision is, however, well motivated because the paper is supposed to provide specialised guidelines for the Event-B modellers, which could not have been obtained if the review was more generic. Furthermore, we trust in the applicability of guidelines described given in the listed papers, since we have not applied all the methods and strategies ourselves.

8 Conclusions

This literature review provides a set of systematic guidelines for Event-B modelling, in order to improve the modelling experience of the Event-B practitioners. We divided the collection of guidelines according to the advancement level of the modellers and then according to their specific nature. The suggested improvements of the modelling experience are drawn from the product and process related artefacts. The review identifies a need for even more industry-driven research, so that the modelling guidelines can be enriched, tool-supported and possibly integrated with the development processes used.

8.1 Challenges and directions for future work

Among the challenges described in the papers that were used in this review, there were many that emphasised the need for further development and enrichment of suggested approaches. These mainly regard the library of common patterns, refinement patterns, general structures, constructs and components, all to increase the applicability of suggested methods. The scalability of suggested methods, developed models or templates, as well as the visual support for them has also been pointed out as issues to be further investigated. Moreover, the guidelines for validating the models were considered as one of the directions for future work.

Furthermore, it was highlighted that there is a need for more investigations on expressions of temporal properties and

the tool support for it, since timing is not explicitly included in the Event-B method. Moreover, the guidelines included in the reviewed papers emphasized the need for tool support for the presented approaches. The Event-B research community understands the need for tooling to facilitate the take up of Event-B in industry and enable collaboration with industrial partners.

Finally, there is an immense need for experimentation expressed in the majority of selected papers. It concerns either the application of the presented approaches and methods, or the synergies of existing approaches so that in the result the modelling is simplified and more flexible for an end user. This regards for instance the decomposition approaches so that the complexity of the model can be managed.

8.2 Measurement-guided modelling

Modelling can be supported not only by the patterns, decomposition strategies or enhancements in the development processes. It can also be facilitated by measurements and metrics, so that the modeller is well informed while creating and proving the model. In general, future work for the assessment methods for the formal approaches is quite vast and there are plenty of opportunities and challenges in this area. Event-B is no different in this respect.

There is a need for meaningful metrics and measurement models. Moreover, the desired values of existing metrics and their thresholds need to be established. More metrics supporting the modelling are vital to not only evaluate the current state of the model, but also to indicate potential problems and modelling trends. They could help with identification of best practices and feasible modelling strategies. These can be categorised for the metrics related to proving or the metrics related to the model itself. Note that some of these may be overlapping.

Modelling approaches should be further supported by metrics, by providing the modeller with the indicators of a "difficult to prove" model. For instance, an invariant which is complicated may indicate that there is a need for decomposing the superstate into more states. Additional complexity may come from having many states in a superstate (there should be a threshold set for an average number of states within a superstate) or including an orthogonal region in the model (many statemachines in one state that are working in parallel). Moreover, disjunctions are more complex and trickier to prove than conjunctions. All of these need to be mapped to the numerical system in order to establish usable and applicable metrics.

Assessment of difficulty of the proving activity and predicting the proving effort should be linked to the complexity of the model and the time spent on proving a model or a set of its properties. The latter can already now be tracked with the Rodin tool (the statistics window in the Rodin platform). For that purpose, the definition of the difficulty of proving needs to be established. It can be based either on the number of relations between the artefacts, e.g. function applications (in terms of iUML-B notation, for class diagrams and for invariants), which are difficult in comparison to the set-based approach in terms of proving. When it comes to human comprehension it is quite the opposite. Yet other artefacts that impact the difficulty in proving are the number of indirections running through the associations, the number of the associations and the number of loops in associations (similar to cycles in programming). Therefore, the metric assessing the difficulty in proving is intricate and needs to be carefully studied and validated.

For this to be at all possible, the factors that impact the complexity of the model need to be identified. Subsequently, it will be possible to create the metrics that evaluate this impact. The metrics can potentially be based on measurements related to, e.g., splitting guards to different conjuncts, which can already be complex by itself. Moreover, the nesting level of subsets could be used and computed as a ratio of leaf states per level of the hierarchy. This is similar to the code and design metric in traditional development called depth of inheritance tree (DIT metric), Furthermore, complexity of a model is impacted by the complexity of invariants, which depends for instance on the number of quantifications (especially the existential ones), sets and instances (dealing with the "for all" statement), as well as the number of implies in clauses. Such metrics will lead to a more feasible modelling experience and would be a useful tool together with the collection of guidelines presented here.

