UNIVERSITI PUTRA MALAYSIA

A METHODOLOGY TO SUPPORT UML-B MODEL DEVELOPMENT

MUHAMMED BASHEER JASSER

FSKTM 2018 19

A METHODOLOGY TO SUPPORT UML-B MODEL DEVELOPMENT

By

MUHAMMED BASHEER JASSER

Thesis submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirements for the Degree of Doctor of Philosophy

January 2018

DEDICATIONS

To my mother Sahar and father Mohammad, thank you for your unconditional love, support, advice and encouragement that motivate me to complete my goals and set higher targets.

To my sisters Lima and Lamees, thank you for providing me with hope and love that always surround me with strength and ambition.

To all my relatives and friends, thank you for your understanding and encouragement in many moments. Your friendship makes my life an unforgettable experience. I cannot list all the names, but you are all in my mind and heart.

Finally, to All whom I love.

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirement for the degree of Doctor of Philosophy

A METHODOLOGY TO SUPPORT UML-B MODEL DEVELOPMENT

By

MUHAMMED BASHEER JASSER

January 2018

Chairman: Mar Yah Said, PhD
Faculty: Computer Science and Information Technology

UML-B is a graphical front-end for the formal method Event-B. UML-B models are translated to Event-B for verification purpose. Modelling and proving become difficult for complex models with many state variables and transitions. Reducing modelling and proving effort is potential for UML-B models. Decomposition, composition, model matching, and pattern reuse are attracting methods to support individual team modelling, comparing, integrating, and reusing models helping UML-B modellers and practitioners to avoid remodelling and reduce the proof effort required to verify the correctness of the models.

Decomposition in Event-B divides a complex model into sub-parts reducing certainly its complexity and facilitating independent modelling for the sub-parts. Two decomposition styles exist for Event-B which are shared-event and shared-variable. The shared-event style has been introduced in UML-B and it is considered suitable for synchronous message passing systems. However, the shared-variable style is not supported before in UML-B. In this research, a correct-by-construction method is introduced to support the shared-variable decomposition in UML-B that is considered suitable for asynchronous shared variable models. The proposed method is provided through three sequential phases which are refinement-preparation, actual-decomposition, and refinement-after-decomposition. The provision of new notions, conditions, and rules is a significant step to maintain the graphical semantics of UML-B and the shared variable formalism of Event-B. We introduce refinement-preparation strategies and conditions, actual-decomposition rules, refinement-after-decomposition conditions, new UML-B notions, and a representation mechanism. The method is formalised and verified via a generic proof on a theoretical level to show how the generated semantic implicit invariants and the shared variable formalism are preserved, and to prove that any recomposed machine is indeed correct and a refinement of the original decomposable machine when the method phases conditions and

rules are followed. The method is illustrated by a case study on the updating of master data, in which the notions, conditions, and rules are applied.

Event-B composition is to reuse existing interacting models specifications to construct a larger one fulfilling the complete system behaviour. In this research, a correct-by-construction method is introduced to compose UML-B machines that interact via the shared-variable style. This saves the modelling and proving effort via the correct machines that are being composed. The method is provided through steps and rules, and its correctness is verified by proving that any composed machine is indeed correct when the method rules are followed. The method applicability is also illustrated by the case study on the updating of master data.

Identifying similarities between models has several benefits such as model comparison, integration, and evolution. Several matching and comparison methods have been done in the context of model driven software engineering. However, matching models via a systematic method is not supported yet in UML-B. In this work, we propose a matching method for UML-B elements based on their semantics. This method includes variable-based, event-based, and state-machine matching. The variable-based matching provides rules for matching UML-B classes, attributes, states, and variables. The event-based matching provides rules and cases for matching UML-B transitions and class-events. The state-machine matching provides rules for matching UML-B state-machines based on the state and transition matching rules. The matching rules are formalized by means of the generated corresponding Event-B specifications. The correctness of the rules is justified via preserving the compatibility of the matched state-variables and corresponding modifying events including their matched guards and actions. These rules are illustrated via a communication-based case study to show their applicability.

