TUCS

Marta Olszewska | Fatima Shokri-Manninen | Andrew Edmunds | Marina Waldén

"Improving the Modelling Experience in Event-B – A Literature Review"

Turku Centre *for* Computer Science

TUCS Technical Report
No 1196, June 2018

"Improving the Modelling Experience in Event-B – A Literature Review”

Marta Olszewska
Åbo Akademi University, Faculty of Science and Engineering,
Agora building, 3rd floor, Vesilinnantie 3, 20500 Turku, Finland
mplaska@abo.fi
Fatima Shokri-Manninen
Åbo Akademi University, Faculty of Science and Engineering,
Agora building, 3rd floor, Vesilinnantie 3, 20500 Turku, Finland
fshokri@abo.fi
Andrew Edmunds
Åbo Akademi University, Faculty of Science and Engineering,
Agora building, 3rd floor, Vesilinnantie 3, 20500 Turku, Finland
andy.ed.edmunds@gmail.com
Marina Waldén
Åbo Akademi University, Faculty of Science and Engineering,
Agora building, 3rd floor, Vesilinnantie 3, 20500 Turku, Finland
mwalden@abo.fi

TUCS Technical Report
No 1196, June 2018

Abstract

Developing safety-critical or software-intensive systems is an intricate task, since it involves 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 the body of knowledge that would support the Event-B practitioners with modelling guidelines. We first define the domains in which the guidelines fall into and divide them into two categories: the beginners and expert. Then we provide guidelines as reinforcements for the generic application of Event-B and its domain-specific use. Our goal is to increase the modelling experience of Event-B practitioners with guidelines that are easily accessible and practical.

Keywords: Event-B, Formal modelling, Guidelines, Systematic literature review, Multi-vocal literature review, Practitioners

TUCS Laboratory
Distributed Systems laboratory

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 [60]. Formal methods are mathematical techniques for specification, development and verification of software and hardware systems. They can be a foundation 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 provide a correct by construction system development, so that the system functions as required [13]. 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 accurately describe their properties, and thus gives confidence of 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 [23].

Modelling in Event-B provides many benefits, as obtaining a correct-by-construction system, control over the specified requirements, as well as sound management of the early stage development process [9]. However, it is considered as a challenging and time-consuming task, in particular for the beginners, since it involves a steep learning curve. The difficulties with creating the initial model in Event-B and then refining it according to certain rules are balanced out by the certainty that the model is proven correct, based on a rigorous mathematical basis according to given requirements and specified assumptions, once the skill is mastered [87]. 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 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 [9]. 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 B-Method). We use the systematic literature review method [50] and multi-vocal literature review method [36] to obtain a body of knowledge that would serve to improve Event-B modelling experience. While the former method bases the evidence on the published and refereed papers, the latter allows us to collect other types of knowledge sources, such as blogs, fora and white papers. The guidelines we collect are meant for the Event-B users, who barely begun to apply this particular formal method, as well as the 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 [48], 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 at the literature and digital sources, 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 [10] (acronym from Adaptive Integrated Formal Design of Safety-Critical Systems, 2013-2017) funded by Academy of Finland. 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 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, it 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 would provide additional guidelines for the modelling process. Furthermore, they would 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 follows the description of the context of our study, including the research questions and data sources, as well as the data collection procedure and data synthesis process. Section 4 and 5 answer the research questions. We continue to discuss our findings and validity issues in Section 6 and conclude in Section 7 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 [68]. We chose to focus on Event-B, as it is not only investigated in the project ADVICeS [10], but also because we have long experience with using this method for development of safety-critical systems, including distributed and reactive systems.

Event-B is derived from the B-Method, which is 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 compared to classical B, since adding new events, merging or splitting events are allowed. Moreover, model decomposition was introduced in Event-B to reduce the complexity of modelling big systems by dividing them into smaller submodels [68].

Event-B uses refinement [17] [18], which enables the system to be created in a stepwise fashion, starting from an abstract model, where each refinement step preserves the correctness of the previous ones [15] [16]. 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 refinement 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 sustaining the refinement process are crucial for the system development.

Event-B is well supported by the Rodin platform [2] [3], which is an Eclipse-based, open-source, rich client platform, extendable with plugins. A diversity of plug-ins is available to provide a more flexible and adaptable environment for Event-B development, as, for instance, ProB and the model decomposition plug-in [79], which include both a separate animator and an Eclipse plugin for animating and model checking of B and Event-B models. Property violations in the model can be detected by finding a counterexample. Furthermore, ProB can be used in automated refinement checking. The refinement error detection in ProB discovers inconsistent behaviours. A model decomposition plug-in is available for breaking a model into sub-models in Event-B. Two approaches are available: shared variable decomposition and shared event decomposition. The user can optimise the decomposition style, depending on the preference and the system [68].

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-functioning 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 the contemporary development processes and practices, at the same time scaling up to the needs of industry. In fact, Event-B has been gaining recognition in an industrial setting. Currently, it is one of the methods used in the European project ENABLE S3, which is an industry-driven undertaking that aims to provide more advanced and efficient methods to commercialise highly automated cyber physical systems (ACPS).

3 Context and review method

The goal of this paper is to review existing approaches and 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 three research questions:

- **RQ1:** How to effectively support the developers using Event-B both on the beginners and expert levels?
- **RQ2** What generic modelling guidelines have been advised to support the construction of Event-B models?
- **RQ3** What system-specific modelling guidelines have been proposed on how to specify, refine and verify the Event-B models?

The main goal of RQ1 is to first categorise the Event-B modelling guidelines and then to find 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 expert levels, where the beginners-level is merely a subset of the guidelines anticipated for the experts. Our reasoning is to provide the beginners with the get go knowledge and not overwhelm them with modelling nuances to (i) introduce them to the Event-B modelling in a step-wise manner (steep learning curve of formal methods) and to (ii) sustain their motivation.

The latter research questions (RQ2 and RQ3) are meant to deepen the knowledge with respect to the modelling area, type of the modelled system or application domain. This approach uses the findings of RQ1 (categorization of guidelines) to provide more detailed knowledge with respect to the chosen paper. While RQ1 gives a horizontal knowledge on the modelling with Event-B, RQ2 and RQ3 present a vertical, step by step approach and show the issues that need to be addressed when modelling and proving a system.

3.1 Data sources and search strategy, data collection

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 query: Event-B AND (guidelines OR tactics OR strategy OR advice OR advices OR ‘modelling guidelines’).