Acknowledgement

This review was undertaken within the project ADVICeS funded by the Academy of Finland (grant No. 266373, https://research.it.abo.fi/ADVICeS/index.html). We would like to thank Dr Colin Snook for the fruitful discussion on metrics for Event-B and Dr Thai Son Hoang and Dr Alexei Iliasov for providing information regarding the Rodin plug-ins. We also thank the anonymous reviewers for their inputs and suggestions.

References

Abrial, J.-R. (2010). *Modeling in Event-B: System and Software Engineering*, 1st edn, Cambridge University Press, New York, USA.

Abrial, J.-R. (2018). On B and Event-B: Principles, success and challenges, *International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z*, Springer, pp. 31–35.

ADVANCE Project (2011-2014). Advanced Model Development and Validation for Improved Analysis of Costs and Impacts of Mitigation Policies. [Online; accessed 16-October-2019].
URL: *http://www.advance-ict.eu/*

ADVICeS Project (2013-2017). Adaptive Integrated Formal Design of Safety-Critical Systems, Åbo Akademi University. [Online; accessed 16-October-2019].
URL: *https://research.it.abo.fi/ADVICeS/index.html*

Ahmad, M., Belloir, N. and Bruel, J.-M. (2015). Modeling and verification of functional and non-functional requirements of ambient self-adaptive systems, *Journal of Systems and Software* **107**: 50–70.

Almeida, J. B., Frade, M. J., Pinto, J. S. and Melo de Sousa, S. (2011). An Overview of Formal Methods Tools and Techniques, *Rigorous Software Development: An Introduction to Program Verification*, Springer London, London, pp. 15–44.

Ambler, S. and Lines, M. (2012). *Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise*, IBM Press.

Atelier B (2016). [Online; accessed 16-October-2019]. **URL:** *http://www.atelierb.eu*

Back, R.-J. (1978). *On the Correctness of Refinement Steps in Program Development*, PhD thesis, University of Helsinki, Finland.

Back, R. J. R. (1990). Refinement calculus, Part II: Parallel and reactive programs, *in* J. W. de Bakker, W. P. de Roever and G. Rozenberg (eds), *Proceedings of the REX Workshop on Stepwise Refinement of Distributed Systems Models, Formalisms, Correctness, The Netherlands May 29 – June 2, 1989*, Springer, Berlin, Heidelberg, pp. 67–93.

Back, R.-J. and Sere, K. (1994). From action systems to modular systems, *International Symposium of Formal Methods Europe*, Springer, pp. 1–25.

Back, R.-J. and von Wright, J. (1998). *Refinement Calculus: A Systematic Introduction*, Springer Heidelberg, Graduate Texts in Computer Science.

BMotionWeb (2018). Heinrich-Heine-University. [Online; accessed 16-October-2019]. **URL:** *https://www3.hhu.de/stups/prob/index.php/BMotion_Studio*

Bowen, J. P. and Hinchey, M. (2012). Ten Commandments of Formal Methods... Ten Years On, *in* M. Hinchey and L. Coyle (eds), *Conquering Complexity*, Springer London, London, pp. 237–251.

Brama (2011). Model B animation tool, Clearsy System Engineering. [Online; accessed 16-October-2019]. **URL:** *http://www.brama.fr*

Butler, M. (2009). Decomposition Structures for Event-B, *International Conference on Integrated Formal Methods (iFM 2009)*, Vol. 5423 of *LNCS*, Springer, Düsseldorf, Germany, pp. 20–38.

Butler, M. and Maamria, I. (2013). Practical theory extension in Event-B, *Theories of Programming and Formal Methods* p. 67.

Cansell, D., Méry, D. and Rehm, J. (2007). Time Constraint Patterns for Event-B Development, *in* J. Julliand and O. Kouchnarenko (eds), *Proceedings of the* 7^{th} *International Conference of B Users (B 2007)*, Springer Berlin Heidelberg, pp. 140–154.

Code Generation Activity (2018). University of Southampton. [Online; accessed 16-October-2019]. **URL:** *http://www.prime-project.org/prime-code-generation-tool/*

Cofer, D. D., Gacek, A., Miller, S. P., Whalen, M. W., LaValley, B. and Sha, L. (2012). Compositional verification of architectural models, *in* A. E. Goodloe and S. Person (eds), *Proceedings of the* 4^{th} *NASA Formal Methods Symposium (NFM 2012)*, Vol. 7226, Springer-Verlag, Berlin, Heidelberg, pp. 126–140.