Pattern reuse allows reusing models in constructing new ones saving the modelling and proving effort. However, there is no systematic method in terms of pattern reuse in UML-B. In this research, a correct-by-construction method is introduced to support the reuse of UML-B models which is based on the Event-B pattern reuse. The method is provided through three sequential phases which are pattern matching, checking, and incorporation. The provision of new guidelines and rules is necessary to preserve both the UML-B graphical semantics and the method correctness. This is proven via a generic proof using the proofs from the pattern that is integrated with the problem model. The proposed method reduces certainly the modelling and proving effort by using existing models in constructing new ones, since it is always that the proof obligations from the pattern model do not need to be proven again during the integration between the pattern and problem models. The method applicability is illustrated by the communication-based case study introduced in the model matching.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Doktor Falsafah

METODOLOGI UNTUK MENYOKONG MODEL PEMBANGUNAN UML-B

Oleh

MUHAMMED BASHEER JASSER

Januari 2018

Pengerusi: Mar Yah Said, PhD
Fakulti: Sains Komputer dan Teknolologi Maklumat

UML-B adalah grafik bahagian hadapan untuk kaedah formal bagi Event-B. Model UML-B diterjemahkan ke Event-B untuk tujuan pengesahan. Pemodelan dan pembuktian menjadi sukar bagi model kompleks yang mempunyai banyak pembolehubah dan peralihan keadaan. Pengurangan model dan usaha pembuktian berpotensi untuk diaplikasikan di dalam Event-B. Penguraian, komposisi, pemadanan model dan penggunaan semula corak adalah kaedah yang menarik untuk menyokong pemodelan pasukan secara individu, perbandingan, integrasi dan penggunaan semula model. Ia membantu UML-B modellers dan pengamal untuk mengelakkan pembentukan semula dan mengurangkan usaha pembuktian yang diperlukan bagi mengesahkan ketepatan model.

Penguraian dalam Event-B akan membahagikan model kompleks kepada beberapa pecahan bahagian bagi mengurangkan kerumitan dan memudahkan pemodelan bebas untuk pecahan bahagian. Dua cara penguraian sedia ada bagi Event-B adalah perkongsian acara dan perkongsian pembolehubah. Gaya perkongsian acara telah diperkenalkan di dalam UML-B dan dianggap sesuai untuk sistem penghantaran pesanan secara serentak. Kaedah Perkongsian pembolehubah pula sebelum ini tidak disokong di dalam UML-B. Di dalam kajian ini, kaedah pembetulan melalui pembinaan diperkenalkan untuk menyokong penguraian perkongsian pembolehubah dalam UML-B yang dianggap sesuai untuk model pembolehubah bersama yang tidak serentak. Kaedah yang dicadangkan dibahagikan kepada tiga fasa berturutan iaitu persediaan penyempurnaan, penguraian sebenar dan penyempurnaan selepas penguraian. Penyediaan idea, syarat dan peraturan baru adalah langkah penting untuk mengekalkan semantik grafik bagi UML-B dan formalisme pembolehubah bersama untuk Event-B. Kami mencadangkan strategi dan syarat persediaan penyempurnaan, peraturan penguraian sebenar, perkongsian pengertian

UML-B dan mekanisme perwakilan. Kaedah ini diformalkan dan disahkan melalui bukti umum pada tahap teoritikal bagi menunjukkan bagaimana penghasilan semantik tersirat tak berubah dan formalisme perkongsian pembolehubah dapat dikekalkan, dan untuk membuktikan bahawa mana-mana mesin yang direkodkan semula adalah tepat dan sempurna dari satu dekomposisi asal apabila syarat dan peraturan bagi fasa di dalam kaedah diikuti. Kaedah ini digambarkan melalui kajian kes dalam pengemaskinian data induk, di mana pengertian, syarat dan peraturan digunakan.

Komposisi Event-B adalah untuk menggunakan semula spesifikasi model interaksi sedia ada untuk membina model yang lebih besar bagi memenuhi tingkah laku sistem yang lengkap. Dalam kajian ini, kaedah pembetulan melalui pembinaan diperkenalkan untuk menyusun model-model UML-B yang akan berinteraksi menggunakan gaya perkongsian pembolehubah. Pemodelan dan pembuktian usaha dapat dijimatkan melalui penghasilan pecahan model yang betul. Kaedah ini disediakan melalui langkah-langkah dan peraturan, dan ketepatannya dapat disahkan dengan membuktikan mesin yang disusun adalah betul apabila peraturan kaedah diikuti. Aplikasi kaedah ini juga digambarkan dengan kajian kes mengenai pengemaskinian data induk.