Data collection We collected the data for the articles that were published until May 2017. We searched for modelling guidelines for Event-B in books, journals and conference proceedings, which are listed as follows:

Books *(*also theses and technical reports): Addison-Wesley Publishing, University of Southampton Technical Report [UK], Turku Centre for Computer Science Technical Report [FI]; University of Applied Sciences and Arts and Zurich University of Applied Sciences (FHNW and ZHAW) [CH];

Journals: Formal Aspects of Computing; International Journal of Software and Informatics, Electronic Communications of the European Association of Software Science and Technology, Science of Computer Programming;

Conferences: International Conference on Information Science and Applications (ICISA), IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering (TASE), International Symposium on High-Assurance Systems Engineering (HASE), NASA Formal Methods Symposium (NFM), International Conference of B and Z Users (ZB) later Abstract State Machines, Alloy, B and Z or Abstract State Machines, Alloy, B, TLA, VDM, and Z (ABZ), International Conference of B Users (B2007), International Conference on Advances and Trends in Software Engineering (SOFTENG), Formal Methods in Software Engineering Education and Training (FM-SEE&T), Product-Focused Process Improvement Conference (PROFES).

It should be mentioned that for the guidelines related to the level of expertise of the modeller, we also used the grey literature [36]. The use of grey literature is motivated by the fact that often the "tips and tricks" used by the developers are not being a part of scientific publications, as being considered as too technical or too low level. Furthermore, the tools rarely have the scientific publications in their background and many sources, code or models are simply posted online. Lastly, the up-to-date information can be posted online on a blog, wiki or repository without the delay that is inevitable when publishing a research paper. The developers need all the support available to be able to effectively advance their skills, and thus our goal was to provide as extensive as possible body of knowledge.

Inclusion and Exclusion Criteria The main criterion for the choice of papers was the practicality of the suggested guidelines. The requirement for the papers that were included in this work was to 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 suggested approach sufficed. Yet another criterion was that the presented work was relatively recent and based on the approaches already well-developed and verified.

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 with-in the group of younger researchers and then iterated with the senior one for approval.

3.2 Data Synthesis

Data synthesis includes assembling and organising the results of the included primary studies. We selected a descriptive synthesis and display the information extracted from the primary studies in figures and tables. For the first research question (RQ1), we collected literature that would be suitable for two levels - beginner and expert.

For the generic and specific guidelines, RQ2 and RQ3 respectively, we relied on a standard systematic literature review. Our objective was to provide self-containing packages in form of paper references, concentrating on the application context or domain. We created a table with a list of 24 entries (publications) that can be used as a handbook, depending on the area of utilisation of formal methods (Event-B): domain specific or generic.

4 Guidelines

The guidelines are meant to facilitate the modelling process, either by providing the knowledge on 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, which 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 the 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 stand for particular methods or Event-B plugins that can enable the fulfilment of certain modelling demand or to facilitate the modelling process.

In this section, we address the RQ1, i.e., “How to effectively support the developers using Event-B both on the beginner and expert levels” We divide the guidelines with respect to the level of experience of potential users, e.g., the beginners and experts, respectively. To show only the most vital information in the plethora of literature on the subject, the guidelines for beginners are a limited subset of these for experts. They are enriched with the sources for knowledge acquisition that enable the smooth start with modelling using Event-B. In the following sections we describe the guidelines for the two target groups.

4.1 Beginner support

In Figure 1 we have divided the type of guidelines into categories with respect to the modelling challenges, such as knowledge acquisition, requirements specifica-

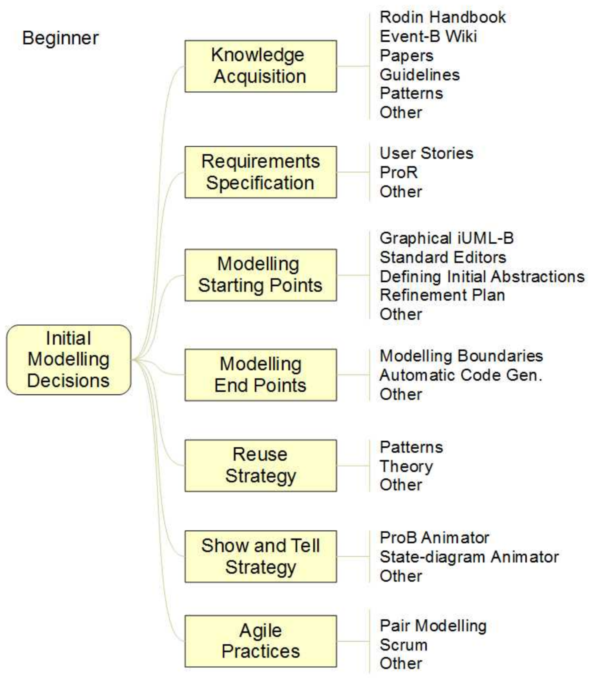

Figure 1: Guidelines for the beginners level

tion, modelling starting- and end-points, reuse strategy, show and tell strategy and agile practices. In this section we will describe them in detail, focusing on the means how 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.

The 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 [48] and Wiki for Event-B and Rodin [1]. Moreover, there is a noteworthy literature source available coming from three European projects RODIN (2004-2007) [2] and DEPLOY (2008-2012) [4] and ADVANCE (2011-2014) [5], as well as from the Academy of Finland funded project ADVICeS (2013-2017) [10]. Furthermore, the guidelines on refinement and building the models can be found in the literature by the founder(s) of the method, e.g., [7] [8] [9] [59]. In addition, learning about the usage of patterns [43], which are frequent in the modelling and proving (certain steps and strategies) will implement the best practices from the beginning. Finally, the tips and tricks from more experienced colleagues, who already maneuvered in the formal modelling world and know the modelling shortcuts are invaluable.

The requirements specification, which is the baseline for the formal modelling process, can be done in various ways from organized, quite detailed and formal listings, which is desired 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. Originating from XP [21], it provides various viewpoints on the feature to be implemented [11]. In the requirements specification category, we also included the tool support for requirements editing, called ProR [38] [39], which can be used as a plugin for the Rodin platform. The requirements can be elicited and processed with the support of the ProR tool. They are presented in a table format, where the relations between the requirements are enabled as annotated links.

