





# The First Twenty-Five Years of Industrial Use of the B Method

Michael Butler<sup>2</sup> , Philipp Körner<sup>1</sup>  , Sebastian Krings<sup>1</sup> , Thierry Lecomte<sup>3</sup>, Michael Leuschel<sup>1</sup>  , Luis-Fernando Mejia<sup>3</sup>, and Laurent Voisin<sup>4</sup> 

<sup>1</sup> Institut für Informatik, Universität Düsseldorf  
Universitätsstr. 1, D-40225 Düsseldorf, Germany  
{p.koerner,sebastian.krings,leuschel}@hhu.de

<sup>2</sup> University of Southampton  
University Road, Southampton, SO17 1BJ, UK  
mjb@ecs.soton.ac.uk

<sup>3</sup> CLEARSY  
320 avenue Archimède, 13100 Aix en Provence, France  
{thierry.lecomte,fernando.mejia}@clearsy.com

<sup>4</sup> Systereel  
1090 rue René Descartes, 13100 Aix-en-Provence, France  
laurent.voisin@systereel.fr

**Abstract.** The B method has an interesting history, where language and tools have evolved over the years. This not only led to considerable research and progress in the area of formal methods, but also to numerous industrial applications, in particular in the railway domain. We present a survey of the industrial usage of the B-Method since the first toolset in 1993 and the inauguration of the driverless metro line 14 in Paris in 1999. We discuss the various areas of applications, from software development to data validation and on to systems modelling. The evolution of the tooling landscape is also analysed and we present an assessment of the current situation, lessons learned and possible new directions.

## 1 Introduction

The B method [4] for software and systems development and its successor Event-B [6] has a rich history. B has originally been developed as a successor to Z [10] by Jean-Raymond Abrial in the 1990s, focusing on two key concepts: refinement to gradually develop models and tool support, in particular proof and code generation. As of today, three classes of industrial applications of B have established themselves:

- the original B for software development (classical B) [4]: refine specifications until a low-level subset of B is reached where code generation is applied.
- B for system modelling (Event-B) [6]: model an entire system, not just a particular software component and then verify critical properties and understand why a system is correct.
- B for data validation: express properties in B and check data and configuration parameters in a certified manner.

42 In this article, we discuss these three classes of applications in turn (Sec-  
43 tions 2, 3 and 4), focussing on railway applications. We then discuss applications  
44 in other areas in Sect. 5, the tools behind the industrial applications in Sect. 6  
45 and finally conclude with lessons learnt over the years in the final Sect. 7.

## 46 2 B for Software: The Early Days and Industrial Uptake

47 In the 80s, RATP (Régie Autonome des Transports Parisiens, Paris railway  
48 transport operator and infrastructure manager) and the consortium<sup>5</sup> in charge  
49 of the development of SACEM, the train protection system deployed on the  
50 Parisian RER Line A, faced the validation of the first control/command software  
51 of a safety critical railway signalling system ever operated in France. Without a  
52 real background in that domain they started using a tool based on Hoare Logic  
53 to verify assertions included in the code. Then they consulted Jean-Raymond  
54 Abrial who proposed the formalisation and verification of a formal specification  
55 of the software with what can be considered as a sketch of the B-Method. His  
56 proposal was accepted, validation engineers were trained, a formal specification  
57 was written and verified. Eventually, in 1988, SACEM was put into operation to  
58 the satisfaction of all.

59 In the same year, Abrial presented “The B Tool” [2] with an unnamed syntax.  
60 Most of the further developments, both on the language and on advanced tooling,  
61 were initiated during industrial projects in Paris.

62 After the SACEM experience, RATP, guided by Claude Hennebert, requested  
63 the use of the B-Method for the development of the safety critical software of  
64 the train protection system of the driverless Paris Métro Line 14. Alstom, with  
65 this project in mind, but also for their own developments, decided to use the  
66 B-Method. These are the origins of the development and use of the B-Method,  
67 and of formal methods in general, in the French railway industry.

68 In 1989, Alstom, RATP and SNCF (Société Nationale des Chemins de fer  
69 Français), willing to industrialise the B-Method, launched a project whose pur-  
70 pose was threefold: firstly, to train engineers in the principles of the method,  
71 secondly, to develop tools to support it and, thirdly, to create methodological  
72 guides to use it. This project, funded partially by the French government and  
73 driven by Alstom, established a close collaboration with J.-R. Abrial and the  
74 team of Ib Sørensen at British Petroleum then at B-Core. Additionally, Abrial  
75 was still in contact with a research group in Oxford and certainly was influenced  
76 in technical details.

77 After some training sessions given by J.-R. Abrial, an Alstom team started  
78 the development of railway applications with the first version of the B language  
79 and tools provided by J.-R. Abrial and his colleagues. The fundamental syntactic  
80 and semantic concepts of the language were already present in this version but,  
81 in 1991 it turned out that evolutions of the language and tools were necessary  
82 to structure, analyse and prove software of industrial size and complexity. This

---

<sup>5</sup> Consisting of Alstom, Compagnie des Signaux (today Hitachi), and Matra Transport (today Siemens Transportation, France).

is the reason why structuring sharing and configuration clauses were introduced in the language. With the support of J.-R. Abrial, Alstom decided to develop its own set of tools for the new language. After two years of development, in 1993, the first version of what was called the B-Toolset was delivered internally, including a type checker, a proof obligation generator and a theorem prover able to manage software of industrial size and complexity. By that time, J.-R. Abrial proposed to Alstom that the Digilog company (then Steria, today CLEARSY) industrialises these tools and make them available to RATP and to Matra Transport, the supplier that won the Paris Métro Line 14 contract (cf. section 2.2). Alstom accepted and Digilog developed ATELIER B based on Alstom's B-Toolset.

The development of the language and of supporting tools was a very important aspect of the industrialisation of the B-Method. A no less important aspect was creating the conditions of its industrial application. Indeed, it was crucial to define effective and efficient use process and methodology for a technology being defined and to train engineers to it simultaneously.

