Formally Checking Large Data Sets in the Railways

Article · October 2012

Source: arXiv

CITATIONS
28

READS
418

3 authors, including:

Thierry Lecomte
ClearSy System Engineering
54 PUBLICATIONS **418** CITATIONS

SEE PROFILE

Michael Leuschel
Heinrich-Heine-Universität Düsseldorf
334 PUBLICATIONS **4,852** CITATIONS

SEE PROFILE

Formally Checking Large Data Sets in the Railways

Thierry Lecomte, Lilian Burdy[1], Michael Leuschel[2], Fernando Mejia[3]

[1] ClearSy,
Aix en Provence, France
Thierry.lecomte@clearsy.com

[2] Formal Mind,
Dusseldorf, Germany
leuschel@cs.uni-duesseldorf.de

[3] Alstom Transportation Systems,
Saint Ouen, France
luis-fernando.mejia@transport.alstom.com

Abstract. This article presents industrial experience of validating large data sets against specification written using the B / Event-B mathematical language and the ProB model checker.

Keywords: B mathematical language, ProB model checker, data validation.

1 Introduction

Historically, the B Method [1] was introduced in the late 80's to design correctly safe software. Promoted and supported by RATP[1], B and Atelier B, the tool implementing it, have been successfully applied to the industry of transportation. Figure 1 depicts the worldwide implementations of the B technology for safety critical software, mainly as automatic pilots for metros. Today, Alstom Transportation Systems, Siemens Transportation Systems and Technicatome-Areva are the main actors in the development of B safety-critical software. They share a product-based strategy and reuse as much as possible existing B models to develop future metros.

A more widely scope use of B appeared in the mid '90s, called *Event-B* [2], to analyse, study and specify not only software, but also whole systems. *Event-B* has been influenced by the work done earlier on Action Systems by the Finnish School (Action System however remained an academic project). *Event-B* is the synthesis between B and Action System. It extends the usage of B to systems that might contain software but also hardware and pieces of equipment. In that respect, one of the outcome of *Event-B* is the proved definition of systems architectures and, more generally, the proved development of, so called, "system studies" [7][8][9][10][11], which are performed before the specification and design of the software. This

[1] Régie Autonome des Transports Parisiens : operates bus and metro public transport in Paris

enlargement allows one to perform failure studies right from the beginning in a large system development. *Event-B* has been used to perform system level safety studies in the Railways [12], allowing to formally verify part of the whole system specification, hence contributing to improve the overall level of confidence of the railways system being built.

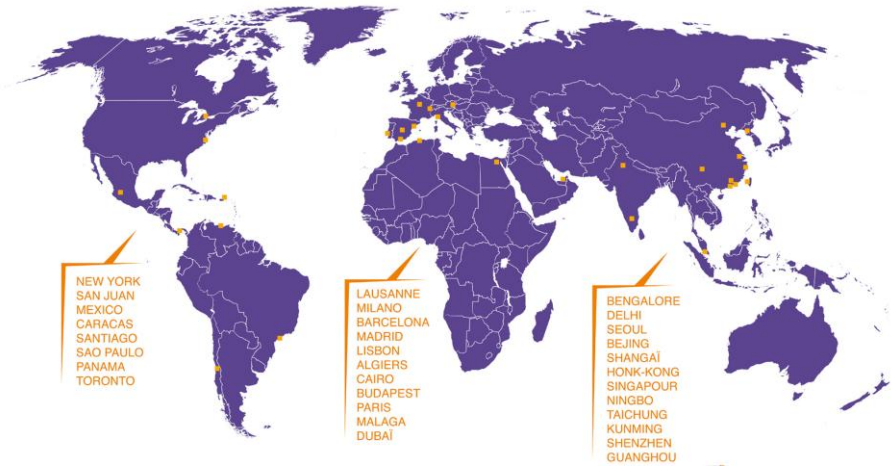

Fig. 1. Worldwide implementations (2012) of systems embedding software generated from B models.