There are many modelling starting points to consider, to begin with the choice of editor. They range from text-based editors, like standard Rodin or Camille text editors, Event-B editor, text editor or tree-like EMF editor, to graphical ones like iUML-B (and older version UML-B [79, 85]. 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 the experience, but at the beginning should be based on some already existing guidelines, like in Kobayashi et al. [51], Sato et al. [80] or Rodriguez [77].

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 the 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, there are various programming and domain specific languages to which Event-B models can be generated to, e.g., JML-specified Java, Java, C, C++, C# and VHDL [89] using diverse approaches. 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 [33].

The system under development is more likely to be implemented by the development team than one particular developer. Due to the collective knowledge, fewer assumptions need to be made. Therefore, it is beneficial for them to relatively early understand the 'limits' of the model. Invariably the abstraction of the environment contains assumptions and would be out of the developer's control. It would then be important to consider all the assumptions, and to consider how they might affect the integrity of the model.

Since reuse is considered as a good practice in coding and modelling, it is also available for Event-B. It can be achieved for instance with application of patterns or the so called theories enabled via Theory plugin [88] for reuse of proofs. The former is not only tool supported [34], but also some guidelines are given for the general purpose [43] or introducing timing patterns in Event-B [28]. The latter extend the Event-B language and the proving infrastructure in a fashion familiar to Rodin users. There are also transformation patterns available and supported by a plugin [55], which allows users to write transformation scripts in EOL (Epsilon Object Language – a simple object-based imperative language) and easily run them over models within the same project. The Theory extension provides a way to add new data types, operators, inference and rewrite rules, as well as code generation translation rules [26]. Theories can then be reused in different projects.

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. ProB Animator (constraint solver and model checker) [41] 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. It can be installed with Rodin, where it comes with BMotionStudio [40] to easily generate domain specific graphical visualizations. On the other hand, the State-diagram Animator [90] 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 the methodologies, like refinement [15] [17] [18], tools (the Rodin platform and the associated plugins), 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 two: pair modelling [21] [22] and Scrum [82] [83]. While the former is bringing the pair programming concepts to the formal modelling activities, e.g., having two modellers involved in modelling – one focusing on details of the model and the other on the bigger picture of the development. The latter process adapts Scrum-based management of product development to the modelling phase in terms of providing time frames for the modelling and prioritizing the requirements, for example.

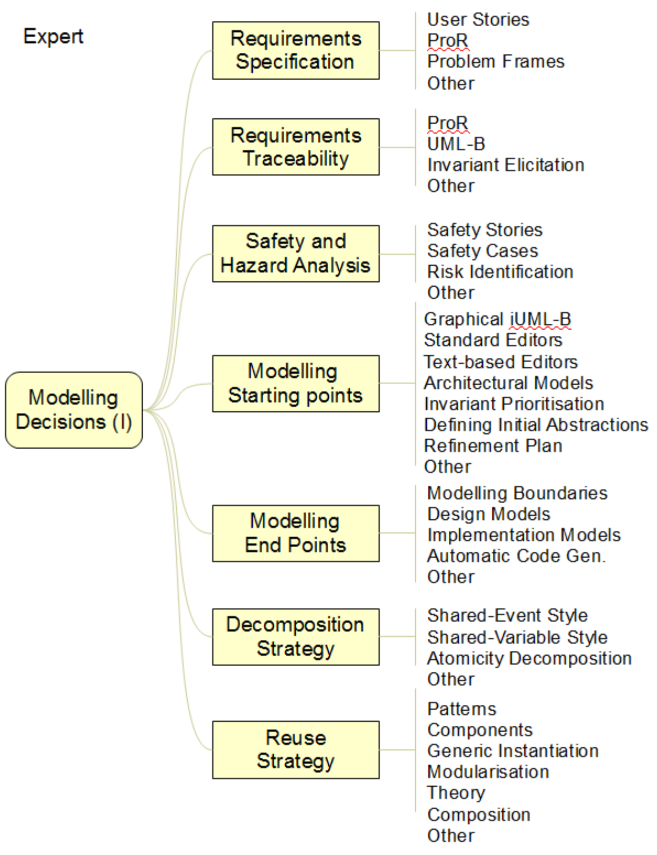

Figure 2: Guidelines for the expert level (part 1)

4.2 Expert support

The developer at the expert-level is more demanding when it comes to guidelines and might have more in-depth questions regarding development issues. Hence, the guidelines for the expert-level are more extensive than for the beginners and consider more alternatives for each category.

We widen the means to tackle requirements specification with Problem Frames [47], which is an approach to software requirements analysis to be used when gathering requirements and creating specifications for computer software. The relation of Problem Frames (informal representation) to Event-B models (formal representation) was investigated in [49]. A clear separation of the requirements, the environment and the system to be built was achieved. Moreover, it is possible to more precisely define the requirements and identify the missing ones [78]. This approach was acknowledged as one facilitating the early stage development and requirement elicitation in the automotive domain for developing a cruise control system [37].

As yet another approach to handling requirements, by following their evolution throughout the modelling process and over refinement steps, is the requirements traceability. The control over requirements is available for instance via the ProR and UML-B tools. Moreover, Event-B, and formal modelling in general, is considered to be useful for requirements engineering, in particular for invariant elicitation [77].

Since formal methods are most useful when applied in 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 it in a variety of ways, just to mention safety cases [73], safety stories [31] [76] or risk identification (risk assessment) [71]. The former is used for instance in the certification process and to justify why a system is safe and if the systems properly tackles safety requirements defined in a system requirement specification [72]. Safety stories, on the other hand, are given in an easy to understand narrative, which can be later modelled formally. The latter category of guidelines, risk identification, stands for fault tolerance techniques, such as Failure Modes and Effects Analysis (FMEA) [74] or Fault Tree Analysis (FTA) [71], which are most commonly used in combination with Event-B for rigorously examining the causes and the consequences of hazards.

Modelling starting points has been extended with more editors for modelling, e.g., the text-based editors, which may be appropriate for those, who are familiar with 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 [29]. Furthermore, invariant prioritization is needed to know which properties of the system are important (and more important than others) and when they should be introduced to the model (at what levels of the refinement chain) [77].

Modelling, as a stage of development and an activity, may have its “endpoint”, 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 the requirements engineering resulting in a concept called the design model [70]. 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.

The refinement of the system and how it evolves depends on the decomposition strategy [44], 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 [24] and shared variable style [8] 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 [35] 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 the composition, which is used in the bottom-up approach or modularity and reuse scenarios. Such composition mechanism for refinement based methods has been presented in [42], where both top-down and bottom-up approaches can be used. The work focuses on providing mechanisms for proving the correctness of machine inclusion.