## 2.1 Early Adoption

The introduction of the B-Method in an existing conventional software development environment necessarily induced the modification of the development process known and accepted by the development, verification and validation teams, the clients and the safety assessors. The questions to answer were numerous considering that the new process should comply with applicable railway standards, that it should be close to the existing process in order to reuse its infrastructure as much as possible, and consequently, that the activities related with the B-Method must be included within the phases of this process. Some questions were:

- Where should the definition of B abstract machines be included? In the software specification phase or in the software design phase?
- How should B components and formal proofs be documented?
- How should B components and formal proofs be verified, when and by whom?
- How should verification efforts be documented?
- How should module testing, integration and validation testing phases be modified in order to take advantage of the formal proof of B components?
- How should the development of the part of the software that does *not* need to be formally developed interact with the development of the part of the software that needs to be formally developed?

The companies, that introduced the B-Method in their software development process, answered these questions according to their own practices and experience. When it was decided to introduce it for the development of safety critical software of railway systems, the B-Method was neither taught nor used anywhere. Its first users were trained by J.-R. Abrial himself, who followed also the first developments with it. The methodology for using the B-Method and good practices were defined during these first developments. It addresses the following questions:

- 127 – How to create the architecture of a large B model?
- 128 – How to write the operations of abstract machines?
- 129 – How to refine abstract data with concrete data?
- 130 – How to refine operations?
- 131 – How to write loops?

132 User guides were written and some rules were automatised with tools. Once  
 133 the tools and the good practices were developed and defined, training courses  
 134 in the B-Method were given to all the persons in the organisation dealing with  
 135 software engineering: software development engineers, verifiers, validators, safety  
 136 assurance engineers and all of their managers. The sustainability of the B-  
 137 Method in industry has been made possible by the creation of an eco-system  
 138 including RATP, the operator that requested the application of the method,  
 139 CLEARSY, the company that industrialised and that maintains the tools sup-  
 140 porting B in the long term, the engineering schools and universities in France and  
 141 abroad, that train engineers, conduct research and provide tools, and finally, the  
 142 companies that provide B expertise and technical assistance. The existence of  
 143 international conferences on formal methods and, particularly, on the B-Method  
 144 and the participation of the industrial companies to these events contribute to  
 145 the dissemination and sustainability of the method.

## 146 2.2 Driverless Metro Software: Météor and its successors

147 *Paris Line 14* The most well-known success story of B is the Parisian Métro Line  
 148 14. The main goal was to reduce the time interval between trains, yet ensuring  
 149 the correctness of the system. For this, the train control was automated and the  
 150 trains are able to travel without a human driver. All safety critical components,  
 151 i.e. the train control and the controllers for the automatic doors dividing the  
 152 passengers from the tracks, have been formally developed using B. Since October  
 153 1998, the metro works flawlessly and not a single issue was caused by software.  
 154 The same holds for the shuttle train at the Roissy Airport that drives since 2007.

155 The B models for the Line 14 and the Roissy Shuttle have 115 000 and 183 000  
 156 lines of code, 27 800 and 43 610 proofs and a manual proof percentage of 8.1 %  
 157 and 3.3 %, respectively. More information concerning both metros and their full  
 158 development statistics can be found in [5] and Sect. 4.2 of [62].

159 In neither case, unit testing was performed. Instead, formal development and  
 160 proof gave enough confidence in the correctness of the generated Ada code which  
 161 was used without change. The only tests performed were tests concerning the  
 162 integration with non-critical software parts and global validation tests.

163 The early projects including Line 14 pushed tool development. One example  
 164 is semi-automatic refinement [21], which by now is included as BART in ATE-  
 165 LIER B. Going through (data) refinement steps manually is tedious. Often, this  
 166 work can be automated though: such a tool can drastically improve development  
 167 speed, in particular if code generation from B0 is required.

168 *Canarsie CBTC* Siemens have evolved the product for Line 14 and installed  
 169 it on many metro lines world-wide, notably on the Canarsie Line [30] in New

170 York. 53 trains operate without interruption on 17 km of track consisting of 24  
 171 stations. In contrast to the Parisian Line 14, two different types of trains are  
 172 mixed on the track: more modern trains are equipped with CBTC (computer-  
 173 or communications-based train control) systems, whereas older trains are not.  
 174 This results in a system that is far more complex than its counterpart in Paris.

175 Again, the software components of the system are split into parts that are  
 176 safety-critical, and those that are non-critical for safety. All safety-critical parts  
 177 have been developed in B; the only exceptions are components that cannot be  
 178 expressed in a B model, including low-level communications, sensor, motors,  
 179 breaks and file input/output. By now, the latter is possible in some tools.

180 The Canarsie Line was one of the first industrial applications that included  
 181 the use of automatic refinement tools. Even though way more proof obligations  
 182 had to be discharged, these tools proved to speed up the process considerably.  
 183 One of the key sentences concerning the usage of formal methods can be found  
 184 in the reflection of the project:

185 *“Beyond the technological challenge of using such a complex formal method in*  
 186 *an industrial context, it is now clear for us that building software using B is not*  
 187 *more expensive than using conventional methods. Better, due to our experience*  
 188 *in using this method, we can assert that using B is cheaper when considering*  
 189 *the whole development process (from specification to validation and sometimes*  
 190 *certification).” [30]*

191 *Safety-Critical Train Software at Alstom* Alstom has been over the years one  
 192 of the proponents of using the B-method to produce safety critical software.  
 193 Most Alstom trains now include some software which was produced from a B  
 194 specification. Table 1 gives an overview of the important railway products where  
 195 software was developed with the B-Method. The URBALIS 400 product, with  
 196 its over 100 installations worldwide, represents currently the most widespread  
 197 use of the B method for software in the world.

## 198 2.3 Code Generation for Hardware

199 There were a few research projects on using B for hardware, e.g., at AWE [31]  
 200 or within the PUSSEE research project [60], with applications for SmartCards  
 201 (see Sect. 5 below). Only recently has B for hardware led to publicly industrial  
 202 railway applications.