Cohn, M. (2004). *User stories applied: For agile software development*, Addison-Wesley Professional.

Comptier, M., Leuschel, M., Mejia, L.-F., Perez, J. M. and Mutz, M. (2019). Property-Based Modelling and Validation of a CBTC Zone Controller in Event-B, *International Conference on Reliability, Safety, and Security of Railway Systems*, Springer, pp. 202–212.

Dalvandi, M., Fathabadi, A. S. and Butler, M. (2018). Using formal methods for automatic platform-independent code generation of run-time management, *University Booth at DATE 2018, Dresden, Germany. 19 - 22 Mar 2018*.

Déharbe, D., Fontaine, P., Guyot, Y. and Voisin, L. (2014). Integrating SMT solvers in Rodin, *Science of Computer Programming* **94**: 130–143.

DEPLOY Project (2008-2012). Industrial deployment of system engineering methods providing high dependability and productivity. [Online; accessed 16-October-2019]. **URL:** *http://www.deploy-project.eu*

Dieumegard, A., Ge, N. and Jenn, E. (2017). Event-B at work: some lessons learnt from an application to a robot anti-collision function, *NASA Formal Methods Symposium*, Springer, pp. 327–341.

Dinca, I., Ipate, F., Mierla, L. and Stefanescu, A. (2012). *Learn and Test for Event-B – A Rodin Plugin*, Springer, Berlin, Heidelberg, pp. 361–364.

Dupont, G., Aït-Ameur, Y., Pantel, M. and Singh, N. K. (2018). Proof-Based Approach to Hybrid Systems Development: Dynamic Logic and Event-B, *International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z*, Springer, pp. 155–170.

Edmunds, A., Olszewska, M. and Waldén, M. (2016). Using the Event-B Formal Method for Disciplined Agile Delivery of Safety-critical Systems, *in* H. Kaindl and R. Meli (eds), *SOFTENG 2016: The* 2^{nd} *International Conference on Advances and Trends in Software Engineering*, IARIA, p. 1–9.

Edmunds, A., Snook, C. and Waldén, M. (2016). On Component-Based Reuse for Event-B, *in* M. Butler, K.-D. Schewe, A. Mashkoor and M. Biro (eds), *Proceedings of the* 5^{th} *International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z, ABZ 2016*, Springer International Publishing, Cham, pp. 151–166.

ENABLE S3 Project (2016-2019). Testing and Validation of Highly Automated Systems. [Online; accessed 16-October-2019].
URL: *https://www.enable-s3.eu/*

Eschbach, R. (2019). Industrial application of Event-B to a wayside train monitoring system: Formal conceptual data analysis, *International Symposium on Formal Methods*, Springer, pp. 738–745.

Event-B (2001). [Online; accessed 16-October-2019].
URL: *http://www.event-b.org/index.html*

Fathabadi, A. S., Butler, M. J., Yang, S., Maeda-Nunez, L. A., Bantock, J., Al-Hashimi, B. M. and Merrett, G. V. (2018). A model-based framework for software portability and verification in embedded power management systems, *Journal of Systems Architecture* **82**: 12–23.

Fathabadi, A. S., Butler, M. and Rezazadeh, A. (2012). A Systematic Approach to Atomicity Decomposition in Event-B, *in* G. Eleftherakis, M. Hinchey and M. Holcombe (eds), *Software Engineering and Formal Methods - 10^{th} International Conference, SEFM 2012, Thessaloniki, Greece, October 1-5, 2012. Proceedings*, Vol. 7504 of *Lecture Notes in Computer Science*, Springer, pp. 78–93.

Fotso, S. J. T., Mammar, A., Laleau, R. and Frappier, M. (2018). Event-B Expression and Verification of Translation Rules Between SysML/KAOS Domain Models and B System Specifications, *International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z*, Springer, pp. 55–70.

Gmehlich, R., Grau, K., Hallerstede, S., Leuschel, M., Lösch, F. and Plagge, D. (2011). On Fitting a Formal Method into Practice, *Proceedings of the 13^{th} International Conference on Formal Engineering Methods, ICFEM 2011, Durham, UK, October 26-28, 2011*, pp. 195–210.

Gnaho, C., Semmak, F. and Laleau, R. (2013). Modeling the impact of non-functional requirements on functional requirements, *International Conference on Conceptual Modeling*, Springer, pp. 59–67.