While for the beginner-level we described patterns and theories, which contribute to the model reuse in the Event-B modelling, for the expert-level we enhance it with a concept of components, generic instantiation and modularization. Modelling with the use of components [32] [67] 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 plugin. 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 [84]. Furthermore, modularization 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 [44]. Finally, the composition mechanism for refinement-based methods supports the bottom-up approach and allows us to combine existing models [42].

There is a number of show and tell strategies that can support the modeller with his work except of ProB Animator and State-Diagram Animator. For instance, Anim-B [58], an animator for the Rodin platform, allows animation of complete model (all refinements) and can be used to create a complex animation with a graphical interface. The B-Motion Studio, on the other hand, also referred to as BMotionWeb [40] is a tool built on top of ProB for creating interactive visualisations of Event-B models [53]. Finally, there are other ways of visualizing models that can facilitate the Event-B development of, e.g., hydraulic systems

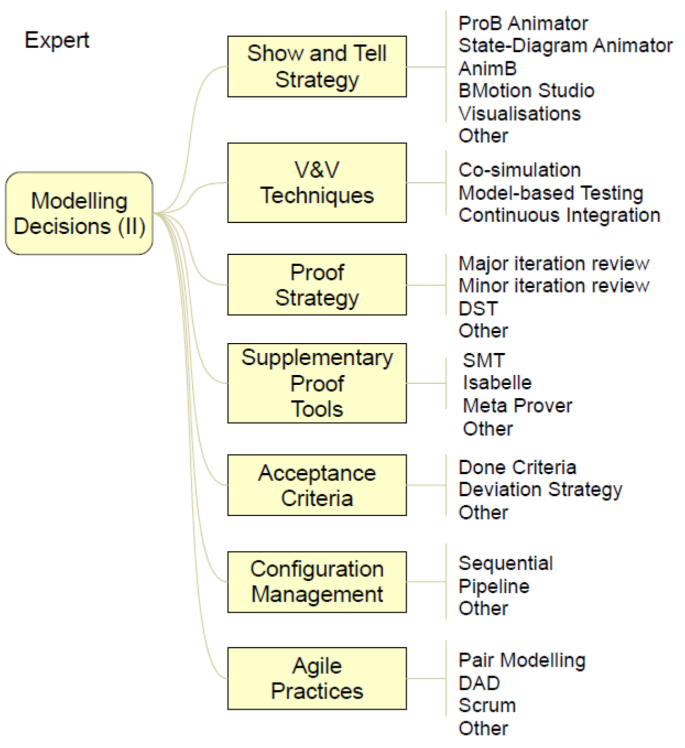

Figure 3: Guidelines for the expert level (part 2)

[67], or components in general [65] [66], or illustrating the refinement process itself [69].

On the expert level, there are broader verification and validation practices (V&V techniques) offered, such as co-simulation and model based testing. The former 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) [81]. 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. The latter, model-based testing, is supported in Event-B by a tool that generates test-cases from the Event-B models based on ProB [30]. It can be used as a follow up after the code generation, as the code generators may not be certified and thus not providing an appropriate level of confidence.

When building a formal model, not only the modelling strategies are 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. 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) [80], which helps to design how to prove the safety requirement in a natural language apart 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, for instance, SMT solvers (plugin to Rodin), Isabelle provers for the Rodin tool that prove proof obligations with Isabelle/HOL and export them to Isabelle/HOL theories, as well as to the Meta Prover [6].

When working with a formal model, one needs to define the acceptance criteria for the model. They can be either 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” [31] [61] [64]. As for other acceptance criteria, also the deviation strategy has to be identified. Deviation strategy is related to a scenario when acceptance criteria are not met. Some criteria will be considered to be more important to the project and be of higher priority than others and thus will need to be identified as, e.g., “show stoppers”.

For safety critical software, a notion of configuration management is used to maintain a balance between the stability and flexibility of the produced system [91]. It is tightly related to the standardization (ISO/IEC 24765 standard) and development process and releases of software within these processes. It concerns a discipline of applying technical and administrative directions and surveillance to e.g. verify compliance with specified requirements. In our setting we concentrate on two ways of working with the software configuration management: pipeline and sequential. The former stands for working on a number of release levels at the same time (as in parallel pipelines), while the latter focuses on providing one release at a time (in a sequential manner). It is particularly important in managing the process of continuous integration and deployment.

Agile practices, which can facilitate the development process, were enhanced for the expert-level with Disciplined Agile Delivery (DAD) framework [14]. It is a high-level agile framework, which embraces agile methods in a broader perspective. It can be referred as a pick-and-mix approach, which, apart from the process of actually building the system, tackles additional process goals, work on the requirements and planning 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 [31].

5 Generic and system-specific guidelines

For the purpose of our investigation on the generic and system-specific guidelines we did a systematic literature review and concentrated solely on the published scientific literature (disregarding the blogs and other digital web-content). The modelling guidelines presented in this section are given in form of tables, where the guidelines are ordered in terms of area, type of the modelled system or the application domain. The tables display the paper title with reference, as well as guidelines area and guidelines categories. The categories originate from the guideline categories for beginners and experts presented in Sections 4.1 and 4.2. At times, we also specify the detailed guideline (a leaf in the guidelines tree), when it is particularly important in the scope of the referenced paper. Among the selected papers are also works that explicitly tackle the extensions and adjustments to the development process. These are indicated in the table by (*). The main criterion for the choice of papers was the practicality of the suggested guidelines. 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 is relatively recent and based on the approaches already well-developed and verified.

5.1 Generic modelling guidelines

The modelling guidelines, apart from being classified according to the experience of the modeller, can also be referring to the specifics of a certain domain. In this section we focus on answering the second research question (RQ2), namely: "What generic modelling guidelines have been advised to support the construction of Event-B models"?

In Table 1, we list the papers that were selected, which contain the generic system guidelines. The papers support the analysis of requirements, traceability in the models, as well as decomposition strategies. All of these are related to the modelling activity as a part of the development process.

