PREUVE DE LA CORRECTION DES DONNÉES STATIQUES AVEC OVADO POUR LA LIGNE 13 DU MÉTRO PARISIEN

PROVING STATIC DATA CORRECTNESS USING OVADO FOR THE PARIS MÉTRO LIGNE 13

Robert Abo, Clément Houtmann and Pauline Humbert
Systerel
Les portes de l’Arbois, bât. A - 1090 rue Descartes
13857 Aix-en-Provence CEDEX 3

David Bonvoisin and Julien Ordioni
RATP - ING/STF/QS/AQL
54 rue Roger Salengro
94724 Fontenay-sous-Bois Cedex

Jean-Marc Mota
Thales - Campus Polytechnique
1 avenue Augustin Fresnel
91767 Palaiseau Cedex

Résumé
Les CBTC (Communication-Based Train Control) sont des systèmes sécuritaires complexes reposant sur des logiciels embarqués paramétrés par de grandes quantités de données de configuration. Ces données décryptent notamment les équipements de voie et les caractéristiques statiques du système. Pour vérifier la sûreté d’un tel système, il est primordial de s’assurer de la conformité de ces données de configuration vis-à-vis d’exigences de sûreté complexes. Le projet Ouragan L13, développé par la RATP et Thales, vise le déploiement d’un CBTC sur la ligne 13 du métro parisien. Dans cet article, nous décrivons le processus de validation des données de configuration de ce CBTC mis en œuvre par Systerel. Ce processus repose sur la plateforme de validation formelle OVADO et suit rigoureusement les recommandations des normes applicables (EN 50128, EN 50129). Nous montrons comment OVADO, en tant que plateforme générique et méthode formelle, est un atout majeur dans la mise en place et l’application de ce processus rigoureux de validation. Outre le projet Ouragan L13, d’autres projets de validation de données reposent aujourd’hui sur la plateforme OVADO, notamment à la RATP. Tous ces projets et notamment le projet Ouragan L13 démontrent qu’OVADO est aujourd’hui l’un des outils industriels le mieux adapté pour la validation de données de configuration.

Summary
CTBC (Communication-Based Train Control) systems are complex safety-critical systems which run embedded software configured by large amounts of configuration data. This configuration data describes trackside equipments and static features of the system. To ensure the safety of the overall system, it is crucial to validate the configuration data with respect to a series of complex safety requirements. The Ouragan L13 project, developed by RATP and Thales, aims at deploying such a CBTC on line 13 of the Paris Metro. In this article we describe the data validation process developed by Systerel for this CBTC system. This process relies on the OVADO platform—a formal data validation tool—and follows meticulously the recommendations of the applicable standards (EN 50128, EN 50129). We show that the generic and formal OVADO platform is a crucial asset in the development and the application of this rigorous data validation process. Besides the Ouragan L13 project, other data validation projects currently rely on the OVADO platform, notably within RATP. All these projects and specifically the Ouragan L13 project prove that OVADO has become one of the leading state of the art tools for industrial data validation.

Introduction
Railway signalling systems and more specifically Communication-Based Train Control systems (CBTC) are complex mission-critical systems, for which risk-management and dependability are crucial. Every day, the transportation of millions of people depends on such systems. The consequences of a failure can be truly dramatic and therefore their safety must be asserted with the highest assurance level.

Safety-related requirements for railway signalling applications are defined by international CENELEC standards EN 50126 (CENELEC, 1999), EN 50128 (CENELEC, 2011) and EN 50129 (CENELEC, 2003). EN 50126 addresses safety life-cycle issues for entire railway systems and methods to be applied to specify and demonstrate their reliability, availability, maintainability and safety (RAMS). EN 50129 depicts the approval process for railway signalling and communication systems. Finally EN 50128 focuses on methods (process and techniques) to be used at software level to guarantee some specific required Safety Integrity Level (SIL). Ranging from SIL0 to SIL4, these levels describe safety objectives, the highest level (namely SIL4) being assigned to the most safety-critical functions. Regarding the safety of critical functions (i.e. SIL3 and SIL4 functions), the use of formal methods is highly recommended by the standard. Indeed formal methods provide a wide range of techniques and tools to analyse and assert the safety of signalling systems: sophisticated modelling frameworks, model checking tools, automatic and interactive theorem provers, etc. Experience demonstrates that formal methods are efficient, reliable and can be cost-effective (Woodcock et al., 2009). The Paris Metro Line 13 ranks among the busiest metro lines in Paris, transporting more than 600,000 persons every day and more than 100 millions per year. To avoid overcrowding and chronic congestion, RATP and Thales are developing an automatic control system: the Ouragan CBTC. This railway signalling system is designed to manage and guide automatically trains through the line. The system consists in wayside and train-borne equipments communicating through a network. Each equipment runs applicable specific software parametrized with a specific set of configuration data describing tracks, stations, signals, junctions, train lengths, speed limits, etc.