Hallerstede, S., Jastram, M. and Ladenberger, L. (2014). A Method and Tool for Tracing Requirements into Specifications, *Science of Computer Programming* **82**: 2–21.

Hoang, T. S., Dghaym, D., Snook, C. and Butler, M. (2017). A composition mechanism for refinement-based methods, *22^{nd} International Conference on Engineering of Complex Computer Systems (ICECCS 2017)*.

Hoang, T. S., Fürst, A. and Abrial, J.-R. (2013). Event-B patterns and their tool support, *Software & Systems Modeling* **12**(2): 229–244.

Hoang, T. S., Iliasov, A., Silva, R. A. and Wei, W. (2011). A survey on Event-B decomposition, *11th International Workshop on Automated Verification of Critical Systems (AVoCS 2011)*.

Hoang, T. S., Snook, C., Ladenberger, L. and Butler, M. (2016). Validating the Requirements and Design of a Hemodialysis Machine Using iUML-B, BMotion Studio, and Co-Simulation, *in* M. Butler, K.-D. Schewe, A. Mashkoor and M. Biro (eds), *Proceedings of the 5^{th} International Conference: Abstract State Machines, Alloy, B, TLA, VDM, and Z, ABZ 2016, Linz, Austria, May 23-27, 2016*, Springer International Publishing, Cham, pp. 360–375.

Iliasov, A., Troubitsyna, E., Laibinis, L., Romanovsky, A., Varpaaniemi, K., Ilić, D. and Latvala, T. (2010). Supporting Reuse in Event-B Development: Modularisation Approach, *in* M. Frappier, U. Glässer, S. Khurshid, R. Laleau and S. Reeves (eds), *Proceedings of the 2^{nd} International Conference: Abstract State Machines, Alloy, B and Z, ABZ 2010, Orford, QC, Canada, February 22-25*, Springer Berlin Heidelberg, pp. 174–188.

Jackson, D. (1996). Lightweight formal methods, *IEEE Comput.* **29**(4): 21–22.

Jackson, M. (2001). *Problem frames: analysing and structuring software development problems*, Addison-Wesley.

Jacquot, J.-P. and Mashkoor, A. (2018). The role of validation in refinement-based formal software development, Models: Concept, Theory, Logic, Reasoning, and Semantics, College Publications.

Jastram, M. and Butler, P. M. (2014). *Rodin User's Handbook: Covers Rodin V.2.8*, CreateSpace Independent Publishing Platform, USA.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J. and Linkman, S. (2009). Systematic literature reviews in software engineering–a systematic literature review, *Information and software technology* **51**(1): 7–15.

Kitchenham, B. and Charters, S. (2007). Guidelines for performing systematic literature reviews in software engineering.

Kobayashi, T., Ishikawa, F. and Honiden, S. (2014). Understanding and planning Event-B refinement through primitive rationales, *in* Y. Ait Ameur and K.-D. Schewe (eds), *Proceedings of the 4^{th} International Conference: Abstract State Machines, Alloy, B, TLA, VDM, and Z (ABZ 2014), Toulouse, France*, Springer Berlin Heidelberg, pp. 277–283.

Krings, S., Bendisposto, J. and Leuschel, M. (2015). From failure to proof: the ProB disprover for B and Event-B, *SEFM 2015 Collocated Workshops*, Springer, pp. 199–214.

Ladenberger, L., Bendisposto, J. and Leuschel, M. (2009). Visualising Event-B Models with B-Motion Studio, *in* M. Alpuente, B. Cook and C. Joubert (eds), *Proceedings of the 14^{th} International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2009), Eindhoven, The Netherlands*, Springer Berlin Heidelberg, pp. 202–204.

Lanoix, A. (2008). Event-B Specification of a Situated Multi-Agent System: Study of a Platoon of Vehicles, *2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering*, pp. 297–304.

Lecomte, T., Deharbe, D., Prun, E. and Mottin, E. (2017). Applying a formal method in industry: a 25-year trajectory, *Brazilian Symposium on Formal Methods*, Springer, pp. 70–87.

Leuschel, M. and Butler, M. (2003). ProB: A model checker for B, *International Symposium of Formal Methods Europe*, Springer, pp. 855–874.

Leuschel, M. and Butler, M. (2008). ProB: an automated analysis toolset for the B method, *International Journal on Software Tools for Technology Transfer* **10**(2): 185–203.