Mengenal pasti persamaan antara model mempunyai beberapa kebaikan seperti perbandingan model, integrasi dan evolusi. Beberapa kaedah pemadanan dan perbandingan telah dilakukan di dalam konteks kejuruteraan perisisan. Walaubagaimanapun,model yang sepadan melalui kaedah yang sistematik masih belum disokong lagi dalam kaedah UML-B. Dalam karya ini, kami mencadangkan kaedah pemadanan untuk elemen UML-B berdasarkan semantik mereka. Kaedah ini termasuk asas pembolehubah, asas peristiwa, dan keadaan pemadanan mesin. Pemadanan berasaskan pembolehubah menyediakan peraturan untuk pemadanan kelas UML-B, attibut, keadaan dan pembolehubah. Pemadanan berasaskan peristiwa menyediakan peraturan dan kes untuk transisi pemadanan UML-B dan peristiwa kelas. Pemadanan berasaskan keadaan mesin pula menyediakan peraturan untuk keadaan mesin UML-B yang sepadan berdasarkan peraturan pemadanan keadaan dan peralihan. Peraturan yang sesuai diformalkan melalui spesifikasi Event-B yang dihasilkan. Ketepatan peraturan dapat dibuktikan dengan memelihara kesesuaian pembolehubah keadaan dan pengubahsuaian peristiwa yang sepadan termasuklah pengawalan dan tindakan yang dipadankan. Peraturan ini digambarkan melalui kajian kes berdasarkan komunikasi untuk menunjukkan kebolehgunaannya.

Penggunaan semula corak yang membolehkan penggunaan semula model dalam membina model yang baru dapat menjimatkan model dan usaha pembuktian. Walaubagaimanapun, tiada kaedah yang sistematik dari segi penggunaan semula corak dalam UML-B. Dalam kajian ini, kaedah pembetulan melalui pembinaan diperkenalkan untuk menyokong penggunaan semula model UML-B di mana ia berdasarkan penggunaan semula corak Event-B. Kaedah ini disediakan melalui tiga fasa berturutan iaitu pemadanan corak, pemeriksaan dan penggabungan. Penyediaan garis panduan dan peraturan yang baru adalah perlu untuk memelihara kedua-dua semantik grafik UML-B dan ketepatan kaedah. Ini terbukti melalui bukti umum menggunakan bukti dari corak yang telah di integrasikan dengan model masalah. Kaedah yang dicadangkan dapat mengurangkan pemodelan dan usaha pembuktian dengan model sedia ada dalam membina model yang baru. Hal demikian kerana pembuktian dari corak model tidak perlu dibuktikan lagi ketika integrasi antara corak dan masalah model. Aplikasi model ini digambarkan melalui kes kajian berdasarkan komunikasi yang diperkenalkan dalam pemadanan model.

ACKNOWLEDGEMENTS

I would like to sincerely thank my supervisor, Dr. Mar Yah, for her guidance and support throughout this work, and especially for her confidence in me. Her willingness to provide feedback made the completion of this work an enjoyable experience. I would also like to thank Professor Abdul Azim and Dr. Pathiah for the discussion and help.

I would also like to thank Dr. Jamilah. I have been so grateful to her support, encouragement and advice in several occasions.

I cannot forget the help during the discussion with Professor Jean Raymond Abrial. His feedback has clarified several issues in this work.

Finally, I would like to thank all the professors and doctors who helped me during this research work.

This thesis was submitted to the Senate of Universiti Putra Malaysia and has been accepted as fulfilment of the requirement for the degree of Doctor of Philosophy.

The members of the Supervisory Committee were as follows:

Mar Yah Said, PhD
Senior Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Chairman)

Abdul Azim Abdul Ghani, PhD
Professor
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

Pathiah Abdul Samat, PhD
Senior Lecturer
Faculty of Computer Science and Information Technology
Universiti Putra Malaysia
(Member)

ROBIAH BINTI YUNUS, PhD
Professor and Dean
School of Graduate Studies
Universiti Putra Malaysia

Date:

Declaration by graduate student

I hereby confirm that:

- this thesis is my original work;
- quotations, illustrations and citations have been duly referenced;
- this thesis has not been submitted previously or concurrently for any other degree at any other institutions;
- intellectual property from the thesis and copyright of thesis are fully-owned by Universiti Putra Malaysia, as according to the Universiti Putra Malaysia (Research) Rules 2012;
- written permission must be obtained from supervisor and the office of Deputy Vice-Chancellor (Research and Innovation) before thesis is published (in the form of written, printed or in electronic form) including books, journals, modules, proceedings, popular writings, seminar papers, manuscripts, posters, reports, lecture notes, learning modules or any other materials as stated in the Universiti Putra Malaysia (Research) Rules 2012;
- there is no plagiarism or data falsification/fabrication in the thesis, and scholarly integrity is upheld as according to the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) and the Universiti Putra Malaysia (Research) Rules 2012. The thesis has undergone plagiarism detection software.

Signature: ____________________ Date: ____________________

Name and Matric No.: Muhammed Basheer Jasser (GS38917)

Declaration by Members of Supervisory Committee

This is to confirm that:

- the research conducted and the writing of this thesis was under our supervision
- supervision responsibilities as stated in the Universiti Putra Malaysia (Graduate Studies) Rules 2003 (Revision 2012-2013) are adhered to.