The safety of the overall system equally relies on the correctness of both the software and the data. As for the CENELEC standard, substantial parts of the Ouragan CBTC software and data are associated with the SIL4 integrity level, denoting that an error would greatly jeopardize the safety of the whole system. In order to avoid such errors, the configuration data undergoes a specific validation process. This process is crucial towards asserting the safety of the whole CBTC system. It is emphasized in the
CENELEC standard EN 50128 which states that formal evidences shall be provided to assert the correctness and the safety of the configuration data.

In the case of the Ouragan L13 CBTC, safety for static data is ensured by means of a set of 919 requirements that data must fulfill. These requirements range from high-level system-wide requirements written in natural language to low-level bit-specific requirements written in mathematical notation. To tackle the problem of validating the CBTC configuration data with respect to this set of requirements, Systérel1—a company specialized in safety-critical embedded systems and formal methods—has defined and implemented the data validation process which fulfills the requisites of the CEONELEC standard and is described in this article.

The validation process relies on the OVADO platform2, a tool owned by RATP specifically designed to model safety requirements and to prove the correctness of configuration data. As for the CEONELEC standard, OVADO is T2 qualified for the validation of SIL4 configuration data, and relies on a 2-version dissimilar software: it includes two independent tool-chains each using a different model-checking paradigm for proving the formal properties of the static data. Since the start of its development in 2006, OVADO has been used to validate the configuration data of various metro lines by RATP and by Systérel (Abo & Voisin, 2013). Also OVADO has been a seminal tool in using the B and Event-B formalisms (Abrial, 1996, 2010) and the related tools for data validation projects (Leuschel et al., 2011; Lecomte et al., 2012; Badeau et al., 2012).

In this article, we describe the design and application of the data validation process used within the Ouragan L13 project to assert static data correctness. This static validation process relies on the innovative OVADO platform designed to formally prove the correctness of static data with respect to its high-level requirements. It includes formal modelling of the requirements, formal development peer review, fault-injection testing and validation. Let us add that this process does not address another important aspect of data validation: proving that this data really corresponds to trackside equipment items. This is done by dedicated teams, including surveyors. Describing this process is out of the scope of this paper.

Through the description of this process, we demonstrate how the OVADO platform is a crucial asset in validating static data correctness and therefore in reducing the risks for such a safety critical system.

This article is organized as follows. The first part introduces the context of this data validation project: section 1, 2 and 3 respectively describe the Ouragan L13 CBTC, its configuration data and the safety requirements associated with these data. The second part describes the OVADO approach: section 4 lists all the advantages of formal validation using a specialized tool such as OVADO; section 5 describes the tool itself. The third part describes the validation process: sections 6, 7 and 8 respectively describe the three parts of the overall validation process, namely modelling, verifications (including fault injection testing) and secure validation. The fourth part presents results in terms of scalability through a series of statistics about the Ouragan L13 CBTC data validation process, proving that OVADO offers enough maturity and scalability for industrial projects. The fifth part presents other OVADO projects by RATP, and the specific generic OVADO process used by RATP to qualify and validate the configuration data of its various CBTC systems. Finally the last part concludes the article with a thorough analysis of the advantages and disadvantages of the OVADO approach, and improvements are proposed both for the OVADO platform and for the validation process.

**Context: the Thales Ouragan L13 CBTC**

1 **The Ouragan L13 CBTC**

The Ouragan L13 CBTC is a project within Thales and RATP which aims at developing and deploying a fully automated CBTC on the Paris Metro Line 13. A CBTC is a railway signalling system which allows for partial or full automation of train management operations. It usually implements Automatic Train Protection (ATP) as well as Automatic Train Operation (ATO) and Automatic Train Supervision (ATS) through high-resolution train location determination; bidirectional train-to-wayside data communications; and train-borne and wayside computers. The Ouragan L13 CBTC is a distributed real-time system which relies on heterogeneous embedded systems communicating and interacting with one another. More importantly, it is a safety-critical system as it implements functions whose failures are likely to cause human casualties.

Among others, the Ouragan L13 CBTC includes the following equipment types.

**Zone Controllers** The purpose of these wayside equipments is to control trains moving in their specific area by reading inputs coming from signalling equipments and by communicating with trains.

**Train-borne Controllers** The purpose of these train-borne embedded equipments is to automatically drive the train through the route they are assigned by their respective zone controller.