Table 1: Generic modelling guidelines

ID	Guidelines area	Paper title	Guidelines category
1	Requirements Analysis	Requirement Analysis for Event-B modelling [20]	Requirements specification
2	Traceability	Building Traceable Event-B Models from Requirements [12]	Requirements specification Modelling starting points Decomposition Show & Tell Strategy (Visualisation)
3	Decomposition	A Survey on Event-B Decomposition [44]	Modelling starting points Decomposition strategy (Shared-Variable Style, Shared-Event Style) Reuse strategy (Modularisation) V&V Techniques
4	Development Process*	Agile Success Factors – A Qualitative Study About What Makes Agile Projects Successful [52]	Agile Practices (Values and Principles)
5	Development Process*	FormAgi – A Concept for More Flexible Formal Developments [63]	Agile Practices Acceptance Criteria (Done Criteria)
6	Development Process*	Using the Event-B Formal Method for Disciplined Agile Delivery of Safety-Critical Systems [31]	Agile Practices (DAD) V&V Techniques (Model-Based Testing, Continuous Integration) Requirement specification Reuse Strategy (Patterns) Modelling starting points (Refinement plan)
7	Development Process*	Using Scrum to Develop a Formal Model – An Experience Report [62]	Agile Practices (Scrum) Requirement specification Acceptance Criteria (Done Criteria) Reuse Strategy (Components) Modelling starting points (Refinement Plan)

5.2 System- and domain-specific guidelines

The characteristics of the modelled system or the application domain might require some more specific modelling tactics or strategies. Therefore, in this section we aim to provide answers to the third research question (RQ3), i.e., what system-specific modelling guidelines have been proposed on how to specify, refine and verify Event-B models. The results are presented in Table 2.

Table 2: System-specific modelling guidelines

ID	Type of system (Domain)	Paper title	Guidelines category
1	Multi-Agent System (Communications and protocol interactions)	An Incremental Process for the Development of Multi-Agent Systems in Event-B [19]	Requirement Specification Modelling starting points Decomposition strategy Reuse strategy (Patterns) Visualisation
2	Multi-Agent System / Control System (Transportation)	Event-B Specification of a Situated Multi-Agent System: Study of a Platoon of Vehicles [54]	Modelling starting points Decomposition strategy
3	Multi-Agent System (Network Proto-cols)	Separation of Considerations in Event-B Refinement toward Industrial Use [80]	Requirement specification Modelling starting points (Refinement plan) Proof Strategy (DST)
4	Control System	Towards a cookbook for modelling and refinement of control problems [25]	Modelling starting points Decomposition strategy (Refinement plan)
5	Control System (Automotive)	Evaluation of a Guideline by Formal Modelling of Cruise Control System in Event-B [92]	Modelling starting points Decomposition strategy
6	Control System (Medical)	Validating the Requirements and Design of a Haemodialysis Machine Using iUML-B, BMotion Studio, and Co-Simulation [45]	Modelling starting points Show & Tell Strategy (Anim-B, BMotion Studio) V&V Techniques (Co-simulation, Model Checking)
7	Control System (Aerospace)	Visual Component-based Development of Formal Models [67]	Modelling starting points Reuse Strategy (Component) Show & Tell Strategy (Visualisation)
8	Control System (Aerospace)	Using Scrum to Develop a Formal Model – An Experience Report [62]	Agile Practices (Scrum) Requirement specification Acceptance Criteria (Done Criteria) Reuse Strategy (Components) Modelling starting points (Refinement Plan)
9	Embedded Software (Spacecraft)	Supporting Reuse in Event-B Development: Modularisation Approach [46]	Reuse strategy (Modularisation) Decomposition strategy (Shared-Variable Style) V&V Techniques

10	Time Constraint System (Firewire network protocol)	Time Constraint Patterns for Event-B Development [28]	Modelling starting points Reuse strategy (Patterns)
11	Web-Based Application (E-commerce)	Some Guidelines for Formal Development of Web-Based Applications in B-Method [75]	Modelling starting points
12	Network Protocols (VLAN Security)	Analysing Security Protocols Using Refinement in iUML-B [86]	Modelling starting points Proof strategy V&V Techniques Show & Tell Strategy (iUML-B, ProB)
13	Mondex Protocol (Banking)	An Incremental Development of the Mondex System in Event-B [27]	Modelling starting points Proof strategy
14	Mondex Protocol (Flash file system)	Invariant Discovery and Refinement Plans for Formal Modelling in Event-B [77]	Requirements traceability (Invariant elicitation) Modelling starting points (Invariant prioritisation, refinement plan)
15	Domain Modelling (Transportation)	Guidelines for formal domain modeling in Event-B [56]	Requirement Specification Modelling starting points Proof strategy Show & Tell Strategy Reuse strategy (Time Patterns)
16	Components (Transportation)	A composition mechanism for refinement-based methods [42]	Reuse Strategy (Inclusion) Decomposition Strategy (Composition)
17	Algorithmic Systems	Refinement-based guidelines for algorithmic systems [57]	Modelling start-points Reuse strategy (Patterns)

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

6 Discussion

As it can be seen from papers, 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 description of generic and specific guidelines, which would lead the potential modeller in obtaining comprehensive guidelines whenever needed. The present review provides that information and serves as a roadmap for Event-B beginners and experts.

6.1 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 a 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 as on the need-be basis.

This review also shows that the classical literature review method would not suffice to support the Event-B modellers with their tasks. Apart from the published and refereed material, which often lacks technical detail and is delayed with respect to the current state of practice, they need the grey literature support, containing the troubleshooting, fora, blogs and other knowledge sources.

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 the industrial needs and (iii) provide a structured learning material for the 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 was 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, the timing is not straightforward and needs to be modelled separately. Often, there are some tips and tricks reported so that the modelling and proving becomes easier. These are not clear-cut solutions, thus should be well documented.

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 are done 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 iterate 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 the up to date knowledge.

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

6.2 Validity of the review

The main strength of this review is the holistic perspective it provides. It embraces the modelling guidelines that are generic, as well as application- or domain- specific, both meant for diverse experience levels – beginners and experts. The work contains recent studies (until May 2017) and thus supports the Event-B modellers with the 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.

The major weaknesses of the review regard the selection of publications (or in case of grey literature, other sources) and imprecision in data extraction. There is 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. However, we included the "lessons-learnt" and experience report papers, so that we could provide more guidelines to the practitioners. To help us ensure that the process of selection was unbiased, we went over the selection iteratively, first in a group of younger researchers and then with a senior one. We used very careful inclusion criteria and potentially eliminated some promising studies, which have not been evaluated thoroughly with a case study or an example, or have not been driven by industry.