Signature: ____________________________
Name of Chairman of Supervisory Committee: ____________________________

Signature: ____________________________
Name of Member of Supervisory Committee: ____________________________

Signature: ____________________________
Name of Member of Supervisory Committee: ____________________________

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

OVERVIEW OF RESEARCH PROJECT

1.1 Introduction

This chapter describes an overview of this thesis. This includes the motivation, the problem statements, the research questions, scope, objectives and summary of contributions of this work.

1.2 Motivation

Formal methods [1, 2] in the software engineering discipline allow the design, modelling, verification, and maintenance of hardware and software systems. Formal methods introduce preciseness, remove ambiguity in specifications, and support the verification of requirements and design properties. The specifications in formal methods could be viewed as mathematical models, which represent the intended behaviour of the systems and they are used to model safety critical systems such as: railway control systems, nuclear power plant control systems, aircraft control systems, intelligent transport systems, and medical systems.

Event-B [1] is a formal method, which is a variant of the B-method and is based on Action Systems. An action system [3] is a collection of actions on a set of state variables. An action system describes the state space (the set of possible values that could be assigned to the state variables) of a system and the possible executable actions that change the values of the system state variables.

UML-B [4, 5, 6, 7] is a graphical front end of Event-B and relies in its infrastructure on Event-B formalism in verifying the system properties. It shares similar properties with object oriented modelling including class instances and attributes. UML-B is similar to UML, but it has a new notation and has its own meta-model. UML-B is supported by a tool which provides the user with the environment for drawing four diagrams. These diagrams are translated to Event-B in order to be verified using Rodin theorem provers.

As the use of formal methods is increasing and the real life scenarios become more complicated, the system specifications are gaining more complexity. This raises the need of methods and approaches to reduce this complexity and manage these formal models. Decomposition, composition, model matching and pattern reuse are proposed in the context of model driven software engineering to facilitate managing the models and dealing with their complexity.

1.3 Problem Statement

Methods and approaches are needed to manage the formal models and handle their complexity. Decomposition, composition, model matching and pattern reuse are proposed in the context of model driven software engineering to facilitate managing the models and dealing with their complexity.

Decomposition splits a model specification into subparts in order to prove and model each subpart individually. Two decomposition styles exist in Event-B which are the shared-event [8] and the shared-variable [9] styles. The shared-event decomposition has been introduced in UML-B [10] and it is considered suitable for synchronous message passing distributed systems. However, the shared-variable decomposition style, which is considered suitable for asynchronous shared-memory models, is not introduced yet in UML-B.

Also, composition reuses existing correct models constructing a larger one by means of integration. Shared-event composition has been introduced in UML-B [10], however shared-variable composition is not yet introduced in UML-B.

In addition, model matching is for identifying similarities between models. This has several benefits such as model comparison, integration, and evolution. Several matching and comparison methods have been done in the context of model driven software engineering [11], however, there is no existing research that concerns the matching and comparison for UML-B models.

Further more, pattern reuse is important in software engineering discipline to save effort and avoid remodelling. Reuse is seen in the use of design patterns which are general solutions to common problems. A pattern reuse method has been introduced in Event-B to avoid remodelling and reproving [12] of Event-B models. It is possible to construct complex UML-B models that are correct-by-construction. However, there no systematic method in terms of pattern reuse to support the construction of these models.

The purpose of this research is to provide a methodology to support managing and modelling UML-B specifications. This methodology supports the development in UML-B in which modelling and proving efforts are handled.

1.4 Research Questions

1. What are the extensions needed for UML-B language to support shared-variable decomposition?
2. What are the necessary rules to prepare and decompose UML-B models by the shared-variable style?.
3. What are the necessary rules to prepare and compose UML-B models by the shared-variable style?.
4. How models in UML-B could be matched compatibly and consistently?
5. How pattern models in UML-B could be reused to construct larger models?

1.5 Research Objectives

The main objective of this research is:

- To design a methodology including a set of methods to facilitate the management of UML-B models and handling the models complexity.

The sub-objectives of this research are:

1. To extend UML-B language to support the shared-variable decomposition.
2. To propose a method of shared-variable decomposition for UML-B models.
3. To propose a method of shared-variable composition for UML-B models.
4. To propose a method of model matching for UML-B models.
5. To propose a method of pattern reuse for UML-B models.

1.6 Summary of Contributions

In this research, multiple methods to support and save the modelling and proving effort in UML-B are provided.

In Chapter 4, a method for shared-variable decomposition is proposed. The sub-contributions in this method are as follows:

- Refinement-preparation invariants and strategies.
- Actual-decomposition rules.
- Refinement-after-decomposition conditions and a representation mechanism.
- New UML-B notions to realize the method.
- A formalisation and a generic formal proof for the method.
- A case study to illustrate the method applicability.