2 **Configuration Data for Ouragan L13**

CBTC train-borne and wayside components are parameterized by a large amount of configuration data which describe the static parts of the system: tracks, signals, speed limitations, junctions, stations, etc. Some of these data are essential for the outcome of the automatic train management and most importantly its safety critical features. One can easily imagine the dramatic consequences of an error in these static data, even when the remaining software parts are totally error-free. In other words, static data are a safety critical part of the system. In the case of the Ouragan L13 CBTC, the configuration data are organized in several layers. The upper layer which is also called the system layer describes the entire line and all the objects associated with it. The system layer is created from measurements and positions of the different items on the track. The intermediate and equipment layers then represent subsequent rewritings of the system layer designed to format the data for the various wayside and train-borne equipments of the CBTC. The system layer contains 1.25 million single entries (a vast majority of 32-bit integers, 8-bit booleans and arbitrary length strings) and the total number of entries for all the layers is approximately 6 million entries for an overall size of 50MB of binary data. These layers are depicted in Figure 1.

3 **Requirements**

To ensure the safety of the system, configuration data must fulfill a series of 919 requirements which follow roughly the same organisation as data. A series of high-level requirements is associated with the system layer and enforces rules that are system-wide and independent from how data gets organised in the lower layers. Then series of requirements are associated with each equipment-specific lower layers. Each such series enforces rules that are equipment-specific and typically describe how the data is

---

1For further information, visit http://www.systerel.fr.
2Further information can be found at http://www.ovado.fr.
transformed into equipment-specific binaries. The intermediate layer allows for the transformation of system data into equipment-
specific data to be broken down into several simpler transformations. This also breaks down the complexity of the corresponding
requirements. For example, a first rule may describe how some intermediate binary file is created from the system layer; then a
second rule may describe how a second intermediate binary file is created from the first one; and finally a third rule may describe
how a final equipment-specific binary file is created from the last intermediate binary file. System requirements are much more
general. For example, the system requirement REG_SEG_08 describes a general rule for segment connection:

REG_SEG_08: If a segment is connected to four segments, then it is the toe of two points.

OVADO: the Formal Validation approach

4 Principles of Data Validation

In this section, we reflect on the reasons which make formal validation using a specialized tool, like OVADO the best choice among
the possible approaches to data validation.

4.1 Different ways of validating configuration data

In most cases, and more specifically for the Ouragan L13 CBTC, configuration data are generated using a dedicated software.
The approach described in this article differentiates the generation phase and a subsequent validation phase whose purpose is
to verify that the safety requirements hold on the data generated during generation phase. One could imagine another approach
without a separated validation phase where the safety requirements are directly enforced on the data generation software. But
then this generation software must be verified and qualified and this would significantly weigh down its development. Indeed if the
generation software is dedicated to a specific metro line (as for the Ouragan L13 CBTC), this verification and qualification effort
would be spent on a non-reusable software. On the contrary, having a separated validation allows to use an already qualified tool
dedicated to data validation such as OVADO. This tool is independent from the specifics of the system under consideration and can
be reused for any data validation project, therefore the qualification effort is done only once and shared for the unlimited number
of data validation projects that would use the validation platform.

4.2 Why Formal Validation?

Now let us focus on this separated validation phase. Three approaches are to be compared (Boulanger, 2013):

1. Manual validation;
2. Automated validation using a dedicated software implementing the specifics of safety requirements to be verified; and
3. Formal validation using a generic tool such as OVADO.

Manual validation leads to a tricky, fastidious, and long-term activity. Thus, in (Leconte et al., 2012), authors note that the process
of manually checking that one hundred thousand data items adhere to two hundred requirements usually lasts more than six
months, and that verifying the same rules automatically with some verification software takes only a few minutes. Considering the
amount of data to be validated for a project like Ouragan, and the number and complexity of safety requirements, we can easily
discard the idea of a manual verification. Some properties for the data validation of the Ouragan L13 project take almost an hour to
be verified on a high-end modern computer which performs billions of operations per second. It is obvious that such a verification
cannot be performed manually. In addition, the error rate in such a manual verification would be tremendously higher compared
to automatic computerized verification.

The second way to address data validation is to implement a dedicated verification software. In that case, each safety requirement
is implemented as a dedicated piece of software in some programming language. Again, the drawbacks are the same as with a
dedicated qualified generation software: costly qualification efforts will be spent on a non-reusable dedicated verification software.
Besides, the designers and implementers of this verification software will have to fully understand the safety requirements, the
verification algorithms, the structure of the configuration data and the programmatic internals of the verification software.
Finally, a much better solution is to automate data validation while separating the business activities of this process from its
implementation. One way to do this is to use a generic tool that takes the safety requirements and data as inputs, checks that this
data is valid against these requirements and finally outputs the results. This is the essence of OVADO. With a particularity: it is
a formal tool, in the sense that requirements are modelled as logical properties in a formal language –the B language as detailed
in the remainder of this paper. Their veracity is proved by the proof engines of OVADO. Hence, OVADO provides a very efficient
separation of concerns:

- The system engineer can focus on the safety requirements.
- The modeller can focus on their formal implementation in B as detailed in Section 6.
• The OVADO platform software engineer can focus on the generic algorithms and the programmatic internals included in the OVADO platform.