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 as vast as possible body of knowledge, 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 few exceptions of the B-Method). Such a decision is, however, well motivated – 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 described guidelines given in the enlisted papers, since we have not applied all the methods and strategies our-selves.

7 Conclusions

This literature review provides a set of guidelines for Event-B modelling, which are given in a systematic manner, in order to improve the modelling experience of the Event-B practitioners. We divided the guidelines according to the expertise level of the modellers and then according to their generic and 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.

7.1 Challenges and directions for future work

Among the challenges described in the papers we 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.

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).

References

[1] Event-B. `http://www.event-b.org/index.html` (2004–2017)

[2] RODIN Project – Rigorous Open Development Environment for Complex Systems. `http://rodin.cs.ncl.ac.uk/` (2004-2007)

[3] Rodin Platform. `http://www.event-b.org/platform.html` (2004-2017)

[4] DEPLOY Project – Industrial deployment of system engineering methods providing high dependability and productivity. `http://www.deploy-project.eu` (2007). [Online; accessed 19-December-2017]

[5] ADVANCE Project – Advanced Model Development and Validation for Improved Analysis of Costs and Impacts of Mitigation Policies. `http://www.advance-ict.eu/` (2017)

[6] Theory and proof plugins. `http://wiki.event-b.org/index.php/Rodin_Plug-ins#Theory_and_Proof` (2017)

[7] Abrial, J.R.: The B-book: Assigning Programs to Meanings. Cambridge University Press, New York, NY, USA (1996)

[8] Abrial, J.R.: Event model decomposition. `http://wiki.event-b.org/images/Event_Model_Decomposition-1.3.pdf` (2009)

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

[10] ADVICeS Project – Adaptive Integrated Formal Design of Safety-Critical Systems: Åbo Akademi University. `https://research.it.abo.fi/ADVICeS/index.html` (2013-2017). Accessed:18.12.2017

[11] Agile Alliance: User Stories. `https://www.agilealliance.org/glossary/user-stories` (2017)

[12] Alkhammash, E., Butler, M., Fathabadi, A.S., Círstea, C.: Building traceable Event-B models from requirements. Science of Computer Programming **111**(Part 2), 318 – 338 (2015). DOI https://doi.org/10.1016/j.scico.2015.06.002. Special Issue on Automated Verification of Critical Systems (AVoCS 2013)

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

[14] Ambler, S., Lines, M.: Disciplined Agile Delivery: A Practitioner's Guide to Agile Software Delivery in the Enterprise. IBM Press. IBM Press (2012). URL `https://books.google.fi/books?id=SLIkMhB2ew0C`

[15] Back, R.J.: On the correctness of refinement steps in program development. Ph.D. thesis, University of Helsinki, Finland (1978)

[16] Back, R.J., Sere, K.: From action systems to modular systems. In: International Symposium of Formal Methods Europe, pp. 1–25. Springer (1994)

[17] Back, R.J., von Wright, J.: Refinement Calculus: A Systematic Introduction. Springer Heidelberg, Graduate Texts in Computer Science (1998). DOI 10.1007/978-1-4612-1674-2

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

[19] Ball, E.: An incremental process for the development of multi-agent systems in Event-B. Ph.D. thesis, University of Southampton, UK (2008)

[20] Batjargal, B., Lee, K.H.: Requirement Analysis for Event-B Modeling. In: 2013 International Conference on Information Science and Applications (ICISA), pp. 1–3 (2013). DOI 10.1109/ICISA.2013.6579481

[21] Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2000)

[22] Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change (2^{nd} Edition). Addison-Wesley Professional (2004)

[23] Bowen, J.P., Hinchey, M.: Ten Commandments of Formal Methods... Ten Years On. In: M. Hinchey, L. Coyle (eds.) Conquering Complexity, pp. 237–251. Springer London, London (2012)

[24] Butler, M.: Decomposition Structures for Event-B. In: International Conference on Integrated Formal Methods, *LNCS*, vol. 5423, pp. 20–38. Springer, Düsseldorf, Germany (2009). DOI 10.1007/978-3-642-00255-7

[25] Butler, M.: Towards a cookbook for modelling and refinement of control problems. Tech. Rep. Working paper, University of Southampton, UK (2009)

[26] Butler, M., Maamria, I.: Practical Theory Extension in Event-B. In: Z. Liu, J. Woodcock, H. Zhu (eds.) Theories of Programming and Formal Methods, pp. 67–81. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). DOI 10.1007/978-3-642-39698-4_5

[27] Butler, M., Yadav, D.: An incremental development of the Mondex system in Event-B. Formal Aspects of Computing **20**(1), 61–77 (2008). DOI 10.1007/s00165-007-0061-4

[28] Cansell, D., Méry, D., Rehm, J.: Time Constraint Patterns for Event-B Development. In: J. Julliand, O. Kouchnarenko (eds.) Proceedings of the 7^{th} International Conference of B Users (B 2007), pp. 140–154. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). DOI 10.1007/11955757_13

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

[30] Dinca, I., Ipate, F., Mierla, L., Stefanescu, A.: Learn and Test for Event-B – A Rodin Plugin. In: J. Derrick, J. Fitzgerald, S. Gnesi, S. Khurshid, M. Leuschel, S. Reeves, E. Riccobene (eds.) Proceedings of the 3^{rd} International Conference on Abstract State Machines, Alloy, B, VDM, and Z, ABZ 2012, pp. 361–364. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). DOI 10.1007/978-3-642-30885-7_32

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

[32] Edmunds, A., Snook, C., Waldén, M.: On Component-Based Reuse for Event-B, pp. 151–166. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-33600-8_9

[33] Edmunds, A., Waldén, M.: Modelling 'Operation-Calls' in Event-B with Shared-Event Composition. In: L. Ribeiro, T. Lecomte (eds.) Proceedings of the 19^{th} Brazilian Symposium on Formal Methods: Foundations and Applications: SBMF 2016, Natal, Brazil, November 23-25, 2016, pp. 97–111. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-49815-7_6

[34] ETH Zurich: Pattern Plug-in. `http://wiki.event-b.org/index.php/Pattern` (2010)

[35] Fathabadi, A.S., Butler, M., Rezazadeh, A.: A Systematic Approach to Atomicity Decomposition in Event-B. In: G. Eleftherakis, M. Hinchey, M. Holcombe (eds.) Software Engineering and Formal Methods - 10^{th} International Conference, SEFM 2012, Thessaloniki, Greece, October 1-5, 2012. Proceedings, *Lecture Notes in Computer Science*, vol. 7504, pp. 78–93. Springer (2012). DOI 10.1007/978-3-642-33826-7_6