In Chapter 5, a composition method is proposed based on the shared-variable style. The method rules are proposed, formalized and evaluated by a case study.

In Chapter 6, a method for matching models in UML-B is proposed based on the set-theory, first-order-logic and Event-B that are the basis semantics for UML-B. The sub-contributions in this method are as follows:

- Variable-based matching rules.
- Event-based matching rules.
- State-machine matching rules.
- A formalization of the rules and a compatibility proof.
- An illustrating case study.

In Chapter 7, a method for pattern reuse is proposed based on the Event-B pattern reuse and the matching method in Chapter 6. The sub-contributions in this method are as follows:

- Specialization of the model matching rules that are introduced in Chapter 6 to correspond to the pattern reuse concept.
- Checking rules for UML-B state-machine transition and class event that use the UML-B class, attribute, state and variable.
- Incorporation rules for UML-B class, attribute, state, variable state-machine, transition and class event.
- A formalization of the pattern reuse method.
- A formal proof for the method on a theoretical level.
- A case study to illustrate the method applicability.

1.7 Research Scope

This research focuses on supporting the modeling of UML-B machines. Shared-variable decomposition and composition are to decompose and compose UML-B machines in the context of shared-memory models. Model matching is to match UML-B machines including their contained classes, attributes, states, variables, transitions and class-events. Pattern reuse is to use any existing UML-B machine model in constructing a larger problem UML-B refinement machine model.

1.8 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2 describes a background study related to this research. A background is presented about the formal methods (B-Method, Event-B, Z, VDM), some of the well-known object-oriented formal languages (VDM++, Object-Z), UML-B, some parallel computing models, refinement and decomposition in (Event-B, UML-B), pattern reuse in Event-B, and related work.

Chapter 3 presents the research operational framework, research methodology needed to meet the objectives, and tools which are used.

Chapter 4 introduces the shared-variable decomposition method. The method is provided in this chapter through three sequential phases: refinement-preparation, actual-decomposition, and refinement-after-decomposition.

Chapter 5 introduces the shared-variable composition method including the required conditions and the necessary rules to compose machines.

Chapter 6 introduces the method for model matching including the variable-based, event-based, and state-machine matching rules.

Chapter 7 introduces the method for patten reuse. The method is provided in this chapter through three phases: matching, checking, and incorporation.

Chapter 8 concludes the thesis. A summary of contributions, limitations, and future works of the thesis are presented. Comparison to other work on decomposition, composition, model matching, and pattern reuse are also discussed.

REFERENCES

[1] Jean-Raymond Abrial. *Modeling in Event-B: System and software engineering*. Cambridge University Press, 2010.

[2] Jean-Raymond Abrial and Jean-Raymond Abrial. *The B-Book: Assigning programs to meanings*. Cambridge University Press, 2005.

[3] R-JR Back and Reino Kurki-Suonio. Decentralization of process nets with centralized control. *Distributed Computing*, 3(2):73–87, 1989.

[4] Colin Snook and Michael Butler. UML-B: Formal modeling and design aided by UML. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 15(1):92–122, 2006.

[5] Colin Snook and Michael Butler. UML-B and Event-B: an integration of languages and tools. 2008.

[6] Colin Snook and Michael Butler. UML-B: A plug-in for the Event-B tool set. 2008.

[7] Colin Snook, Ian Oliver, and Michael Butler. *The UML-B profile for formal systems modelling in UML*, pages 69–84. Springer, 2004.

[8] Renato Silva and Michael Butler. Shared event composition/decomposition in Event-B. In *Formal Methods for Components and Objects*, pages 122–141. Springer, 2012.

[9] Jean-Raymond Abrial. Event model decomposition, ETH Zurich, Technical Report 626. 2009.

[10] Mar Yah Said. *Methodology of refinement and decomposition in UML-B*. Thesis, 2010.

[11] Dimitrios S Kolovos, Davide Di Ruscio, Alfonso Pierantonio, and Richard F Paige. Different models for model matching: An analysis of approaches to support model differencing. In *Proceedings of the 2009 ICSE Workshop on Comparison and Versioning of Software Models*, pages 1–6. IEEE Computer Society, 2009.

[12] Thai Son Hoang, Andreas Fürst, and Jean-Raymond Abrial. Event-B patterns and their tool support. *Software & Systems Modeling*, 12(2):229–244, 2013.

[13] Stephen J Garland, John V Guttag, and James J Horning. An overview of larch. In *Functional programming, concurrency, simulation and automated reasoning*, pages 329–348. Springer, 1993.