This modularity is a crucial asset for handling the complexity of a project like data validation for the Ouragan L13 CBTC. Finally, let us emphasize that since OVADO is a formal tool, it is backed by mathematical rigour. According to EN 50128 (CENELEC, 2011, D.28):

"Formal methods" refer to mathematically rigorous techniques and tools for the specification, design and verification of software and hardware systems.

Formal methods are recommended by EN 50129 and highly recommended by EN 50128 for designing SIL 3 and SIL 4 components. In particular for specifying requirements and designing (CENELEC, 2003, annex E) or modelling them (CENELEC, 2011, annex A). Thus, OVADO complies with the recommendations of standards for designing safety-related components.

4.3 Other existing tools
Since the early development of OVADO in 2006, the idea of applying the B formalism to data validation has influenced various other developments. Let us mention first RDV (Railway Data Validator) (Leuschel et al., 2011), developed by Siemens, which uses the B language and relies on the ProB animator and model checker (Leuschel & Butler, 2008). RDV was introduced as an improvement of earlier attempts by Siemens based on Atelier-B. RDV has been successfully tested against the configuration data and requirements of the following metro lines: San Juan, Algiers line 1, São Paulo line 4 and Paris line 1. Alstom’s Data Table Validation Tool (DTVT) (Lecomte et al., 2012) is yet another formal data validation tool using the B language. Similarly, this tool translates data and rules written in B language into B machines that are used as inputs for ProB. The tool establishes the compliance of data against the rules, or its non-compliance, yielding in that case an exhaustive list of counterexamples. Data is described in the CSV format while rules are translated into B operations. DTVT has also been successfully tested against the configuration data of some worldwide railway signalling projects: Mexico, São Paulo, Panama and Toronto. OVADO, RDV and DTVT all use the B language to formalise requirements and they rely on similar tool-chains. Let us stress that modelling in DTVT is oriented toward the search for counter-examples, whereas modelling in OVADO is oriented toward the validation of requirements. In the end, OVADO and DTVT achieve the same goal:

• In DTVT, a requirement is valid whenever no counter-example is found, otherwise, an exhaustive list of counter-examples is returned by the tool.

• In OVADO, a requirement is valid whenever the tool returns true, and the exhaustive list of counter-examples can be requested through the debugging interface.

Let us emphasize that to our knowledge, OVADO is the only platform which includes two complete dissimilar tool-chains. We believe that other generic formal methods and tools exist in the industry for the validation of large configuration data sets, but such tools are privately owned and no public information can be found about them.

5 The OVADO platform
The OVADO platform is an integrated tool designed to model safety requirements and to prove the correctness of configuration data. The tool is specifically designed for data validation projects, but is general and reusable: it can be applied to any data validation project, independently of the requirements or the data format. The tool is owned by RATP (Régie Autonome des Transports Parisiens, the main operator in charge of the public transports in Paris, France), which used it for the Paris’ metro lines #1, #3, #5 and #9, while Systerei is its current main developer and distributes it.

5.1 Data validation in practice
OVADO is used as follows. Prior to the analysis of data, safety requirements are transformed into formal properties. These properties are written in a subset of the B mathematical language (Abrial, 1996) as well as its evolution: the Event-B notation (Abrial, 2010). This subset is precisely the logical language of B/Event-B and is deeply rooted in first-order set theory and integer arithmetic. It makes an intensive use of standard mathematical notation, making it particularly easy to use for any engineer with a standard mathematical background. The OVADO platform can be extended through plug-ins and this is how OVADO reads project-specific data formats: for example for the Ouragan L13 CBTC project, a plug-in was developed to read binary data in the Ouragan format. The execution of a validation run is as follows: the core of the platform reads the provided data and validates it with respect to the properties. For each property, the platform outputs a result which may be either

ture when the provided data is correct with respect to this property;
false when the provided data violates the property;
error in case of some syntactic error in the property, some misconfiguration of the platform or an exhaustion of its resources (timeout/memout).