[36] Garousi, V., Felderer, M., Mäntylä, M.V.: The Need for Multivocal Literature Reviews in Software Engineering: Complementing Systematic Literature Reviews with Grey Literature. In: Proceedings of the 20^{th} International Conference on Evaluation and Assessment in Software Engineering, EASE '16, pp. 26:1–26:6. ACM, New York, NY, USA (2016). DOI 10.1145/2915970.2916008

[37] Gmehlich, R., Grau, K., Hallerstede, S., Leuschel, M., Lösch, F., Plagge, D.: On Fitting a Formal Method into Practice. In: Proceedings of the 13^{th} International Conference on Formal Engineering Methods, ICFEM 2011, Durham, UK, October 26-28, 2011, pp. 195–210 (2011). DOI 10.1007/978-3-642-24559-6_15

[38] Hallerstede, S., Jastram, M., Ladenberger, L.: A Method and Tool for Tracing Requirements into Specifications. Science of Computer Programming **82**, 2–21 (2014). DOI 10.1016/j.scico.2013.03.008

[39] Hallerstede, Stefan and Jastram, Michael and Ladenberge, Lukas: ProR Plugin. `http://wiki.event-b.org/index.php/ProR` (2013)

[40] Heinrich-Heine-University: BMotionWeb (earlier BMotionStudio). `https://www3.hhu.de/stups/prob/index.php/BMotion_Studio` (2017)

[41] Heinrich-Heine-University: The ProB Animator and Model Checker. `https://www3.hhu.de/stups/prob/index.php/Main_Page` (2017)

[42] Hoang, T.S., Dghaym, D., Snook, C., Butler, M.: A composition mechanism for refinement-based methods. In: 22^{nd} International Conference on Engineering of Complex Computer Systems (2017). URL `https://eprints.soton.ac.uk/413132/`

[43] Hoang, T.S., Fürst, A., Abrial, J.R.: Event-B patterns and their tool support. Software & Systems Modeling **12**(2), 229–244 (2013). DOI 10.1007/s10270-010-0183-7

[44] Hoang, T.S., Iliasov, A., Silva, R., Wei, W.: A survey on Event-B decomposition. Electronic Communications of the EASST **46** (2011). URL `http://journal.ub.tu-berlin.de/eceasst/article/view/688`

[45] Hoang, T.S., Snook, C., Ladenberger, L., Butler, M.: 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, M. Biro (eds.) Abstract State Machines, Alloy, B, TLA, VDM, and Z: 5^{th} International Conference, ABZ 2016, Linz, Austria, May 23-27, 2016, Proceedings, pp. 360–375. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-33600-8_31

[46] Iliasov, A., Troubitsyna, E., Laibinis, L., Romanovsky, A., Varpaaniemi, K., Ilić, D., Latvala, T.: Supporting Reuse in Event-B Development: Modularisation Approach. In: M. Frappier, U. Glässer, S. Khurshid, R. Laleau, S. Reeves (eds.) Abstract State Machines, Alloy, B and Z: Second International Conference, ABZ 2010, Orford, QC, Canada, February 22-25, 2010. Proceedings, pp. 174–188. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). DOI 10.1007/978-3-642-11811-1_14

[47] Jackson, M.: Problem Frames: Analyzing and Structuring Software Development Problems. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (2000)

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

[49] Jones, C.: From Problem Frames to HJJ (and its known unknowns). Tech. rep., Newcastle University, Newcastle, United Kingdom (2009)

[50] Kitchenham, B., Charters, S.: Guidelines for performing Systematic Literature Reviews in Software Engineering. Tech. Rep. EBSE 2007-001, Keele University and Durham University Joint Report (2007). URL `http://www.dur.ac.uk/ebse/resources/Systematic-reviews-5-8.pdf`

[51] Kobayashi, T., Ishikawa, F., Honiden, S.: Understanding and planning Event-B refinement through primitive rationales. In: Y. Ait Ameur, K.D. Schewe (eds.) Proceedings of the 4^{th} International Conference: Abstract State Machines, Alloy, B, TLA, VDM, and Z (ABZ 2014), Toulouse, France, pp. 277–283. Springer Berlin Heidelberg, Berlin, Heidelberg (2014). DOI 10.1007/978-3-662-43652-3_24

[52] Kropp, M., Meier, A.: Agile Success Factors - A qualitative study about what makes agile projects successful. Tech. rep., University of Applied Sciences

Northwestern Switzerland (FHNW and ZHAW) (2015). URL http://www.swissagilestudy.ch/

[53] Ladenberger, L., Bendisposto, J., Leuschel, M.: Visualising Event-B Models with B-Motion Studio. In: M. Alpuente, B. Cook, C. Joubert (eds.) Proceedings of the 14th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2009), Eindhoven, The Netherlands, pp. 202–204. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). DOI 10.1007/978-3-642-04570-7_17

[54] Lanoix, A.: Event-B Specification of a Situated Multi-Agent System: Study of a Platoon of Vehicles. In: 2008 2nd IFIP/IEEE International Symposium on Theoretical Aspects of Software Engineering, pp. 297–304 (2008). DOI 10.1109/TASE.2008.39

[55] Lopatkin, I.: Transformation Patterns Plug-in. http://wiki.event-b.org/index.php/Transformation_patterns (2010)

[56] Mashkoor, A., Jacquot, J.P.: Guidelines for formal domain modeling in event-b. In: 2011 IEEE 13th International Symposium on High-Assurance Systems Engineering, pp. 138–145 (2011). DOI 10.1109/HASE.2011.47

[57] Méry, D.: Refinement-Based Guidelines for Algorithmic Systems. International Journal of Software and Informatics (IJSI) **3**(2-3), 197–239 (2009). URL https://hal.inria.fr/inria-00426383

[58] Métayer, C.: AnimB - Open Source B Animator. http://www.animb.org/index.xml (2017)

[59] Métayer, C., Abrial, J.R., Voisin, L.: Event-B Language. Tech. rep., Deliverable 3.2, EU Project IST-511599 - RODIN (2005). URL http://rodin.cs.ncl.ac.uk

[60] Olszewska, M.: On the Impact of Rigorous Approaches on the Quality of Development. Ph.D. thesis, Turku Centre for Computer Science, Turku, Finland (2011)