[14] Gary T Leavens, Erik Poll, Curtis Clifton, Yoonsik Cheon, Clyde Ruby, David Cok, Peter Müller, Joseph Kiniry, Patrice Chalin, Daniel M Zimmerman, et al. JML reference manual, 2008.

[15] Kevin Lano. *The B language and method: a guide to practical formal development*. Springer Science & Business Media, 2012.

[16] Jean-Raymond Abrial. From Z to B and then Event-B: Assigning proofs to meaningful programs. In *Integrated Formal Methods*, pages 1–15. Springer.

[17] Jonathan Peter Bowen. *Formal specification and documentation using Z: A case study approach*, volume 66. International Thomson Computer Press London, 1996.

[18] Jonathan Peter Bowen. Comp. specification. z and Z forum frequently asked questions. In *International Conference of Z Users*, pages 407–416. Springer, 1998.

[19] Jonathan P Bowen. Z: A formal specification notation. In *Software specification methods*, pages 3–19. Springer, 2001.

[20] Petra Malik and Mark Utting. CZT: A framework for Z tools. In *International Conference of B and Z Users*, pages 65–84. Springer, 2005.

[21] Rob Arthan. Proofpower. *on-line documentation. URL: http://www. lemma-one. com/ProofPower*, 2007.

[22] Daniel Plagge and Michael Leuschel. Validating Z specifications using the ProB animator and model checker. In *International Conference on Integrated Formal Methods*, pages 480–500. Springer, 2007.

[23] Michael Leuschel and Michael Butler. ProB: A model checker for B. In *International Symposium of Formal Methods Europe*, pages 855–874. Springer, 2003.

[24] Dines Bjørner. The Vienna Development Method (VDM). In *Mathematical Studies of Information Processing*, pages 326–359. Springer, 1979.

[25] Cliff B Jones. Scientific decisions which characterize VDM. In *International Symposium on Formal Methods*, pages 28–47. Springer, 1999.

[26] Cliff B Jones. *Systematic software development using VDM*, volume 2. Citeseer, 1986.

[27] Vangalur S Alagar and Kasilingam Periyasamy. *Specification of software systems*. Springer Science & Business Media, 2011.

[28] Peter Gorm Larsen. Ten years of historical development bootstrapping PDM tools VX. *Journal of Universal Computer Science*, 7(8):692–709, 2001.

[29] Peter Gorm Larsen, Kenneth Lausdahl, Augusto Ribeiro, Sune Wolff, Nick Battle, and B RG12. Overture VDM-10 tool support: User guide. Technical report, Technical Report TR-2010-02, The Overture Initiative, www. overturetool. org, 2010.

[30] Robin Bloomfield, Peter Froome, and Brian Monahan. Specbox: a toolkit for BSI-VDM. *SafetyNet*, 5:4–7, 1989.

[31] B Atelier. The industrial tool to efficiently deploy the B-Method. *URL: http://www.atelierb.eu. (access date 04/09/2017)*, 2017.

[32] Juan C Bicarregui and Brian Ritchie. Reasoning about VDM developments using the VDM support tool in mural. In *International Symposium of VDM Europe*, pages 371–388. Springer, 1991.

[33] Yves Ledru. Proof-based development of specifications with KIDS/VDM. In *International Symposium of Formal Methods Europe*, pages 214–232. Springer, 1994.

[34] Kevin Lano and Howard Haughton. *Specification in B: An introduction using the B toolkit*, volume 71. World Scientific, 1996.

[35] Jean-Raymond Abrial, Michael Butler, Stefan Hallerstede, Thai Son Hoang, Farhad Mehta, and Laurent Voisin. Rodin: an open toolset for modelling and reasoning in Event-B. *International journal on software tools for technology transfer*, 12(6):447–466, 2010.

[36] Object Management Group. Unified modeling language UML, 2017.

[37] James Rumbaugh, Ivar Jacobson, and Grady Booch. *The Unified Modeling Language Reference Manual*. Pearson Higher Education, 2004.

[38] Susan Stepney, Rosalind Barden, and David Cooper. A survey of object orientation in Z. *Software Engineering Journal*, 7(2):150–160, 1992.

[39] Graeme Smith. *An object-oriented approach to formal specification*. PhD thesis, Citeseer, 1992.

[40] Roger Duke, Gordon Rose, and Graeme Smith. Object-Z: A specification language advocated for the description of standards. *Computer Standards & Interfaces*, 17(5):511–533, 1995.

[41] Graeme Smith. *The Object-Z specification language*, volume 1. Springer Science & Business Media, 2012.