203 *Platform Screen Doors Controllers* CLEARSY has developed several safety sys-  
 204 tems controlling the opening and closing of the Platform Screen Doors (PSD)  
 205 installed in Metro stations in order to insure passengers protection. These sys-  
 206 tems are independent of the train signalling and automatic operating systems;  
 207 they can be installed in a Metro which is already in service. Due to the expanse  
 208 of the urban population in most big cities in the world, PSD are a first step  
 209 towards full automation of a non-automatic existing metro line. The PSD con-  
 210 trollers developed by CLEARSY are specified and programmed with B. First  
 211 controllers used a dedicated translator from B to Ladder Logic [44], more recent

**Table 1.** Overview of Alstom Projects

Product	Size (kloc)	First commissioning
Train speed controller for Calcutta Metro	Small (< 10)	1992
KVB, train protection system for French mainlines (no high-speed trains) trains.	Medium (10..50)	1995
SACEM extension for Paris RER Line A.	Small (< 10)	1996
Train speed controller for Cairo Metro.	Small (< 10)	1997
Speed controller for Lyon Metro.	Small (< 10)	1998
Lineside Electronic Unit (LEU) for mainlines in Australia, China, France, Greece, Italy, Spain, The Netherlands.	Small (< 10)	2000
URBALIS 200, train protection system for metro lines in Chile, China, Egypt, India, South Korea, Spain.	Medium (10..50)	2003
URBALIS 400, CBTC system for metro lines in Australia, Brazil, Chile, China, Dubai, France, Italy, Mexico, Panama, Qatar, Saudi Arabia, Singapore, Spain, The Netherlands, Vietnam.	Large (50..250)	2008

ones use the CLEARSY Safety Platform. Paris lines 1 and 13, São Paulo lines 2 and 3, and Stockholm Citybanan metros have been equipped with such PSD controllers, certified SIL3 or SIL4.

*The CLEARSY Safety Platform* The CLEARSY Safety Platform [39] is both a hardware and software platform, aimed at easing the development and the deployment of safety critical applications, up to SIL4. It relies on the integration of the B-Method for programming (including mathematical proof), redundant code generation and compilation, and a hardware platform that ensures a safe execution of the software. Safety principles are built-in in the hardware and the safety library. The associated IDE is based on ATELIER B and the B language supported [42] has been specialised to address the specific hardware, to better ensure safety, and to minimise the proof effort. As of today, the CLEARSY Safety Platform has been certified 3 times with different certification bodies, for international railways applications.

### 3 B for System Modelling

*From Software to Systems: Event-B for System Modelling* The success of the Parisian Métro Line 14 showed that, given a set of software requirements, one could develop formally a program that fulfils it and prove it correct. But the software requirements used as input make some assumption on the environment in which the software is to be operated: logical interfaces to other pieces of software as well as electronic and physical devices such as motors.

233 Therefore, if the software requirements are wrong or do not fit the operational  
234 environment, the resulting system as a whole would malfunction. It was thus felt  
235 necessary to move the application of formal methods to an earlier phase in the  
236 system development process, namely in the system design phase. System design  
237 is performed by very capable engineers, but addresses very complex systems with  
238 a lot of moving parts and is difficult to reason about informally.

239 It was thus felt necessary to extend the B method to system design activi-  
240 ties [9,3] and another notation was gradually derived from the B language, finally  
241 crystallising into Event-B, aiming for proven system studies where computation  
242 is distributed. In four EU projects concerning the development and industrialisa-  
243 tion of Event-B, numerous industrial partners were involved, including Siemens,  
244 Bosch, SAP, Space Systems Finland, Alstom, CLEARSY, Gemplus, Leonardo  
245 and Critical Software Technologies.

246 *New York Flushing Line* CLEARSY used Event-B, supported by the ATELIER B  
247 tool, on two major industrial rail projects for New York City Transit (NYCT) to  
248 support safety assurance [57]. In the first project for the New York Flushing line,  
249 formal models of a CBTC were developed and key safety properties were specified  
250 and proved at the system level. The main safety properties addressed were avoid-  
251 ance of train collisions, avoidance of trains traversing unlocked switches (causing  
252 derailment) and avoidance of over-speeding. The second project for NYCT in-  
253 volved a different implementation of interlocking from the first. Because the  
254 system level models were abstracted from details of the implementation, it was  
255 possible to reuse models from the first project in the second project, considerably  
256 reducing safety analysis effort in the second project. A key benefit of the system  
257 level formal analysis identified in [57] was the way it identified precise properties  
258 required of the various sub-systems in the design and the assumptions made in  
259 one sub-system about other sub-systems. Since different sub-systems were pro-  
260 vided by different companies, this ensured that these assumptions were clearly  
261 communicated to relevant stakeholders at early stages of the development, avoid-  
262 ing problems later during the systems' integration phase.

263 *Octys* In [27], CLEARSY outline how they used Event-B supported by ATE-  
264 LIER B to perform safety analysis of an existing CBTC system called Octys  
265 for RATP. Some key insights into the benefits of formalisation are described.  
266 For each safety property to be verified, the taken approach was to describe the  
267 property informally and an informal argument was developed to explain why  
268 the property held. This helped to frame the subsequent formal modelling and  
269 reasoning. It was found that it was very difficult to achieve a high level of rigour  
270 through the informal reasoning and that the formal reasoning filled in gaps in the  
271 reasoning, providing more complete arguments for safety. The formal reasoning  
272 also allowed the isolation of the minimal set of assumptions required to prove  
273 the desired property. This allowed for identification of gaps in assumptions and  
274 also allowed the informal safety requirements to be improved.

275 *URBALIS 400 Zone Controller* In 2018 [28] the software for the Zone Controller  
 276 of the Alstom URBALIS 400 CBTC developed using classical B (see Sect. 2.2  
 277 above) underwent a rigorous systems analysis. While the classical B method  
 278 ensured that the implementation is correct wrt the software specification, it  
 279 does not guarantee that the algorithms themselves are correct wrt system level  
 280 requirements. The analysis was formalised with an Event-B model which links  
 281 environment variables (the real position of the trains) with software variables  
 282 (protection envelopes). ATELIER B and PROB were used to analyse the system  
 283 and extract key properties that ensure the correct and safe functioning. These  
 284 properties are of crucial importance when tuning or extending the algorithms of  
 285 the zone controller.

286 *RailGround* As part of the EU H2020 Enable-S3 project, Thales Austria GmbH  
 287 and the University of Southampton applied Event-B and UML-B to the *Rail-*  
 288 *Ground* interlocking system [24]. The project used UML-B, which allows editing  
 289 Event-B models using a UML-like graphical representation.

290 As well as demonstrating the feasibility of modelling a complex interlocking  
 291 system in UML-B, the project also demonstrated benefits of using the diagram-  
 292 matic notation. The diagrammatic models were found to be easier to communi-  
 293 cate to domain experts than the textual models.

