
Contents

INTRODUCTION	xv
Jean-Louis BOULANGER	
CHAPTER 1. PRESENTATION OF THE B METHOD	1
Jean-Louis BOULANGER	
1.1. Introduction.	1
1.2. The B method	3
1.2.1. Presentation	3
1.2.2. The concept of an abstract machine	4
1.2.3. From machines to implementations	11
1.3. Verification and validation (V&V)	15
1.3.1. Internal verification.	15
1.3.2. Validation or external verification	20
1.4. Methodology	20
1.4.1. Development by layer	21
1.4.2. Role of the breakdown in the makeup of the POs	23
1.4.3. Development cycle of a B project	23
1.5. Feedback based on experience	26
1.5.1. A few figures	26
1.5.2. Some uses	26
1.6. Conclusion	30
1.7. Glossary	31
1.8. Bibliography	32
CHAPTER 2. ATELIER B	35
Thierry LECOMTE	
2.1. Introduction.	35
2.2. Automatic refinement.	37
2.3. Code generation	39

2.4. Proof and model animation	43
2.5. The move toward open source.	44
2.6. Glossary.	45
2.7. Bibliography	45
CHAPTER 3. B TOOLS	47
Jean-Louis BOULANGER	
3.1. Introduction.	47
3.2. General principles	47
3.3. Atelier B	48
3.3.1. Project management	48
3.3.2. Typechecking and PO generation	50
3.3.3. Code generation.	52
3.3.4. Prover.	53
3.3.5. Tool qualification.	56
3.4. Open source tools	57
3.4.1. Presentation	57
3.4.2. ABTools	58
3.5. Conclusion	78
3.6. Glossary.	79
3.7. Bibliography	79
CHAPTER 4. THE B METHOD AT SIEMENS	83
Daniel DOLLE	
4.1. Introduction.	83
4.1.1. Siemens Industry Mobility	83
4.1.2. The CBTC system	85
4.1.3. Characteristics of B programs	87
4.1.4. The target calculator	88
4.2. The development process using B	89
4.2.1. Development	89
4.2.2. Informal specification	90
4.2.3. Formalization of the specification	92
4.2.4. Refinement and coding	95
4.2.5. Proof	101
4.3. Monitoring	103
4.3.1. Development review	103
4.3.2. Testing	105
4.3.3. Safety validation	106
4.4. Digging deeper.	109
4.4.1. Translation from B to Ada	109
4.4.2. Abstract models and concrete models	110

4.4.3. Functional calculation with safety monitoring	112
4.4.4. Configuration	115
4.4.5. Limitations.	117
4.5. Automatic refinement.	119
4.5.1. History	119
4.5.2. Operational principles	120
4.5.3. Interactive refinement	123
4.6. Conclusion	123
4.7. Glossary.	125
4.8. Bibliography	126
CHAPTER 5. INDUSTRIAL APPLICATIONS FOR MODELING WITH THE B METHOD	129
Thierry LECOMTE	
5.1. Introduction.	129
5.2. Control-command systems for controlling platform doors	131
5.3. Safety of microelectronic components	142
5.4. Conclusion	147
5.5. Glossary.	148
5.6. Bibliography	149
CHAPTER 6. FORMALIZATION OF DIGITAL CIRCUITS USING THE B METHOD	151
Jean-Louis BOULANGER	
6.1. Introduction.	151
6.2. B method and VHDL	152
6.3. Modeling digital circuits	153
6.3.1. Modeling methodology	154
6.3.2. Modeling a basic logic gate, NOT	155
6.3.3. Modeling an additioner	157
6.3.4. Modeling of complex circuit: a multiplexer	166
6.4. VHDL libraries.	170
6.4.1. The STD_LOGIC_1164 library	171
6.4.2. The B components for STD_LOGIC_1164	172
6.4.3. The multiplexer	178
6.5. VHDL to B	180
6.6. Conclusions.	181
6.6.1. Some limitations	181
6.6.2. Advantages	181
6.6.3. Future work	182
6.6.4. To finish	183
6.7. Bibliography	184