However, if the verification of Event-B system specification or B software specification is quite easily reachable by semi-automated proof[2], verifying embedded data against properties[3] may turn to be a nightmare in case of large data sets. For the Meteor metro (line 14, Paris), software and data were altogether in a B project [5]. Demonstrating data correctness regarding expected properties was really difficult as it requires to iterate over large sets of variables and constants (and their domains) and the Atelier B main theorem prover[4] is not designed for this activity [6], that requires more a model checker than a theorem prover. Later on, software and data started to be developed and validated within two different processes, in order to avoid a new compilation if the data are modified but not the software. Data validation started to be entirely human, leading to painful, error-prone, long-term activities (usually more than six months to manually check 100 000 data against 200 rules)

In this article, we present a formal approach, based on the B/Event-B mathematical language and other ProB model-checker, designed and experimented by Alstom for the validation of railways data.

---

[2] Automatic theorem provers usually demonstrate 90-95 % of « well written" B models, the remaining has to be demonstrated during interactive sessions with the tool.

[3] In the case of a metro, these data may represent the topology of the tracks, the position of the signals, switches, etc.

[4] That is used by both Atelier B and Rodin platform

2 The Genesis

Verifying railways systems covers many aspects and requires a large number of cross-verifications, performed by a wide range of actors including the designer of the system, the company in charge of its exploitation, the certification body, etc. Even if complete automation is not possible, any automatic verification is welcome as it helps to improve the overall level of confidence. Indeed a railways system is a collection of highly dependent sub-system specification and these dependencies need to be checked. They may be based on railways signalling rules (that are specific to every country or even every company in a single country), on rolling stock features (constant or variable train size or configuration) and exploitation conditions.
In France, AQL RATP laboratory initiated the development of a generic tool, OVADO[5], to verify trackside data for the metro line 1 in Paris that is being automated[6]. This tool, based on PredicateB predicate evaluator[7], is able to parse data (XML, csv or text-based formats), load rules and verify that these rules are compliant with the data. Initially tested on line 13 configuration data, the tool has been able to check 400 definitions and 125 rules in 5 minutes (see Fig 2).

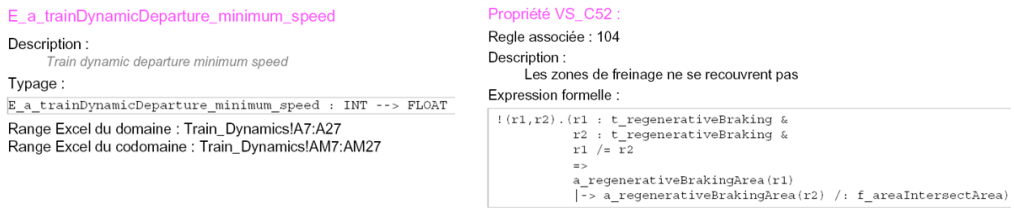

Fig. 2. Example of data definition and property

However the PredicateB tool is just a calculator able to manipulate B/Event-B mathematical language predicates: he is not able to find all possible values for any non-deterministic substitution or to find all counter-examples. Moreover the way the errors are displayed may lead to difficult analysis when the faulty predicate is complex.

During the DEPLOY project[8], the University of Dusseldorf and Siemens Transportation Systems have elaborated a new approach, based on the ProB model checker to dramatically reduce validation duration from about six months to some minutes [3][4]. Data are extracted from ADA source code and properties come from B models. In the case of the San Juan project, 79 files with a total of 23,000 lines of B are parsed to extract 226 properties and 147 assertions. The verification took 1017 seconds and led to the discovery of 4 false formulas. ProB was then experimented

[5] Tool for checking the B properties on railway invariants, initially developed by ClearSy
[6] A specific tool, initially developed for validating line 14 data, representing more than 300 000 lines of C++ code, was too difficult to maintain and to adapt to other lines. It was not reused for other lines.
[7] Hosted by the Rodin SVN Sourceforge service (http://rodin-b-sharp.svn.sourceforge.net)
[8] http://www.deploy-project.eu/

with great success on several projects: Roissy Charles de Gaule airport shuttle, Barcelona line 9, San Paulo line 4, Paris line 1 and Algiers line 1. At that occasion, ProB was slightly improved in order to deal with large scale problems and well validated in order to ease its acceptance by a certification body. However analysing false properties remains difficult (see fig 3.).