294 *ETCS Hybrid Level 3* HL3 is a novel train control concept, that aims at increas-  
 295 ing the throughput of trains without additional rails. Thales Deutschland GmbH  
 296 and Universität Düsseldorf used the B Method to develop a reference model for  
 297 a new approach to railway interlocking, Hybrid ERTMS/ETCS Level 3 (HL3),  
 298 as part of a field demonstration of the feasibility of the HL3 principles [34]. The  
 299 focus of the project was on the use of the model-checking and execution capa-  
 300 bilities of PROB both for validation of the model *and* for use of the model as a  
 301 reference implementation of the HL3 principles during the field demonstrator. A  
 302 graphical visualisation of the railway environment made it easy for the domain  
 303 experts to provide feedback on the formal model, leading to improvements of  
 304 the specification. A lot of the complexity of HL3 concerns degraded modes and  
 305 corner cases, and the formalisation and validation approach allowed these to be  
 306 addressed comprehensively. Execution of the B model on PROB was used to  
 307 conduct field tests with real trains in realtime.

308 *EULYNX* Founded in 2014, EULYNX is a joined European project by several  
 309 railway infrastructure providers aiming at a standardisation of interfaces and  
 310 signaling systems. One of EULYNX members, the infrastructure division of the  
 311 German railway company Deutsche Bahn, uses model-based systems engineering  
 312 for their interlocking systems. Using SysML has lead to improved specifications  
 313 and thus increased the quality of the interlocking system. However, SysML is  
 314 merely a semi-formal language. In consequence, within the European Shift2Rail  
 315 2 project, an approach based on UML-B has been used to introduce an Event-  
 316 B representation into the development process [54,55]. This effectively enables  
 317 formal verification of interlocking systems specified in SysML.



## 318 4 B for Data Validation

319 In the last decade, the B language gained a new application area: aside from  
320 proving software, it can be used to ensure that *assumptions* about configura-  
321 tion data hold (often dubbed data validation). Indeed, a safety critical system  
322 often contains many data parameters which are instantiated differently for each  
323 particular deployment of the system. These parameters underlie restrictions to  
324 ensure the proper functioning of the system. When a system is incorrectly con-  
325 figured, this can lead to disaster. It turned out that the B language was very  
326 convenient to express properties for correct configuration.

327 This intuition gave rise to the development of the OVADO [1] tool for RATP,  
328 which took place in parallel of the early development of the Rodin platform.  
329 Before that tool, RATP used to have dedicated tools developed in order to check  
330 the correctness of configuration data for systems received from its suppliers.  
331 But these dedicated tools were quite expensive to develop and inflexible. Any  
332 change in the requirements made it necessary to change the software of the tool.  
333 In contrast, with a generic tool like OVADO, one just needs to modify the B  
334 expression of the property to reflect the change and run the OVADO tool again.

335 In some cases, when the software was developed from a B specification, these  
336 properties were already expressed in B and used during the formal safety proof.  
337 This was, e.g., the case for the Paris Line 1 and 14 metro systems and other  
338 installations of the same system in Barcelona or São Paulo. This was one of the  
339 first industrial uses of data validation using PROB by Siemens [48,49,32], inde-  
340 pendently<sup>6</sup> conducted in 2008-2009 within the EU Deploy project. Before 2009,  
341 Siemens was using ATELIER B with custom proof rules and tactics, dedicated  
342 to dealing with larger data values [18,19]. This, however, did not scale to many  
343 larger properties or data values, meaning that manual validation was required  
344 and thus cost intensive and error prone. Indeed, the use of PROB did uncover  
345 at least one issue that was missed by the manual validation.

346 In order to better address industrial needs, tools developed dialects of the  
347 B language and domain specific data validation languages on top of B [43]. In  
348 the context of data validation string manipulations are important; hence PROB  
349 now allows usage of B's sequence operators on strings (e.g., for concatenation).  
350 Additionally, support for reading and writing XML was added to PROB during  
351 a case study in cooperation with Thales [35].

352 Data validation with B has now been applied to many railway systems world-  
353 wide, some of which are:

- 354 – Line 1 Paris, CDGVAL LISA, São Paulo line 4, ALGER line 1, Barcelona  
355 line 9 by Siemens using a tool called RDV built-on top of PROB [48,49,32],
- 356 – by RATP for its lines using Ovado using a tool developed by CLEARSY  
357 called predicateB as first chain, and PROB as secondary tool chain [1,12]
- 358 – by Alstom for their URBALIS 400 CBTC system in 2014 using a tool based  
359 on PROB called DTVT developed by CLEARSY for various lines, e.g., in  
360 Mexico, Toronto, São Paulo and Panama [43].

---

<sup>6</sup> Initially the PROB team was unaware of the development of Ovado.

361 – Alstom and SNCF also applied data validation for ETCS-Level 1 software  
 362 in 2018 using another tool developed by CLEARSY using PROB.  
 363 – Together with Systere, Alstom conducted data validation of the Octys CBTC  
 364 for RATP in 2017 using the Ovado tool.  
 365 – by Thales using a tool based on PROB called Rubin for checking engineering  
 366 rules of their ETCS Radio Block Centre (some aspects of Rubin are discussed  
 367 in [35]).  
 368 – Other tools based on PROB were developed by CLEARSY such as Dave for  
 369 General Electric or the latest generation tool called Caval.  
 370 An important aspect of these applications is the certification of the tools  
 371 according to EN50128. Indeed, this norm stipulates that a data validation tool  
 372 is of class T2, namely a tool that “*supports the test or verification of the design*  
 373 *or executable code, where errors in the tool can fail to reveal but cannot directly*  
 374 *create errors in the executable software*” [25]. The tools mentioned above satisfy  
 375 the T2 requirements, e.g., by using a rigorous specification of the tool’s purpose  
 376 and a rigorous testing process (see, e.g. [14]). The Caval tool obtained a T2  
 377 certificate in November 2019. The tools DTVT and Ovado even use a double  
 378 chain: a primary tool that conducts the verification and a secondary tool that  
 379 re-checks the result of the first tool.

## 380 5 Projects Outside the Railway Domain

381 Only few projects outside the railway industry are known to use B. Below, we  
 382 present two additional areas of application.

383 *Modelling Vehicles* In the early 2000s, several projects were initiated to model  
 384 vehicles, e.g., to improve the failure diagnostic of the first full-electronic multi-  
 385 plexed Peugeots as well as to ease the integration of the sub-systems of a one-time  
 386 built military vehicle.