5.2 Simplified Architecture
The simplified architecture of OVADO is illustrated by Figure 2. A reader unfamiliar with the principles of dependability might be astonished by seeing that its architecture is redundant: two tool-chains doing the same thing. This section aims at briefly explaining the reasons of this choice. OVADO integrates two tool-chains based on two tools, PredicateB and ProB, that are completely dissimilar: each tool-chain contains a specific B parser, a specific data reader front-end and implements a specific verification algorithm. Each component relies on a different algorithm and is written using a different programming paradigm and programming language. For example, the logical core verifying the properties in the first tool-chain (PredicateB) uses a forward algorithm written in Java, whereas the logical core in the second tool-chain (ProB) relies on a constraint-based backward algorithm written in Prolog. Thus, the fundamental redundancy principle of dependability (Laprie, 1996; CENELEC, 2011, 2003) is ensured. Let us illustrate this point by quoting the EN 50129 standard (CENELEC, 2003, Section B.3.2):
The system/sub-system design shall be arranged to minimise potentially hazardous consequences of loss-of-independence caused by, for example, a systematic design fault, if it could exist.

Every time the verification of a property is executed during a validation run, both tool-chains are executed and the two obtained results are expected to be the same. The dissimilarity of the tool-chains greatly reduces the probability of obtaining an erroneous validation result because of some systematic software bug in the OVADO platform.

5.3 Qualification and Independent Assessment of OVADO

Initially developed by ClearSy and RATP, OVADO is now developed, maintained and distributed by RATP and Systerel. Thanks to its knowledge of the OVADO’s internals, Systerel has been able to qualify OVADO against the EN 50128:2011 standard. According to this standard, OVADO is required to qualify as a T2 class tool dealing with specific application data to be embedded into a SIL4 (safety critical) train control system. To obtain such a qualification, it must be proved that OVADO fulfils a set of requirements defined by the standard. Systerel has provided such a proof and demonstrated that OVADO could be used in a high safety integrity level process by means of two diversified tool-chains with no common mode failure. The qualification report gathers all the evidences requested by the standard. On that base, RATP has performed an assessment of this qualification to check the compliance with the EN 50128:2011 standard, the relevance of the engaged technical solutions and the accuracy of the non common mode failure demonstration. These activities have led to establish a list of safety application conditions (SAC) to be fulfilled when using OVADO in a high safety integrity level process.

Ouragan L13 CBTC Data Validation Process

The data validation process, illustrated by Figure 3, is divided into the three following subsequent phases: modelling, verification and validation. Let us now describe these three phases.

4 As defined in (CENELEC, 2011), a T2 tool is a tool which “supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software.” Configuration data is considered as part of the executable code, since it is embedded in the software application.

5The use of a T2 class tool must be justified. This justification will include potential failures identification and the safety barriers to manage or avoid them. Also A T2 class tool must be developed using adequate configuration management techniques. Each modification available in a new release of the tool must be justified. Only justified version can be used. A T2 class tool specification must be provided. Such a specification should depict the behaviors of the tool and the constraints affecting its use…
6 Modelling

The purpose of the modelling phase is to formalize requirements as formulas (also called properties) which OVADO will eventually check against the actual data. To do so, the modeller uses the logical subsets of B and Event-B which are greatly inspired from standard mathematical notation. For example, Figure 4 defines some symbols and their B ASCII equivalent used for modelling:

In the model, data are represented as first-class B values such as integers, sets, relations or functions. The modeller is therefore provided with a unified abstract way to handle all potential sorts of data such as binary, XML, Excel or CSV. The modeller can also define intermediate constants, types, sets of objects and functions (written in the very same B specification language) which become available to the entire modelling team through a shared library. These definitions break down complexity and allow formula refactoring. The design of this library is also essential for long-term maintenance. Let us demonstrate this on the REG_SEG_08 example:

**REG_SEG_08:** If a segment is connected to four segments, then it is the toe of two points.

For example, on Figure 5, `seg_0` is connected to four segments and is the toe of two points whereas `seg_0` is connected to four segments but is the toe of only one point.

To model this requirement, the modeller will use three definitions:

- `t_seg` set of all segments
- `r_adj` adjacency relation associating with each segment its adjacent segments
- `f_point_toe_seg` function associating with each point the segment of its toe

We omit the actual definitions for these identifiers. The property for REG_SEG_08 is

\[
\forall s \in t_{seg}, \quad \text{card}(r_{adj}[s]) = 4 \implies \text{card}(f_{point\_toe\_seg}^{-1}[s]) = 2
\]

In this equation, `r_adj[s]` represents the set of segments which are connected to `s`, and `f_point_toe_seg^{-1}[s]` represents the set of points which have toe `s`. This formula is written in ASCII B as follows.

```plaintext
seg.(seg : t_seg &
    card(r_adj[seg]) = 4
    =>
    card(f_point_toe_seg^{-1}[seg]) = 2)
```

The reader should notice that OVADO provides the modeller with a specification language as opposed to a programming language. In OVADO, formal properties describe what should be verified, but not how. This is essential since it allows the modeller to focus on the formalization of requirements without worrying about the actual verification algorithms. Also this permits each tool-chain to implement a distinct algorithm to verify the property.