CHAPTER 7. PRAGMATIC USE OF B: THE POWER OF FORMAL METHODS WITHOUT THE BULK	187
Christophe METAYER, François BUSTANY and Mathieu CLABAUT	
7.1. Introduction	187
7.2. Prototyping for formal models	187
7.3. Inspiration from agile methods	189
7.4. Simultaneous development and validation	189
7.5. Performances of software developed in B	190
7.6. Use of infinity: separating algorithmic thinking and programming issues	193
7.7. Industrial implementation of event-B	196
7.8. B method for software and event-B	198
7.9. Conclusion	199
7.10. Glossary	199
7.11. Bibliography	200
CHAPTER 8. BRILLANT/BCAML — A FREE TOOLS PLATFORM FOR THE B METHOD	201
Samuel COLIN and Dorian PETIT	
8.1. What is BRILLANT/BCaml?	201
8.2. Organization	202
8.3. Functions	204
8.3.1. The historic kernel	204
8.3.2. Code manipulation	206
8.3.3. Proving B specifications	207
8.4. Perspectives	207
8.5. Bibliography	209
CHAPTER 9. TRANSLATING B AND EVENT-B MACHINES TO JAVA AND JML	211
Néstor CATAÑO, Víctor RIVERA, Camilo RUEDA and Tim WAHLS	
9.1. Introduction	211
9.2. Background	214
9.2.1. The B method	215
9.2.2. The Event-B method	217
9.2.3. JML	219
9.3. Translating B to JML	220
9.3.1. The translation	220
9.3.2. The B2Jml tool	228
9.3.3. Case study: translating the B social networking model to JML	228

9.4. Translating Event-B to JML and Java	232
9.4.1. The translation	233
9.4.2. The EventB2Java tool	238
9.4.3. Case Study: translating the Event-B social networking model to Java and JML	242
9.5. Future work and conclusion	247
9.6. Bibliography	249
CHAPTER 10. EVENT B.	253
Dominique MÉRY and Neeraj Kumar SINGH	
10.1. Introduction	254
10.2. Modeling and verification of a system	254
10.2.1. Modeling	254
10.2.2. Safety properties.	257
10.3. Event B: a modeling language	260
10.3.1. Basic elements of an Event B model.	262
10.3.2. Invariance properties in Event B	263
10.3.3. Refinement of events	265
10.3.4. Structures for Event B models	266
10.4. Formal development of a sequential algorithm	269
10.4.1. Derivation of an algorithm for computing the sum of a sequence of values by refinement and transformation of the model into an algorithm	270
10.4.2. Development of a sequential algorithm using the proof-based pattern call-as-event	278
10.5. Development of a distributed algorithm	284
10.5.1. Modeling distributed algorithms	284
10.5.2. Elements of a proof-based pattern	287
10.6. Tools	291
10.6.1. Atelier B	291
10.6.2. The Rodin platform	291
10.7. Conclusion and perspectives	292
10.7.1. Applications in case studies	292
10.7.2. Conclusion and perspectives	293
10.8. Bibliography	294
CHAPTER 11. B-RAIL: UML TO B TRANSFORMATION IN MODELING A LEVEL CROSSING	299
Jean-Louis BOULANGER	
11.1. Introduction	299
11.2. Level crossings: general overview	300

11.3. Managing requirements	301
11.3.1. Requirements	301
11.3.2. Recommendations, requirements and properties	303
11.3.3. Requirements engineering	306
11.4. UML notation and the B method	314
11.4.1. UML notation	314
11.4.2. The B method	316
11.4.3. Overview	317
11.5. Step 1: requirement acquisition	318
11.5.1. Requirement extraction.	318
11.5.2. Risk identification.	320
11.5.3. Identification of services.	321
11.6. Step 2: environment and risk analysis	323
11.6.1. Identification of the environment.	323
11.6.2. Description of the environment	326
11.6.3. Environmental faults	328
11.6.4. Maintenance	332
11.6.5. Impact of the environment on the system	333
11.6.6. Results	334
11.7. Step 3: component breakdown.	336
11.7.1. Requirement selection	336
11.7.2. Architecture	337
11.7.3. Behavior	338
11.8. Step 4: verification.	340
11.8.1. Introduction	340
11.8.2. Description of formal models	341
11.9. UML2B	342
11.10. Conclusions	343
11.11. Glossary	344
11.12. Bibliography.	345
CHAPTER 12. FEASIBILITY OF THE USE OF FORMAL METHODS FOR MANUFACTURING SYSTEMS	349
Pascal LAMY, Philippe CHARPENTIER, Jean-François PETIN and Dominique EVROT	
12.1. Introduction	349
12.2. Presentation of the requirement	350
12.3. The methods chosen and a brief description of them.	352
12.3.1. The B method	352
12.3.2. Specification with SysML and formal verification by model checking	354