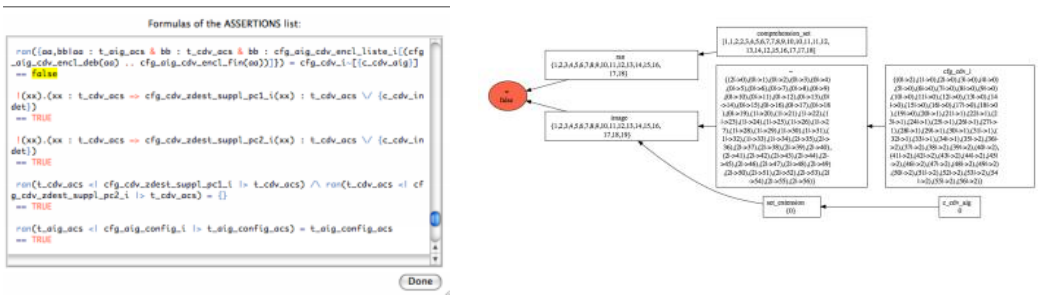

Fig. 3. A false property and its graphical representation

3 DTVT

Alstom Transportation Systems decided to experiment a new approach by reusing successful features of previous experiments. A new tool, DTVT, is defined and implemented. Its structure is presented on figure 4.

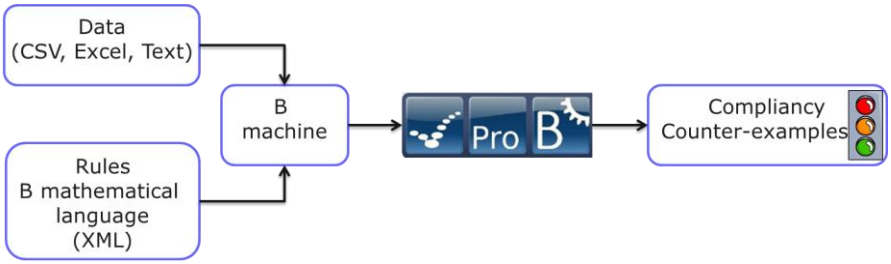

Fig. 4. DTVT tool structure

Input data are in csv format. Data are identified through their container file and their name. For example, *Curvatures_Cap!BeginValueCm* refers to the variable *BeginValueCm* in the file *Curvatures_Cap.xls* (see figure 5).

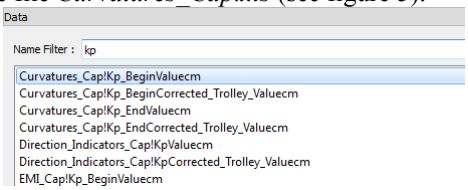

Fig. 5. Example of data declaration

Supported basic types are INT, BOOL and STRING. Data are sequences of these basic types. Values are extracted from xls files (see figure 6, the positions are expressed in centimeters).

Curvatures_Cap!Kp_BeginValuecm

```
Curvatures_Cap!Kp_BeginValuecm: seq(INT) =
[402240,416643,428132,450065,458742,491023,554622,594304,603370,616067,649696,674063,736472,771470,802893,875138,943344,1080130,1117338,1163438,1188794,1217486,1225136,1249441,1325
231,1389276,1420621,1424969,1444460,1479132,1515969,1561282,1602306,1640197,1674053,1715161,1727417,1747674,1807331,1894910,1942644,1963763,1983824,402435,416578,430283,447999,454
537,458162,492882,554871,597210,616579,649494,673761,737091,771512,802136,875196,942226,1080535,1117756,1162718,1188675,1218116,1225730,401260,1250376,413028,1324875,452109,138881
4,454569,1420158,1027313,1424509,1048847,1443998,1479259,1514980,1560912,1602616,1637622,1674770,1715291,1728021,1747768,1807487,1894901,1942335,1964763,1983825,1989334,2013367,20
58321,2075505,2120694,2125943,2185320,2254562,2261949,2306482,2346413,2368876,2404410,2498757,2556685,2580410,2634032,2654261,2717628,2727890,2764850,2771385,2790928,2856675,28985
26,2917115,1989334,2013421,2058344,2075834,2124341,2125943,2185320,2254562,2261949,2306479,2346413,2368876,2404381,2498785,2556697,2580397,2634032,2654261,2717629,2727889,2764874,
2771367,2790928,2851982,2898560,2917131]
```

Fig. 6. Example of data valuation

The verification rules are expressed using the B mathematical language and structured as B operations. Instead of having to deal with to large, quantified predicates, a verification rule is decomposed in small steps that allow displaying accurate error message helping to determine the source of the error.
A rule is composed of one or several COUNTERAMPLEs. COUNTEREXAMPLES are evaluated in the order they are defined. A COUNTEREXAMPLE is followed by a formatted message (%1, %2, %3, etc. represent the value of the first, second, third parameter of the following ANY substitution).
The ANY substitution allows to filter data or to calculate values. In the figure 7, the first rule computes the number of couples of the sequence ATC_Equipment_Type corresponding to the type "*Trackside OMAP*".
The ANY substitution is followed by an EXPECTED field. If the value calculated by the ANY substitution doesn't comply with this field, the error message is displayed with its parameters instantiated. In the figure 7, the error message of the second rule displays the value of *urbalisSectorID* (%1) and *idZC* (%3).