In the example above, the requirement is implemented as a single property. However, complex requirements can be broken down into several complementary properties. Let us also note that properties are often more precise than their corresponding requirement in natural language. First, the requirement may be formulated in an imprecise fashion. In addition to this, the modeller is allowed to implement a requirement with a strictly stronger property (in the logical sense), as long as this property is still verified by the actual data (otherwise validation for this property will result in a false negative).
7 Verification and Fault Injection Testing

Once a requirement is implemented as properties in the formal model, the implementation (i.e. the properties) undergoes the two following verification steps. First, the properties are proofread by an independent verification engineer. If a mistake is found in the model, or if any question or remark arises, the verification engineer reports it in writing to the modeller, who in turn can either correct the properties or answer the queries of the verification engineer. This process is repeated until all errors are corrected and all questions are answered. In the end, the two engineers (namely the modeller and the verifier) must agree on the fact that the properties correctly model the corresponding requirement. Subsequently, the properties undergo fault injection testing. An independent verification engineer writes for each property one or several tests which consists in erroneous data. Each property is then tested against these erroneous data and the verification engineer tests that the OVADO platform correctly detects these voluntary errors by outputting false as a result. Whenever the OVADO platform fails at detecting the error (i.e. it outputs either true or fail), then further analysis is conducted by the modeller to understand why:

- If an error in the corresponding property is found, it is corrected and another verification cycle starts: the property goes through proofreading and fault injection testing once again.
- Otherwise, an error should be found in the tests itself, which may be directly corrected by the verification engineer. Such errors are corrected until that OVADO successfully detects the voluntary error in the test data by outputting false as a result.

Two ways exist for voluntarily creating erroneous data: one can either create erroneous data from scratch, or insert errors in the real existing data. The second option is much easier as every single property usually relies on lots of data: for example, REG_SEG_08 relies on segments and points. Writing all these data from scratch is tedious whereas inserting a single error on existing data is usually quite simple. In the case of REG_SEG_08, the verification engineer created test data from nominal data through the following simple modification: a segment which is not the toe of any point is artificially connected to four other segments.

Let us stress the fact that fault injection testing alone does not provide any proof or guarantee that the properties are correct. In particular, no coverage criterion is ensured for tests with respect to the properties to be tested, aside from the simple property criterion which states that each property must be associated with at least one test. Let us recall that configuration data undergo a series of tests (the properties) which themselves undergo a series of tests (fault injection testing). The criteria for these second-order tests are less strict than the criteria for properties because properties directly test data which will be embedded in equipment, and because properties already undergo proofreading. The correctness of properties is rather ensured through proofreading. Still fault injection testing is complementary to proofreading. Its purpose is to detect subtle errors which may easily go unnoticed when reading some property. For example, a universally quantified property of the form \( \forall x, P(x) \Rightarrow Q(x) \) may always evaluate to true because of some inconsistency inside \( P(x) \). Indeed, in such a case, the set of elements to be verified is empty, and the property is trivially true. This may be hard to detect through proofreading if the definition of \( P(x) \) is complex and includes several other definitions. The error will mechanically be detected by fault injection testing since no test can make such a property false.

8 Secure Validation

The final phase in the overall validation process is the secure validation phase. It consists in running both OVADO tool-chains in an isolated and clearly identified environment. Secure validation takes place on a specific dedicated machine with the following specifications: it is a hyper-threaded quad-core 64-bit machine equipped with 24Go of RAM running Ubuntu 12.04.4 LTS (Linux x86_64). To be isolated from the specific software configuration of the machine, the validation takes place in a specific identified chrooted environment. The first step of a secure validation is the identification of all the software components used for the validation. These include:

- The archive containing the chrooted environment.
- The archive containing OVADO.
- The OVADO plug-ins needed for the Ouragan L13 validation.
- An archive containing the model and the data to be validated.