387 Due to the existence of the vehicles and the complexity of the design, the  
 388 modelling adopted was a flat (no refinement) Event-B specification of the func-  
 389 tional specification sided with a dictionary model providing additional seman-  
 390 tics and natural language translation elements. A tool, Composys, was developed  
 391 to automate validation and testing of functional architectures. It contained a B  
 392 model checking tool, a component-based consistency checking tool, and a natural  
 393 language technical documentation generator.

394 *Smart Cards* When it comes to smart cards, the use of formal methods is manda-  
 395 tory for certification, if a high EAL security level is required. In this case, the  
 396 functional specification of the software library is proved to comply with the se-  
 397 curity policy, both formalised with Event-B. Hence, application developers are  
 398 assured that whatever the API calls, the smart card security is enforced i.e. no  
 399 secret is disclosed. Several certifications have been obtained at the highest levels,  
 400 in France and in Germany.

401 B was also used for embedded software development [38] while Event-B was  
 402 used for hardware development [16]. The former used the default ATELIER B C

code generator while the latter was based on a dedicated translator from Event-B to synthesisable VHDL.

## 6 B-method Tools Throughout the Years

In this section, we will discuss tools for the B-method that were developed throughout the years. As expected, not all of them survived. While some have been replaced by successors, others were only of academic interest. Given that most of the tools, including their features and peculiarities, are documented by various research papers and journal articles we keep things brief and reference the publications below.

*B-Toolkit* One of the first tools for use with the B-method is the B-Toolkit [45,56] by B-Core. The B-Toolkit already was reasonably complete, offering editing, type checking, animation, PO generation and discharge, documentation generation, and a first code generator targeting C. B-Toolkit is no longer supported, however, the source code has been released under a BSD license at <https://github.com/edwardcrichton/BToolkit>.

*Click'n Prove* Click'n Prove [8] was an experimental user interface meant to explore new ways to interact with a prover (by clicking rather than command-line) and served as a basis for the Rodin interactive prover interface. Click'n Prove was built on top of XEmacs. Internally, it was using the ATELIER B tools for proving.

*ATELIER B* As mentioned above, the success of B in the railway domain drove the implementation and improvement of B-method tools, such as ATELIER B, initially to be used for software validation [26,2].

In order to be useful for the SIL applications mentioned, ATELIER B needed to be verified and validated itself. To do so, several tasks were performed under the overall regiment of RATP [40]:

- the theorem prover was subject to external expertise,
- a dedicated tableau-based prover was built to validate most of the theorem prover's mathematical rules,
- a committee was set up to demonstrate unprocessed rules by hand,
- a small automated prover was developed to verify the correctness of the dedicated tableau-based prover.

When it was created in 2001, CLEARSY gathered ATELIER B property rights from Alstom and RATP. ATELIER B is currently used by more than 30% CBTC-based automatic metros worldwide, for embedded and track-side safety software. This IDE is continuously developed with new peripheral functions, e.g.:

- an automatic refiner tool, BART [41], similar to the one used by Siemens for the Canarsie line,
- a framework to automatically prove and review added mathematical proof rules, that generates a report for the safety case,

- 443 – a generic new proof obligation generator,
- 444 – integration of the PROB model checker, SMT solvers and the Why3 platform
- 445 in the interactive prover,
- 446 – an improved C code generator targeting PIC32 microcontrollers,
- 447 – a compiler from B0 models to binary files for the CLEARSY Safety Platform.

448 While initially only supporting classical B, current versions of ATELIER B  
449 can handle Event-B machines as well. This renders ATELIER B one of the two  
450 major IDEs for Event-B, the other one being Rodin.

451 *Rodin* Rodin [7] has been developed during the Rodin, Deploy and Advance  
452 projects mentioned above. As a complete IDE, Rodin features the Event-B mod-  
453 elling database / storage, a type checker for Event-B and a proof engine [59] as  
454 well as different editors. Central parts of Rodin have been formally specified and  
455 proved using Click’n Prove [7].

456 As the team developing Rodin knew how much implementation effort went  
457 into building ATELIER B from the ground up, they were looking into ways to  
458 build Rodin on top of an existing framework and finally settled for Eclipse, from  
459 which Rodin inherits its main UI, the handling of workspaces and many internals.  
460 Just like Eclipse, Rodin is based on plugins and could (at least in theory) be  
461 extended to formalisms other than Event-B.

462 Rodin itself does not include a prover for Event-B. Instead, its just maintains  
463 proof trees in sequent calculus and allows reasoners and tactics to be added by  
464 plugins. In particular, ATELIER B’s provers can be added to Rodin alongside  
465 others, e.g., SMT solvers. Influenced by the interactive control of provers imple-  
466 mented in Click’n Prove, Rodin provides a user interface for interactive proof,  
467 in which the different reasoners can be applied, proof tactics can be performed  
468 and the proof tree can be explored graphically. By doing so, provers can be used  
469 collaboratively inside a single proof.

470 Several code generators are available for Rodin, as are other extensions via  
471 the plugin mechanism. Among the most prominent ones are extensions for the  
472 composition and decomposition of models [22] and the theory plugin, which  
473 permits extending the mathematical core of Event-B by custom theories [51,50].

474 *PROB* PROB [47,46] is an animator, constraint solver and model checker for the  
475 B-Method. Its development started in 2001 with a first alpha release made in Oc-  
476 tober 2003. It filled a gap in the B tooling landscape at the time, supporting the  
477 interactive and automatic validation of high-level specifications. Indeed, follow-  
478 ing classical B’s correct-by-construction approach it is vital that the high-level  
479 specifications correctly capture the high-level requirements and functionality.

480 Only the B-Toolkit animator provided some very limited form of validation,  
481 and required the user to provide values for parameters and existentially quan-  
482 tified variables, the validity of which was checked by the BToolkit prover. This  
483 approach was justified by the undecidability of the B language, but was tedious  
484 for the user and prevented automated validation. In contrast, PROB allows fully  
485 automatic animation of specifications, i.e., values for constants, parameters are  
486 computed by PROB’s constraint solver rather than explicitly given by users.

Unknown to the PROB team at the time (around 2000), another team pursued similar ideas leading to the CLP-S solver [20] and the BZTT tool [11] based on it. This work also gave rise to a company (Lerios), which concentrated on model-based test-case generation and later ported the technology to an imperative programming language. Unfortunately, the development of BZTT and CLP-S has been halted; the tool is no longer available.