Rule_DB_ATCEQUIP_0007

```
/* For a given project, there shall be at most 10 local OMAP, and one Trackside OMAP shall be defined for each ZC. */

COUNTEREXAMPLE
  %1 local OMAP found
ANY
  nbLocOMAP
TYPE
  INT
WHERE
  nbLocOMAP = card(ATC_Equipments_Cap!ATC_Equipment_Type|>{"Trackside OMAP"})
EXPECTED
  nbLocOMAP <= 10
END;

COUNTEREXAMPLE
  ZC %1 has %3 defined trackside OMAP
ANY
  urbalisSectorID, idZC, nb, nameZC
TYPE
  STRING, INT, INT, STRING
WHERE
  idZC : ATC_Equipments_Cap!ATC_Equipment_Type~[{"ZC"}] &
  urbalisSectorID = ATC_Equipments_Cap!Urbalis_Sector_ID(idZC) &
  nameZC = ATC_Equipments_Cap!Name(idZC) &
  nb = card(dom(ATC_Equipments_Cap!Urbalis_Sector_ID|>{urbalisSectorID}) /\ dom(ATC_Equipments_Cap!ATC_Equipment_Type|>{"Trackside OMAP"}))
EXPECTED
  nb = 1
END
```

Fig. 7. Example of a verification rule

ProB is the central tool for the verification. It is modified in order to produce a file containing all counter examples detected (see figure 5) and slightly improved to better support some B keywords.

Fig. 6. Example of faulty verification: meaningful messages are generated for all counter examples

DTVT has been experimented with success on several ongoing developments (Mexico, Toronto, and Sao Paulo, Panama) to verify up to 50,000 Excel cells against up to 200 rules. A first round allowed defining required concepts, intermediate constructs (predicates used by several rules) and formalizing a set of generic rules that are shared by all projects. During the next rounds, specific project rules and data files were added. A complete verification is performed in about 10 minutes, including the verification report. The process is completely automatic and can be replayed without any human intervention when data values are modified.

5 Conclusion

Data validation appears to be of paramount importance in safety critical systems. The results obtained in this domain during the DEPLOY project have allowed to create and experiment with success on real scale projects a method for validating data against properties, based on the ProB model-checker.

References

1. Abrial, J.R. (1996) , *The B-book: Assigning programs to meanings,* Cambridge University Press
2. Abrial, J.R (2005)., *Rigorous Open Development Environment for Complex Systems: event B language*
3. Leuschel, Michael and Falampin, Jérôme and Fabian, Fritz and Daniel, Plagge (2009), *Automated Property Verification for Large Scale B Models*. In: *Proceedings FM 2009*. Springer-Verlag.
4. Michael Leuschel (2012), Formal Mind, ProB, ProR and Data Validation with B, FM'2012, Industray Day.
5. Patrick Behm , Paul Benoit , Alain Faivre , Jean-marc Meynadier (1999) , *Météor: A Successful Application of B in a Large Project*

6. Milonnet C. (1999) , *B Validation Book*; Internal document ref (Matra Transport International)
7. Sabatier, D. & al (2008), *FDIR Strategy Validation with the B method*, DASIA 2008
8. Hoffmann, S. & al (2007), *The B Method for the Construction of Micro-Kernel Based Systems*, ZB 2007
9. Sabatier, D. & al (2006), *Use of the Formal B Method for a SIL3 System Landing Door Commands for line 13 of the Paris subway*, Lambda Mu 15
10. Lecomte, T. (2008), *Safe and Reliable Metro Platform Screen Doors Control/Command Systems*, FM 2008
11. Lecomte, T. & al (2007), *Formal Methods in Safety Critical Railway Systems*, SBMF 2007
12. Sabatier, D., *Formal proofs for the NYCT line 7 (Flushing) modernization project*, DEPLOY Industry Day, Fontainebleau (2012)