[42] Kevin Lano. Z++, an object-orientated extension to Z. In *Z User Workshop, Oxford 1990*, pages 151–172. Springer, 1991.

[43] Antonio J Alencar and Joseph A Goguen. OOZE: An object oriented Z environment. In *European Conference on Object-Oriented Programming*, pages 180–199. Springer, 1991.

[44] Anthony Hall. Using Z as a specification calculus for object-oriented systems. In *International Symposium of VDM Europe*, pages 290–318. Springer, 1990.

[45] Stephen A Schuman, David H Pitt, and PJ Byers. Object-oriented process specification. In *Specification and Verification of Concurrent Systems*, pages 21–70. Springer, 1990.

[46] Eugene Diirr and J van Katwijk. VDM++, a formal specification language for object-oriented designs. In *Proceedings 6th Annual European Computer Conference, Compeuro*, pages 214–219, 1992.

[47] Hung Ledang and Jeanine Souquieres. Formalizing uml behavioral diagrams with B. In *Tenth OOPSLA Workshop on Behavioral Semantics: Back to Basics*, pages 12–p, 2001.

[48] Hung Ledang and Jeanine Souquières. Modeling class operations in B: application to UML behavioral diagrams. In *Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on*, pages 289–296. IEEE, 2001.

[49] Hung Ledang and Jeanine Souquières. Integration of UML and B specification techniques: Systematic transformation from OCL expressions into B. In *Ninth Asia Pacific Software Engineering Conference-APSEC'2002*, pages 10–p, 2002.

[50] Ninh-Thuan Truong and Jeanine Souquieres. Verification of behavioural elements of UML models using B. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 1546–1552. ACM, 2005.

[51] Colin Snook. Combining UML and B. 2002.

[52] Colin Snook and Michael Butler. U2b-a tool for translating UML-B models into B. 2004.

[53] The Eclipse Foundation. Eclipse Modelling Framework. `http://www.eclipse.org/modeling/emf/`, 2017. Accessed: 04/09/2017.

[54] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: eclipse modeling framework*. Pearson Education, 2008.

[55] The Eclipse Foundation. Graphical Modelling Framework. `http://projects.eclipse.org/projects/modeling.gmp`, 2017. Accessed: 04/09/2017.

[56] Hesham El-Rewini and Mostafa Abd-El-Barr. *Advanced computer architecture and parallel processing*, volume 42. John Wiley & Sons, 2005.

[57] Blaise Barney et al. Introduction to parallel computing. *Lawrence Livermore National Laboratory*, 6(13):10, 2010.

[58] Guy E Blelloch and Bruce M Maggs. Parallel algorithms. In *Algorithms and theory of computation handbook*, pages 25–25. Chapman & Hall/CRC, 2010.

[59] Ana Cavalcanti, Augusto Sampaio, and Jim Woodcock. *Refinement Techniques in Software Engineering: First Pernambuco Summer School on Software Engineering, PSSE 2004, Recife, Brazil, November 23-December 5, 2004, Revised Lectures*, volume 3167. Springer, 2006.

[60] Edsger Wybe Dijkstra, Edsger Wybe Dijkstra, Edsger Wybe Dijkstra, Etats-Unis Informaticien, and Edsger Wybe Dijkstra. *A discipline of programming*, volume 1. prentice-hall Englewood Cliffs, 1976.

[61] Jean-Raymond Abrial and Stefan Hallerstede. Refinement, decomposition, and instantiation of discrete models: Application to Event-B. *Fundamenta Informaticae*, 77(1-2):1–28, 2007.

[62] John Derrick and Eerke Boiten. *Refinement in Z and Object-Z*, volume 30. Springer, 2001.

[63] John Derrick and Eerke Boiten. Refinement in Object-Z. In *Refinement in Z and Object-Z*, pages 385–402. Springer, 2014.

[64] Graeme Smith and John Derrick. Refinement and verification of concurrent systems specified in Object-Z and CSP. In *Formal Engineering Methods., 1997. Proceedings., First IEEE International Conference on*, pages 293–302. IEEE, 1997.

[65] John Derrick and Eerke Boiten. Refinement of objects and operations in Object-Z. In *Formal Methods for Open Object-based Distributed Systems IV*, pages 257–277. Springer, 2000.

[66] Tim McComb and Graeme Smith. Compositional class refinement in Object-Z. In *International Symposium on Formal Methods*, pages 205–220. Springer, 2006.

[67] Kevin Lano and SJ Goldsack. Refinement, subtyping and subclassing in VDM++. *Theory and Formal Methods*, 95, 1994.

[68] SJ Goldsack and Kevin Lano. Annealing and data decomposition in VDM. *ACM Sigplan Notices*, 31(4):32–38, 1996.