Using a variety of explicit-state and symbolic model checking approaches, PROB can be used to systematically check a specification for invariant violations [47]. Furthermore, PROB supports LTL model checking, distributed explicit state model checking. Model checking aside, the constraint-solving capabilities of PROB can also be used for model finding, symbolic model checking and deadlock checking as well as test-case generation and drive many of the animation, visualisation and data validation tools that will be discussed below.

*Animation & Visualisation Tools* For the industrial applicability of formal methods, visualisation and graphical model animation allow formal method experts to communicate with domain experts and enable them to identify errors. This may go as far as having a “management view”, that is easy to understand and hides all technical details [40]. Many visualisation and animation tools have been developed for B and Event-B. Among the first ones are BRAMA [58], which uses Flash to graphically visualise models. AnimB [53], a plugin for Rodin, also provides graphical visualisations based on Flash. Several tools were developed building on top of PROB: starting from an early prototype using Flash [15], BMotionStudio has been developed for editing and displaying visualisations [36]. Later on, BMotionStudio was superseded by BMotionWeb [37], an animation engine based on common web technologies, and the simpler VisB [61] based on SVG graphics. Another web-based animator was JEB [63], which was independently developed in JavaScript.

## 7 Discussion and Conclusions

Development of the B language and tools has been driven by industrial needs, which probably explains part of its success. A recent survey in the railway domain [13] cites mature tooling as the most important reasons to use a specific formal method. Mainly ATELIER B and PROB have been developed for a long time and have proven themselves in industry projects. They both are mature tools that also are actively maintained and further features are developed. The reader may also wish to consult older surveys on industrial use of formal methods in general such as [17,62,33,52]. Here is our assessment of the current situation concerning the use of B in industry:

- B is arguably one of the formal methods with the most industrial impact, albeit mainly in the railway sector.
- There is still little industrial use of B in production outside of railways. B seems like a DSL for the railways: topologies can be well expressed using B relations, integer arithmetic is sufficient in many applications. In railways we

- 529 have clearly defined operating environments which enable exhaustive formal  
 530 modelling and inductive reasoning.
- 531 – The flagship products of Alstom’s U400 and the successors of Météor are  
 532 still operating and being installed on new lines. URBALIS 400 is running on  
 533 over 100 lines and has 25% of the worldwide market in CBTC systems.<sup>7</sup>
  - 534 – Code generation for B has now moved to hardware level but the use of  
 535 classical B for software (outside of hardware) is not increasing. It has not  
 536 caught on in Siemens to other products and is not being applied to new  
 537 products at Alstom anymore. One reason may be the need for experienced  
 538 people. Moving from formal modelling to code generation requires a lot of  
 539 extra resources. New tools like automatic refinement (BART) help to some  
 540 extent, but one still spends a lot of time discharging proof obligations of  
 541 little practical value (and it takes time to identify the really crucial proof  
 542 obligations that pose essential problems).
  - 543 – Rodin has had a lot of academic impact, but real industrial use for pro-  
 544 duction systems is still somewhat disappointing. Several aspects of Rodin  
 545 were stimulating academic research and experimentation, but were possibly  
 546 detrimental for industrial use, e.g.: the use of Eclipse with its extension mech-  
 547 anism, the core language without sequences and machine inclusions, models  
 548 being stored in an extensible database rather than a textual format. An ex-  
 549 ception here is UML-B [29] and Coda [23], where the tight integration with  
 550 Eclipse enabled graphical modelling and industrial applications (cf. Sect. 3).  
 551 Also, machine inclusion and textual format are now supported by the new  
 552 CamilleX plug-in.<sup>8</sup>
  - 553 – B for data validation has caught on and is being used for a wide range of  
 554 railway products.
  - 555 – There is an increased interest and activity by a wider range of industrial  
 556 players for systems modelling with B. This is one area where we foresee con-  
 557 siderable growth in the coming years, and where B could maybe move to  
 558 more widespread use outside of the railway domain.

559 **Acknowledgements** We would like to show our gratitude to Jean-Raymond  
 560 Abrial, who provided us with sources, discussions, insider information and knowl-  
 561 edge from his personal experiences developing B and Event-B.

## 562 References

- 563 1. R. Abo and L. Voisin. Formal implementation of data validation for railway safety-  
 564 related systems with OVADO. In *Proceedings SEFM 2013*, volume 8368 of *LNCS*,  
 565 pages 221–236. Springer, 2014.
- 566 2. J.-R. Abrial. The B Tool (Abstract). In *Proceedings VDM*, volume 328 of *LNCS*,  
 567 pages 86–87. Springer, 1988.

<sup>7</sup> See the site (accessed 25/5/2020): <https://www.alstom.com/our-solutions/signalling/urbalis-cbtc-range-future-signalling-systems>