Each release of OVADO is identified by a specific MD5 checksum for the OVADO archive. This MD5 checksum is retrieved and checked against the provided archive. The SHA256 checksums for all the software components are computed and stored in a log file so that the actual perimeter of the validation is clearly identified. In addition to this, SHA256 checksums are also computed for each file in the archive containing the data and the model, and stored in the log file. The integrity of the configuration data is also ensured with checksums: data come with CRC32 checksums which are computed beforehand and are checked every time the integrity of the data must be asserted (for example when loading the data into an embedded equipment). In order to assert the integrity of the configuration data, OVADO can read each CRC and recompute it from the binary file. Data integrity is then expressed by a property stating the equality between the read CRC and the recomputed CRC. This property is verified before running the validation: the property must check the CRC of each configuration data file included for the validation. The chrooted environment is then installed on the machine. OVADO and the L13 plug-ins are installed in the chrooted environment. Then the archive containing the data and the model is decompressed in the chrooted environment. Finally both tool-chains are sequentially executed, therefore producing two results for each property in the model. After the execution of the two tool-chains, a verification engineer verifies the following points:

- The integrity of each software component used for the validation is checked against the reference. This is done using the SHA256 checksums computed at the beginning of the validation by recomputing the checksums from the reference software components.
- The integrity of each file in the model and the data is checked against the reference. This is done using the SHA256 checksums computed as for the software components.

6This modification is possible because adjacency is described independently from points in the actual data
7Intel® Core™ i7-4770 CPU running at 3.40GHz
A report is written summarizing all the results of the operations described above. It contains the checksums, the results of the validation for each tool-chain, and the results of the verifications performed by the verification engineer. Finally, the logs, the software components, the results and the report are archived in a safe place. This allows for future replays of all the steps in the secure validation process, and is required for the reproducibility of the development and for the maintenance activities, as stated in the EN 50128 standard (CENELEC, 2011, 6.4.5.12).

Results and Scalability

This part presents statistics and results from the Ouragan L13 data validation projet.

Data

The system layer contains 1.25 million single entries (a vast majority of 32-bit integers, 8-bit booleans and arbitrary length strings) and the total number of entries for all the layers is approximately 6 million entries for an overall size of 50MB of binary data.

Requirements

In total, 919 requirements have been implemented into 1241 properties. Most requirements correspond to the system layer, the wayside zone controllers or the train-borne controllers.

<table>
<thead>
<tr>
<th>Component</th>
<th>Number of requirements</th>
<th>Number of properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>System layer</td>
<td>303</td>
<td>324</td>
</tr>
<tr>
<td>Zone controllers</td>
<td>270</td>
<td>402</td>
</tr>
<tr>
<td>Train-borne controllers</td>
<td>184</td>
<td>273</td>
</tr>
<tr>
<td>Others</td>
<td>162</td>
<td>242</td>
</tr>
<tr>
<td>Total</td>
<td>919</td>
<td>1241</td>
</tr>
</tbody>
</table>

Let us emphasize the high complexity of the requirements for the zone controllers: many requirements are more than 5 page long (the longest being 19 page long). This complexity explains the high ratio of properties against requirements.

Validation time

The execution of the OVADO tool-chains on all the properties and data takes a significant amount of CPU time. For example, the evaluation of all the 402 properties corresponding to zone controllers requirements on the dataset for a single zone controller takes almost 2 hours of CPU time with the PredicateB based tool-chain and almost 8 hours of CPU time with the ProB based tool-chain. This is clearly an acceptable validation time considering the large amount of data involved, the number of properties, their complexity, and the fact that the validation runs on nothing more than a high-end desktop computer.

These statistics about the Ouragan L13 data validation project are a clear demonstration that the OVADO tool is mature, scalable and ready for industrial data validation projects.

Related OVADO deployments by RATP

The OVADO platform is currently used on several other ongoing projects. Most of them are developed within RATP, relating to the RATP specific approach toward qualification and validation of configuration data. This final part of the article describes the RATP approach and its results.

9 Genesis of OVADO

Being an Independent Safety Assessor for its own projects, RATP has to use working methods as independent as possible from those used by its suppliers. For SACEM and METEOR pre-CBTC projects, RATP has thus developed two specific tools in order to be able to qualify/validate on its own the large amount of configuration data needed for these systems to get a representation of their physical environment. These tools had to comply with both the development process that the suppliers had used to generate the data and the design of the system itself. The development of such tools has finally proved to be tough and very expensive. In the beginning of the 2000s, RATP was about to purchase and deploy a new generation of CBTC (Metro lines #1, #3, #5, #9, #12, #13). In this context, developing project-specific tools was not relevant any longer. The idea to use a single process based on a single tool-kit to validate the configuration data sets on every project has then arisen. Such a single process implies the need for a versatile tool able to cope with the diversity of supplier development processes. This versatility was brought by the B formal language, due to its strong expressiveness and mathematical abstraction level. This had led to the development of the OVADO tool described above. OVADO has then become the cornerstone of a new generic process, using formal methods (for all the reasons quoted above) and applicable to all the RATP projects.