Lopatkin, I., Iliasov, A., Romanovsky, A., Prokhorova, Y. and Troubitsyna, E. (2011). Patterns for representing FMEA in formal specification of control systems, *2011 IEEE 13th International Symposium on High-Assurance Systems Engineering*, IEEE, pp. 146–151.

Mammar, A. and Laleau, R. (2016). On the use of domain and system knowledge modeling in goal-based Event-B specifications, *International Symposium on Leveraging Applications of Formal Methods*, Springer, pp. 325–339.

Mammar, A. and Laleau, R. (2017). Modeling a landing gear system in Event-B, *International Journal on Software Tools for Technology Transfer* **19**(2): 167–186.

Mashkoor, A. and Jacquot, J. P. (2011). Guidelines for Formal Domain Modeling in Event-B, *2011 IEEE* 13^{th} *International Symposium on High-Assurance Systems Engineering*, pp. 138–145.

Mashkoor, A., Yang, F. and Jacquot, J.-P. (2017). Refinement-based validation of Event-B specifications, *Software & Systems Modeling* **16**(3): 789–808.

Méry, D., Sawant, R. and Tarasyuk, A. (2015). Integrating Domain-Based Features into Event-B: A Nose Gear Velocity Case Study, *Proceedings of the 5th International Conference on Model and Data Engineering (MEDI) - Volume 9344*, Springer-Verlag New York, Inc., pp. 89–102.

Olszewska, M. (2011). *On the Impact of Rigorous Approaches on the Quality of Development*, PhD thesis, Turku Centre for Computer Science, Turku, Finland.

Olszewska, M., Ostroumov, S. and Waldén, M. (2016). Using Scrum to Develop a Formal Model – An Experience Report, *in* P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki and T. Mikkonen (eds), *Proceedings of the* 17^{th} *International Conference on Product-Focused Software Process Improvement (PROFES 2016), Trondheim, Norway, November 22-24, 2016*, Springer International Publishing, Cham, pp. 621–626.

Olszewska, M. and Waldén, M. (2015). DevOps Meets Formal Modelling in High-criticality Complex Systems, *Proceedings of the* 1^{st} *International Workshop on Quality-Aware DevOps*, QUDOS 2015, ACM, New York, USA, pp. 7–12.

Ostroumov, S. and Waldén, M. (2017). Visual Component-Based Development of Formal Models, *in* M. Kajko-Mattsson, P. Ellingsen and P. Maresca (eds), *The* 3^{rd} *International Conference on Advances and Trends in Software Engineering (SoftEng)*, IARIA, p. 43–50.

Pląska, M., Waldén, M. and Snook, C. (2007). Documenting the progress of the system development, *in* M. Butler, C. Jones, A. Romanovsky and E. Troubitsyna (eds), *Workshop on Methods, Models and Tools for Fault Tolerance - Proceedings*, pp. 118–127.

Ponsard, C. and Devroey, X. (2011). Generating High-Level Event-B System Models from KAOS Requirements Models, *INFormatique des ORganisations et Systèmes d'Information et de Décision (INFORSID)*.

ProB (2018). The ProB Animator and Model Checker, Heinrich-Heine-University. [Online; accessed 16-October-2019].
URL: *https://www3.hhu.de/stups/prob*

Prokhorova, Y. (2015). *Rigorous Development of Safety-Critical Systems*, PhD thesis, Turku Centre for Computer Science, Turku, Finland.

Prokhorova, Y., Laibinis, L. and Troubitsyna, E. (2015). Facilitating Construction of Safety Cases from Formal Models in Event-B, *Information and Software Technology* **60**: 51–76.

Prokhorova, Y. and Troubitsyna, E. (2012). Linking Modelling in Event-B with Safety Cases, *in* P. Avgeriou (ed.), *Proceedings of the* 4^{th} *International Workshop on Software Engineering for Resilient Systems (SERENE 2012)*, Vol. 7527 of *Lecture Notes in Computer Science*, Springer-Verlag Berlin Heidelberg, p. 47–62.

ProR tool (2014). [Online; accessed 16-October-2019].
URL: *https://www.eclipse.org/rmf/pror/*

Rezazadeh, A. and Butler, M. (2005). Some Guidelines for Formal Development of Web-Based Applications in B-Method, *in* H. Treharne, S. King, M. Henson and S. Schneider (eds), *Proceedings of the* 4^{th} *International Conference of B and Z Users (ZB 2005): Formal Specification and Development in Z and B, Guildford, UK, April 13-15, 2005*, Springer Berlin Heidelberg, pp. 472–492.