[69] Yojiro Kawamata, Christian Sommer, Fuyuki Ishikawa, and Shinichi Honiden. Specifying and checking refinement relationships in VDM++. In *2009 Seventh IEEE International Conference on Software Engineering and Formal Methods*, pages 220–227. IEEE, 2009.

[70] Mar Yah Said, Michael Butler, and Colin Snook. Class and state machine refinement in UML-B. In *Proceedings of Workshop on Integration of Model-based Formal Methods and Tools (associated with IFM 2009)*.

[71] Mar Yah Said, Michael Butler, and Colin Snook. A method of refinement in UML-B. *Software & Systems Modeling*, pages 1–24, 2015.

[72] Daniel Jackson and Michael Jackson. Problem decomposition for reuse. *Software Engineering Journal*, 11(1):19–30, 1996.

[73] Grady Booch. *Object oriented analysis & design with application*. Pearson Education India, 2006.

[74] Michael Butler. Decomposition structures for Event-B. In *Integrated Formal Methods*, pages 20–38. Springer, 2009.

[75] Ali Gondal, Michael Poppleton, and Michael Butler. Composing Event-B specifications-case-study experience. In *Software Composition*, pages 100–115. Springer, 2011.

[76] Thai Son Hoang, Alexei Iliasov, Renato A Silva, and Wei Wei. A survey on Event-B decomposition. *Electronic Communications of the EASST*, 46, 2011.

[77] Ingo Brückner. Slicing CSP-OZ specifications. In *Nordic Workshop on Programming Theory*, page 71, 2004.

[78] Ingo Brückner and Heike Wehrheim. Slicing an integrated formal method for verification. In *International Conference on Formal Engineering Methods*, pages 360–374. Springer, 2005.

[79] Ingo Brückner and Heike Wehrheim. Slicing Object-Z specifications for verification. In *International Conference of B and Z Users*, pages 414–433. Springer, 2005.

[80] Björn Metzler, Heike Wehrheim, and Daniel Wonisch. Decomposition for compositional verification. In *International Conference on Formal Engineering Methods*, pages 105–125. Springer, 2008.

[81] Kirsten Winter and Graeme Smith. Compositional verification for Object-Z. In *International Conference of B and Z Users*, pages 280–299. Springer, 2003.

[82] Renato Silva and Michael Butler. Supporting reuse of event-b developments through generic instantiation. In *International Conference on Formal Engineering Methods*, pages 466–484. Springer, 2009.

[83] Jun-Jang Jeng and Betty HC Cheng. Specification matching for software reuse: A foundation. In *ACM SIGSOFT Software Engineering Notes*, volume 20, pages 97–105. ACM, 1995.

[84] Amy Moormann Zaremski and Jeannette M Wing. Specification matching of software components. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 6(4):333–369, 1997.

[85] Frank Feiks and David Hemer. Specification matching of object-oriented components. In *Software Engineering and Formal Methods, 2003. Proceedings. First International Conference on*, pages 182–190. IEEE, 2003.

[86] David Hemer. Specification matching of state-based modular components. In *Software Engineering Conference, 2003. Tenth Asia-Pacific*, pages 446–455. IEEE, 2003.

[87] Jörg Becker, Patrick Delfmann, Sebastian Herwig, and Łukasz Lis. A generic set theory-based pattern matching approach for the analysis of conceptual models. In *International Conference on Conceptual Modeling*, pages 41–54. Springer, 2009.

[88] Fathi Taibi, Fouad Mohammed Abbou, and Md Jahangir Alam. A matching approach for object-oriented formal specifications. *Journal of Object Technology*, 7(8):139–153, 2008.

[89] Wei-Jin Park and Doo-Hwan Bae. A two-stage framework for uml specification matching. *Information and Software Technology*, 53(3):230–244, 2011.

[90] Richard C Gronback. *Eclipse modeling project: a domain-specific language (DSL) toolkit*. Pearson Education, 2009.

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

[92] Dave Steinberg, Frank Budinsky, Ed Merks, and Marcelo Paternostro. *EMF: Eclipse Modeling Framework*. Pearson Education, 2008.

[93] Clark Barrett and Cesare Tinelli. Cvc3. In *Computer Aided Verification*, pages 298–302. Springer, 2007.

[94] Thomas Bouton, Diego Caminha B De Oliveira, David DÃl'harbe, and Pascal Fontaine. *veriT: an open, trustable and efficient SMT-solver*, pages 151–156. Springer, 2009.

[95] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[96] Thai Son Hoang, Andreas Fürst, and Jean-Raymond Abrial. Event-B patterns and their tool support. *Software & Systems Modeling*, 12(2):229–244, 2013.