12.4. Description of the machine: mechanical press with clutch-brake . . .	356
12.4.1. Description of the press	356
12.4.2. Brief description of the operating modes	358
12.4.3. Brief description of the means of protection	359
12.4.4. Characteristics of the programmable logic controller	359
12.5. Process followed for the design, validation and generation of the software using the B method	359
12.5.1. Creation of a B compatible specification	360
12.5.2. B Model: specification and design	362
12.5.3. Generation of a C code and simulation	366
12.5.4. Generation of the code for the PLC and validation	367
12.5.5. Conclusion on the use of the B method for the creation of application software in an industrial and manufacturing context.	370
12.6. Formalization of the requirements and properties helping SysML and verification of the unitary modules by model checker	371
12.6.1. Overall view of the design process for manufacturing systems . .	371
12.6.2. Modeling the requirements	373
12.6.3. Modeling functional and organic architectures.	378
12.6.4. Traceability of the requirements	380
12.6.5. Development and verification of the software command components	382
12.6.6. Discussion	385
12.7. Conclusion on the use of formal techniques in the field of manufacturing	387
12.8. Glossary	388
12.9. Bibliography	388
 CHAPTER 13. B EXTENDED TO FLOATING-POINT NUMBERS: IS IT SUFFICIENT FOR PROVING AVIONICS SOFTWARE?	 391
Jean-Louis DUFOUR	
13.1. Introduction	391
13.2. Motivation.	392
13.3. Integers and the railway origins of the B method.	393
13.3.1. The SACEM project	393
13.3.2. The need for an innovative software method	394
13.3.3. The coded processor and integers	395
13.3.4. The limitations of Hoare logic and the beginnings of B	396
13.3.5. Successes of B, and integers once more!	397
13.3.6. The positive influence of “fail-safe” on complexity.	397
13.4. The avionics context: floating-point numbers and complexity	398
13.5. Barking up the wrong tree: separation between integer and floating-point calculations.	401

13.6. IEEE 754 Floating-point numbers	403
13.6.1. Scope of the standard	403
13.6.2. The behavior of floating-point numbers is complex	405
13.6.3. Infinities and NaNs	407
13.7. Reasons underlying extension to floating-point numbers	408
13.7.1. Overview	408
13.7.2. Real numbers	409
13.7.3. Concrete floating-point numbers	410
13.7.4. Abstract floating-point numbers	411
13.8. Returning to the useful properties that need to be proved	413
13.8.1. In avionics, specifications are complex	413
13.8.2. Can vector data be abstracted?	414
13.8.3. The gap between algorithmic specifications and pre-conditions of leaf procedures	415
13.8.4. Integrators and the formalization of the system boundaries	416
13.9. Conclusion	417
13.10. Appendix: the confusion between overflow, infinity and illegal parameters	418
13.10.1. Presentation of the issue	418
13.10.2. Confusion between overflow and infinity	419
13.10.3. Confusion between infinity and illegal parameters.	421
13.11. Glossary	422
13.12. Bibliography.	423
CHAPTER 14. FROM ANIMATION TO DATA VALIDATION: THE PROB CONSTRAINT SOLVER 10 YEARS ON	427
Michael LEUSCHEL, Jens BENDISPOSTO, Ivo DOBRIKOV, Sebastian KRINGS and Daniel PLAGGE	
14.1. The problem.	427
14.1.1. Animation for B	428
14.1.2. Model checking B.	430
14.1.3. Data validation	431
14.1.4. Constraint-based checking and disproving for B.	432
14.1.5. Summary	433
14.2. Choice of implementation technology	433
14.2.1. What was used before?	433
14.2.2. Why was constraint logic programming used?	434
14.3. Implementation of the PROB constraint solver	435
14.3.1. Architecture	435
14.3.2. Validation	438
14.4. Added value of constraint programming	440
14.4.1. Cost of development	440
14.4.2. User feedback	440

14.4.3. Was it difficult/necessary for the end user to understand constraint technology?	441
14.4.4. Comparison with non-constraint solving tools	441
14.4.5. Comparison with other technologies	442
14.4.6. Future plans	443
14.4.7. Lessons	443
14.5. Acknowledgments	444
14.6. Bibliography	444
CHAPTER 15. UNIFIED TRAIN DRIVING POLICY	447
Alexei ILIASOV, Ilya LOPATKIN and Alexander ROMANOVSKY	
15.1. Introduction	447
15.2. Overview	449
15.3. Semantics	452
15.4. Modeling notation	454
15.5. Verification	461
15.5.1. Constraint satisfiability.	462
15.5.2. Hazard avoidance	466
15.5.3. Example.	467
15.6. Discussion	469
15.7. Conclusions	471
15.8. Bibliography	472
CONCLUSION	475
GLOSSARY	481
LIST OF AUTHORS	487
INDEX	489