Ricketts, M. (2015). Using stories to teach safety: Practical, research-based tips, *Professional safety, American Society of Safety Engineers* **60**(5): 51.

Rodin Platform (2006). [Online; accessed 16-October-2019].
URL: *http://www.event-b.org/platform.html*

RODIN Project (2004-2007). Rigorous Open Development Environment for Complex Systems. [Online; accessed 16-October-2019].
URL: *http://rodin.cs.ncl.ac.uk/*

Rodriguez, M. T. (2013). *Invariant discovery and refinement plans for formal modelling in Event-B*, PhD thesis, Heriot-Watt University, Edinburgh, Scotland.

Romanovsky, A. and Thomas, M. (2013). *Industrial Deployment of System Engineering Methods*, Springer Publishing Company, Incorporated.

Said, M. Y., Butler, M. and Snook, C. (2015). A method of refinement in UML-B, *Software & Systems Modeling* **14**(4): 1557–1580.

Sato, N. and Ishikawa, F. (2015). Separation of Considerations in Event-B Refinement toward Industrial Use, *Proceedings of the First Workshop on Formal Methods in Software Engineering Education and Training, FMSEE&T 2015, co-located with* 20^{th} *International Symposium on Formal Methods (FM 2015), Oslo, Norway, June 23, 2015.*, pp. 43–50.

Savicks, V., Butler, M. and Colley, J. (2014). Co-simulation Environment for Rodin: Landing Gear Case Study, *in* F. Boniol, V. Wiels, Y. Ait Ameur and K.-D. Schewe (eds), *Proceedings of the* 4^{th} *International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z (ABZ 2014): The Landing Gear Case Study: Case Study Track, Toulouse, France, June 2-6, 2014*, Springer International Publishing, Cham, pp. 148–153.

Servat, T. (2007). Brama: A new graphic animation tool for B models, *International Conference of B Users*, Springer, pp. 274–276.

Silva, R. and Butler, M. (2009). Supporting Reuse of Event-B Developments through Generic Instantiation, *in* K. Breitman and A. Cavalcanti (eds), *Formal Methods and Software Engineering:* 11^{th} *International Conference on Formal Engineering Methods ICFEM 2009, Rio de Janeiro, Brazil, December 9-12, 2009. Proceedings*, Springer Berlin Heidelberg, pp. 466–484.

Siqueira, F. L., de Sousa, T. C. and Silva, P. S. M. (2017). Using BDD and SBVR to refine business goals into an Event-B model: a research idea, *Proceedings of the 5th International FME Workshop on Formal Methods in Software Engineering*, IEEE Press, pp. 31–36.

Snook, C. and Butler, M. (2008). UML-B and Event-B: an integration of languages and tools, *The IASTED International Conference on Software Engineering - SE2008*.

Snook, C., Hoang, T. S. and Butler, M. (2017). Analysing Security Protocols Using Refinement in iUML-B, *in* C. Barrett, M. Davies and T. Kahsai (eds), *Proceedings of the* 9^{th} *NASA Formal Methods International Symposium (NFM 2017), Moffett Field, CA, USA, May 16-18, 2017,*, Springer International Publishing, Cham, pp. 84–98.

Su, W. and Abrial, J.-R. (2017). Aircraft landing gear system: approaches with Event-B to the modeling of an industrial system, *International Journal on Software Tools for Technology Transfer* **19**(2): 141–166.

Su, W., Abrial, J.-R. and Zhu, H. (2014). Formalizing Hybrid Systems with Event-B and the Rodin Platform, *Science of Computer Programming* **94**: 164–202.

Wohlin, C. (2014). Guidelines for snowballing in systematic literature studies and a replication in software engineering, *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, Citeseer, p. 38.

Wolff, S. (2012). Scrum goes formal: Agile methods for safety-critical systems, *Proceedings of the First International Workshop on Formal Methods in Software Engineering: Rigorous and Agile Approaches*, IEEE Press, pp. 23–29.

Yang, F., Jacquot, J.-P. and Souquieres, J. (2013). Jeb: safe simulation of event-b models in javascript, *20th Asia-Pacific Software Engineering Conference (APSEC), 2013*, Vol. 1, IEEE, pp. 571–576.

Yeganefard, S., Butler, M. and Rezazadeh, A. (2010). Evaluation of a guideline by formal modelling of cruise control system in Event-B, *Proceedings of the Second NASA Formal Methods Symposium (NFM 2010), NASA/CP-2010-216215*, pp. 182–191.