[61] Olszewska, M., Ostroumov, S., Waldén, M.: Synergising Event-B and Scrum – Experimentation on a Formal Development in an Agile Setting. Tech. Rep. 1152, Åbo Akademi University, Turku, Finland (2016)

[62] Olszewska, M., Ostroumov, S., Waldén, M.: Using Scrum to Develop a Formal Model – An Experience Report. In: P. Abrahamsson, A. Jedlitschka, A. Nguyen Duc, M. Felderer, S. Amasaki, T. Mikkonen (eds.) Proceedings of the 17th International Conference on Product-Focused Software Process Improvement (PROFES 2016), Trondheim, Norway, November 22-24,

2016, pp. 621–626. Springer International Publishing, Cham (2016). DOI 10.1007/978-3-319-49094-6_48

[63] Olszewska, M., Waldén, M.: FormAgi – A Concept for More Flexible Formal Developments. Tech. Rep. 1124, Åbo Akademi University, Turku, Finland (2014)

[64] Olszewska, M., Waldén, M.: DevOps Meets Formal Modelling in High-criticality Complex Systems. In: Proceedings of the 1^{st} International Workshop on Quality-Aware DevOps, QUDOS 2015, pp. 7–12. ACM, New York, NY, USA (2015). DOI 10.1145/2804371.2804373

[65] Ostroumov, S., Waldén, M.: Facilitating Formal Event-B Development by Visual Component-Based Design. Tech. Rep. 1148, Åbo Akademi University, Turku, Finland (2015)

[66] Ostroumov, S., Waldén, M.: Formal library of visual components. Tech. Rep. 1147, Åbo Akademi University, Turku, Finland (2015)

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

[68] Parsa, M., Waldén, M., Snook, C.: An Overview of Formal Specification Languages and Tools Supporting Visualisation of System Development. Tech. Rep. 1127, Åbo Akademi University, Turku, Finland (2014)

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

[70] Ponsard, C., Devroey, X.: Generating high-level Event-B system models from kaos requirements models pp. 317–332 (2011)

[71] Prokhorova, Y.: Rigorous Development of Safety-Critical Systems. Ph.D. thesis, Turku Centre for Computer Science, Turku, Finland (2015)

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

[73] Prokhorova, Y., Troubitsyna, E.: 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), *Lecture*

Notes in Computer Science, vol. 7527, p. 47–62. Springer-Verlag Berlin Heidelberg (2012)

[74] Prokhorova, Y., Troubitsyna, E., Laibinis, L., Kharchenko, V.: Development of Safety-Critical Control Systems in Event-B Using FMEA. In: L. Petre, K. Sere, E. Troubitsyna (eds.) Dependability and Computer Engineering: Concepts for Software-Intensive Systems, p. 75–91. IGI Global (2012)

[75] Rezazadeh, A., Butler, M.: Some Guidelines for Formal Development of Web-Based Applications in B-Method. In: H. Treharne, S. King, M. Henson, 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, pp. 472–492. Springer Berlin Heidelberg, Berlin, Heidelberg (2005). DOI 10.1007/11415787_27

[76] Ricketts, M.: Using stories to teach safety: Practical, research-based tips. Professional safety **60**(5), 51 (2015)

[77] Rodriguez, M.T.: Invariant discovery and refinement plans for formal modelling in Event-B. Ph.D. thesis, Heriot-Watt University, Edinburgh, Scotland (2013)

[78] Romanovsky, A., Thomas, M.: Industrial Deployment of System Engineering Methods. Springer Publishing Company, Incorporated (2013)

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

[80] Sato, N., Ishikawa, F.: Separation of Considerations in Event-B Refinement toward Industrial Use. In: 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 (2015). URL `http://ceur-ws.org/Vol-1385/paper7.pdf`

[81] Savicks, V., Butler, M., Colley, J.: Co-simulation Environment for Rodin: Landing Gear Case Study. In: F. Boniol, V. Wiels, Y. Ait Ameur, 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, pp. 148–153. Springer International Publishing, Cham (2014). DOI 10.1007/978-3-319-07512-9_11

[82] Schwaber, K.: Agile Project Management with Scrum. Microsoft Press, Redmond, WA (2004). URL `http://my.safaribooksonline.com/9780735619937`

[83] Schwaber, K., Sutherland, J.: Scrum. The Official Guide. Scrum Guides (2010)

[84] Silva, R., Butler, M.: Supporting Reuse of Event-B Developments through Generic Instantiation. In: K. Breitman, 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, pp. 466–484. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). DOI 10.1007/978-3-642-10373-5_24

[85] Snook, C., Butler, M.: UML-B and Event-B: an integration of languages and tools (2008). URL `https://eprints.soton.ac.uk/264926/`

[86] Snook, C., Hoang, T.S., Butler, M.: Analysing Security Protocols Using Refinement in iUML-B. In: C. Barrett, M. Davies, T. Kahsai (eds.) Proceedings of the 9^{th} NASA Formal Methods International Symposium (NFM 2017), Moffett Field, CA, USA, May 16-18, 2017,, pp. 84–98. Springer International Publishing, Cham (2017). DOI 10.1007/978-3-319-57288-8_6

[87] Su, W., Abrial, J.R., Zhu, H.: Formalizing Hybrid Systems with Event-B and the Rodin Platform. Science of Computer Programming **94**, 164–202 (2014). DOI 10.1016/j.scico.2014.04.015

[88] University of Southampton: Theory Plug-in. `http://wiki.event-b.org/index.php/Theory_Plug-in` (2010)

[89] University of Southampton: Code generation activity. `http://wiki.event-b.org/index.php/Code_Generation_Activity` (2015)

[90] University of Southampton: UML-B- Statemachine Animation. `http://wiki.event-b.org/index.php/UML-B_Statemachine_Animation` (2017)

[91] Westfall, L.: Safety critical software configuration management practices. In: Proceedings of the International Conference on Software Quality (ICSQ 2011). Westfall Team, Inc. (2011). URL `http://www.westfallteam.com/sites/default/files/papers/Safety%20Critical%20Software%20Configuration%20Management%20Practices.pdf`

[92] Yeganefard, S., Butler, M., Rezazadeh, A.: Evaluation of a guideline by formal modelling of cruise control system in Event-B pp. 182–191 (2010). URL `https://eprints.soton.ac.uk/270899/`

ISBN 978-952-12-3724-9
ISSN 1239-1891