<sup>8</sup> See <https://wiki.event-b.org/index.php/CamilleX>

- 568 3. J.-R. Abrial. Extending B without changing it. In *Proceedings B*, 1996.
- 569 4. J.-R. Abrial. *The B-Book*. Cambridge University Press, 1996.
- 570 5. J.-R. Abrial. Formal Methods: Theory Becoming Practice. *Journal of Universal*  
571 *Computer Science*, 13(5):619–628, 2007.
- 572 6. J.-R. Abrial. *Modeling in Event-B: System and Software Engineering*. Cambridge  
573 University Press, 2010.
- 574 7. J.-R. Abrial, M. Butler, S. Hallerstede, and L. Voisin. An Open Extensible Tool  
575 Environment for Event-B. In *Proceedings ICFEM*, volume 4260 of *LNCS*, pages  
576 588–605. Springer, 2006.
- 577 8. J.-R. Abrial and D. Cansell. Click’n Prove: Interactive Proofs within Set Theory.  
578 In *Proceedings TPHOL*, volume 2758 of *LNCS*, pages 1–24. Springer, 2003.
- 579 9. J.-R. Abrial and L. Mussat. Introducing dynamic constraints in B. In *Proceedings*  
580 *B*, volume 1393 of *LNCS*, pages 83–128. Springer, 1998.
- 581 10. J.-R. Abrial, S. Schuman, and B. Meyer. Specification language. *On the Construc-*  
582 *tion of Programs: An Advanced Course*, page 343, 1980.
- 583 11. F. Ambert, F. Bouquet, S. Chemin, S. Guenau, B. Legeard, F. Peureux, M. Ut-  
584 ting, and N. Vacelet. BZ-testing-tools: A tool-set for test generation from Z and B  
585 using constraint logic programming. In *Proceedings FATES*, pages 105–120, 2002.  
586 Technical Report, INRIA.
- 587 12. F. Badeau and M. Doche-Petit. Formal data validation with Event-B. *CoRR*,  
588 abs/1210.7039, 2012. Proceedings of DS-Event-B 2012, Kyoto.
- 589 13. D. Basile, M. H. ter Beek, A. Fantechi, S. Gnesi, F. Mazzanti, A. Piattino, D. Tren-  
590 tini, and A. Ferrari. On the industrial uptake of formal methods in the railway  
591 domain. In *Proceedings iFM*, volume 11023 of *LNCS*, pages 20–29. Springer, 2018.
- 592 14. J. Bendisposto, S. Krings, and M. Leuschel. Who watches the watchers: Validating  
593 the ProB Validation Tool. In *Proceedings F-IDE*, volume 149. EPTCS, 2014.
- 594 15. J. Bendisposto and M. Leuschel. A Generic Flash-Based Animation Engine for  
595 ProB. In *Proceedings B*, volume 4355 of *LNCS*, pages 266–269. Springer, 2007.
- 596 16. M. Benveniste. On Using B in the Design of Secure Micro-controllers: An Experi-  
597 ence Report. *ENTCS*, 280:3–22, 2011.
- 598 17. J. Bicarregui, J. S. Fitzgerald, P. G. Larsen, and J. C. P. Woodcock. Industrial  
599 practice in formal methods: A review. In A. Cavalcanti and D. Dams, editors,  
600 *FM 2009: Formal Methods, Second World Congress, Eindhoven, The Netherlands,*  
601 *November 2-6, 2009. Proceedings*, volume 5850 of *Lecture Notes in Computer Sci-*  
602 *ence*, pages 810–813. Springer, 2009.
- 603 18. O. Boite. Méthode B et Validation des Invariants Ferroviaires. Master’s thesis,  
604 Université Denis Diderot, 2000. Mémoire de DEA de logique et fondements de  
605 l’informatique.
- 606 19. O. Boite. Automatiser les preuves d’un sous-langage de la méthode B. *Technique*  
607 *et Science Informatiques*, 21(8):1099–1120, 2002.
- 608 20. F. Bouquet, B. Legeard, and F. Peureux. CLPS-B - a constraint solver for B. In  
609 *Proceedings TACAS*, volume 2280 of *LNCS*, pages 188–204. Springer, 2002.
- 610 21. L. Burdy and J.-M. Meynadier. Automatic refinement. *Proceedings BUGM*, 1999.
- 611 22. M. Butler. Decomposition Structures for Event-B. In *Proceedings IFM*, volume  
612 5423 of *LNCS*, pages 20–38. Springer, 2009.
- 613 23. M. J. Butler, J. Colley, A. Edmunds, C. F. Snook, N. Evans, N. Grant, and  
614 H. Marshall. Modelling and refinement in CODA. In J. Derrick, E. A. Boiten,  
615 and S. Reeves, editors, *Proceedings 16th International Refinement Workshop, Re-*  
616 *fine@IFM 2013, Turku, Finland, 11th June 2013*, volume 115 of *EPTCS*, pages  
617 36–51, 2013.

- 618 24. M. J. Butler, D. Dghaym, T. Fischer, T. S. Hoang, K. Reichl, C. F. Snook, and  
619 P. Tummeltshammer. Formal Modelling Techniques for Efficient Development of  
620 Railway Control Products. In *Proceedings RSSRail*, volume 10598 of *LNCS*, pages  
621 71–86. Springer, 2017.
- 622 25. E. CENELEC. Railway Applications: Communications, Signalling and Processing  
623 Systems. Software for Railway Control and Protection Systems. *EN50128: 2001*,  
624 2001.
- 625 26. ClearSy. *Atelier B, User and Reference Manuals*. Aix-en-Provence, France, 2009.  
626 Available at <http://www.atelierb.eu/>.
- 627 27. M. Comptier, D. Déharbe, J. M. Perez, L. Mussat, P. Thibaut, and D. Sabatier.  
628 Safety Analysis of a CBTC System: A Rigorous Approach with Event-B. In *Pro-*  
629 *ceedings RSSRail*, volume 10598 of *LNCS*, pages 148–159. Springer, 2017.
- 630 28. M. Comptier, M. Leuschel, L. Mejia, J. M. Perez, and M. Mutz. Property-Based  
631 Modelling and Validation of a CBTC Zone Controller in Event-B. In *Proceedings*  
632 *RSSRail*, volume 11495 of *LNCS*, pages 202–212, 2019.
- 633 29. D. Dghaym, M. Dalvandi, M. Poppleton, and C. F. Snook. Formalising the hybrid  
634 ERTMS level 3 specification in iuml-b and event-b. *Int. J. Softw. Tools Technol.*  
635 *Transf.*, 22(3):297–313, 2020.
- 636 30. D. Essamé and D. Dollé. B in Large-Scale Projects: The Canarsie Line CBTC  
637 Experience. In *Proceedings B*, volume 4355 of *LNCS*, pages 252–254. Springer,  
638 2007.
- 639 31. N. Evans and W. Ifill. Hardware verification and beyond: Using b at awe. In J. Jul-  
640 liand and O. Kouchnarenko, editors, *B 2007: Formal Specification and Development*  
641 *in B*, pages 260–261, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- 642 32. J. Falampin, H. Le-Dang, M. Leuschel, M. Mokrani, and D. Plagge. Improving  
643 Railway Data Validation with ProB. In *Industrial Deployment of System Engi-*  
644 *neering Methods*, pages 27–43. Springer, 2013.
- 645 33. J. S. Fitzgerald, J. Bicarregui, P. G. Larsen, and J. Woodcock. Industrial de-  
646 ployment of formal methods: Trends and challenges. In A. B. Romanovsky and  
647 M. Thomas, editors, *Industrial Deployment of System Engineering Methods*, pages  
648 123–143. Springer, 2013.
- 649 34. D. Hansen, M. Leuschel, D. Schneider, S. Krings, P. Körner, T. Naulin, N. Nayeri,  
650 and F. Skowron. Using a Formal B Model at Runtime in a Demonstration of the  
651 ETCS Hybrid Level 3 Concept with Real Trains. In *Proceedings ABZ*, volume  
652 10817 of *LNCS*, pages 292–306. Springer, 2018.
- 653 35. D. Hansen, D. Schneider, and M. Leuschel. Using B and ProB for Data Validation  
654 Projects. In *Proceedings ABZ*, volume 9675 of *LNCS*, pages 167–182. Springer,  
655 2016.
- 656 36. L. Ladenberger, J. Bendisposto, and M. Leuschel. Visualising Event-B Models with  
657 B-Motion Studio. In *Proceedings FMICS*, volume 5825 of *LNCS*, pages 202–204.  
658 Springer, 2009.
- 659 37. L. Ladenberger and M. Leuschel. BMotionWeb: A Tool for Rapid Creation of  
660 Formal Prototypes. In *Proceedings SEFM*, volume 9763 of *LNCS*, pages 403–417.  
661 Springer, 2016.
- 662 38. J.-L. Lanet. The use of B for Smart Card. In *Proceedings FDL*, 2002.
- 663 39. T. Lecomte. The CLEARSY Safety Platform. [https://www.clearsy.com/en/](https://www.clearsy.com/en/our-tools/clearsy-safety-platform/)  
664 [our-tools/clearsy-safety-platform/](https://www.clearsy.com/en/our-tools/clearsy-safety-platform/). Accessed: 2020-01-21.
- 665 40. T. Lecomte. Applying a Formal Method in Industry: A 15-Year Trajectory. In  
666 *Proceedings FMICS*, volume 5825 of *LNCS*, pages 26–34. Springer, 2009.
- 667 41. T. Lecomte. Return of Experience on Automating Refinement in B. In *Proceedings*  
668 *SETS*, 2014.



- 669 42. T. Lecomte. *Developing Safety Critical Applications*. CLEARSY Systems Engi-  
670 neering, 2019. Accessed: 2020-01-21.
- 671 43. T. Lecomte, L. Burdy, and M. Leuschel. Formally Checking Large Data Sets in the  
672 Railways. *CoRR*, abs/1210.6815, 2012. Proceedings of DS-Event-B 2012, Kyoto.
- 673 44. T. Lecomte, T. Servat, G. Pouzancré, et al. Formal methods in safety-critical  
674 railway systems. In *Proceedings SBMF*, pages 29–31, 2007.
- 675 45. M. Lee and I. H. Sørensen. B-tool. In *Proceedings VDM*, volume 551 of *LNCS*,  
676 pages 695–696. Springer, 1991.
- 677 46. M. Leuschel, J. Bendisposto, I. Dobrikov, S. Krings, and D. Plagge. From Ani-  
678 mation to Data Validation: The ProB Constraint Solver 10 Years On. In *Formal*  
679 *Methods Applied to Complex Systems: Implementation of the B Method*, chapter 14,  
680 pages 427–446. Wiley ISTE, 2014.
- 681 47. M. Leuschel and M. J. Butler. ProB: an automated analysis toolset for the B  
682 method. *STTT*, 10(2):185–203, 2008.
- 683 48. M. Leuschel, J. Falampin, F. Fritz, and D. Plagge. Automated Property Verifica-  
684 tion for Large Scale B Models. In *Proceedings FM*, volume 5850 of *LNCS*, pages  
685 708–723. Springer, 2009.
- 686 49. M. Leuschel, J. Falampin, F. Fritz, and D. Plagge. Automated property verification  
687 for large scale B models with ProB. *Formal Asp. Comput.*, 23(6):683–709, 2011.
- 688 50. I. Maamria and M. Butler. Rewriting and Well-Definedness within a Proof System.  
689 In *Proceedings PAR*, volume 43. EPTCS, 2010.
- 690 51. I. Maamria, M. Butler, A. Edmunds, and A. Rezazadeh. On an Extensible Rule-  
691 Based Prover for Event-B. In *Proceedings ABZ*, volume 5977 of *LNCS*, pages  
692 407–407. Springer, 2010.
- 693 52. A. Mashkoor, F. Kossak, and A. Egyed. Evaluating the suitability of state-based  
694 formal methods for industrial deployment. *Softw. Pract. Exp.*, 48(12):2350–2379,  
695 2018.
- 696 53. C. Metayer. AnimB website.
- 697 54. A. R. Randolph Bergelehner, Ibtihel Cherif. An Approach to Improve SysML Railway  
698 Specification using UML-B and EVENT-B. Poster presented at RSSRail 2019,  
699 Newcastle, GB.
- 700 55. A. Rasheeq. An Approach To Improve SysML Railway Specification Using UML-B  
701 And Event-B. Master’s thesis, Frankfurt University of Applied Sciences, 2019.
- 702 56. K. Robinson. The B method and the B toolkit. In *Proceedings AMAST*, volume  
703 1349 of *LNCS*, pages 576–580. Springer, 1997.
- 704 57. D. Sabatier. Using Formal Proof and B Method at System Level for Industrial  
705 Projects. In *Proceedings RSSRail*, volume 9707 of *LNCS*, pages 20–31. Springer,  
706 2016.
- 707 58. T. Servat. BRAMA: A New Graphic Animation Tool for B Models. In *Proceedings*  
708 *B*, volume 4355 of *LNCS*, pages 274–276. Springer, 2007.
- 709 59. L. Voisin and J.-R. Abrial. The Rodin Platform Has Turned Ten. In *Proceedings*  
710 *ABZ*, volume 8477 of *LNCS*, pages 1–8. Springer, 2014.
- 711 60. N. S. Voros, C. F. Snook, S. Hallerstede, and K. Masselos. Embedded system  
712 design using formal model refinement: An approach based on the combined use of  
713 UML and the B language. *Design Autom. for Emb. Sys.*, 9(2):67–99, 2004.
- 714 61. M. Werth and M. Leuschel. VisB: A lightweight tool to visualize formal models  
715 with SVG graphics. In *Proceedings ABZ 2020*, LNCS, 2020. to appear.
- 716 62. J. Woodcock, P. G. Larsen, J. Bicarregui, and J. S. Fitzgerald. Formal methods:  
717 Practice and experience. *ACM Comput. Surv.*, 41(4):19:1–19:36, 2009.
- 718 63. F. Yang, J. Jacquot, and J. Souquères. JeB: Safe simulation of Event-B models  
719 in javascript. In *Proceedings APSEC, Volume 1*, pages 571–576. IEEE, 2013.