10 Process

This generic process is based on 2 kinds of verifications related to configuration data:

1. Fulfilment of Safety constraints. These properties are on the one hand identified by the supplier and, on the other hand, completed by RATP thanks to its capitalized knowledge and expertise about application data. Safety properties can either apply to the CBTC system configuration data or to the software configuration data. In both cases, these safety properties result from design choices made at different phases of the project.

2. Correct generation of software configuration data. These configuration data are produced from system level configuration data with a dedicated generation tool using algorithms that may be complex. The correction of generated software data is checked by:
   (a) Modelling the linking properties between the system level data and software level data. These properties result from the analysis of the way these data are used by the software.
   (b) Evaluating these linking properties (also called transformation requirements) with OVADO.
For both kinds of verification, and due to the need for independence from the supplier, RATP aims to draw its own understanding of constraints and transformations and model them into the non-ambiguous B language.

The RATP generic process can finally be summarized by Figure 6. In this figure:

- The round box on the left illustrates the supplier’s process.
- The grey boxes represent the documents and data used as input for the validation activity.
- The white boxes represent the formal properties (i.e. predicates), written by RATP in the B language, to be evaluated by OVADO.

The different steps of the process are:

1. Identify the needs (depending on technology used, development constraints and habits, technical constraints, etc.) concerning the constraints or transformations.
2. Reverse-engineer (as far as possible without using the supplier’s documentation):
   (a) The way to produce correct software data from system level data.
   (b) The safety properties that have to be fulfilled by both software level data and system level data.
3. Write formal predicates and properties and perform a formal evaluation (computation) of these properties with OVADO.
4. Analyse the counter-examples, if any.

11 Results

After a research and development period, the industrialisation of OVADO was successfully led through different big projects like line #1, #3, #5 and #9 CBTCs (automatic train control). Using OVADO all along these projects has allowed RATP to capitalize and formalize knowledge about CBTC configuration data and build a formal properties database. These formal predicates can then be instantiated for different projects with minimal work load and cost. Furthermore, this capitalization allows RATP to complement the verifications performed by the supplier in a very efficient way, improving the confidence in safety. Currently, this database contains approximately 1,000 properties and 1,000 definitions.

In the same way, regression analysis is very efficient and very quick with OVADO (“push button”). Only 1 to 2 weeks are needed to revalidate a complete set of CBTC data if there are no structural modifications of the system or software data. Moreover, using a unique data validation tool makes skills management much easier. Once the OVADO tool has been mastered, attention can be focused on the safety verifications themselves: RATP spends more time on its core business (i.e. safety assessment activities) than on developing, maintaining and debugging various project specific tools.

We can notice that RATP uses OVADO for non-safety purposes too. Indeed, OVADO, thanks to its formal approach, can be used to list exhaustively specific “functional” configurations like topological singularities, combining parameters, etc. Eventually, RATP also aims at widening the scope of OVADO, for instance like the validation of Computer Based Interlocking configuration data.

Conclusion

We have described the validation process for the configuration data of the Ouragan L13 CBTC, a safety-critical automatic train management and signalling system designed for the line 13 of the Paris Métro. This configuration data contains millions of entries to be checked against 919 elaborate requirements. The data validation process relies on the OVADO platform, a generic data validation tool which relies on the B formal language. OVADO is a crucial asset to handle such an industrial complexity and scale.

Through the article, we have insisted on the fact that our validation process rigorously complies with the CENELEC standards (CENELEC, 2011, 2003). Assets for this compliance include:

Figure 6. RATP generic process
• The two dissimilar tool-chains of the OVADO platform and its T2 qualification for validation of SIL4 configuration data.

• A careful validation process including several verification steps (proofreading and fault injection testing).

• A secure validation procedure with a specific emphasis on reproducibility.

We have described the internals of the OVADO platform and its advantages for proving the correctness of large data sets against complex requirements. Through a series of statistics and measurements, we have showed that the Ouragan L13 CBTC data validation project is a clear demonstration of the maturity and industrial strength of the tool. We have also described the process and ongoing projects within RATP which use OVADO to qualify large configuration data sets for CBTC systems. The OVADO platform therefore stands as a leading state-of-the-art tool for the validation of large industrial complex configuration data sets. The Ouragan L13 data validation project is one of the largest successful data validation projects based on the OVADO platform. During the development of all these projects, we have gained substantial experience toward the use of the platform. This leads us to propose the following improvements.

• During the modelling phase, the modeller shall test the performances of the properties he writes with both OVADO tool-chains. Indeed, each tool-chain is based on a different model-checking paradigm and therefore behaves quite differently. OVADO provides an interface which is rather focused on the first tool-chain and testing with the second tool-chain is burdensome. A better integration of both tool-chains in the user interface would lighten this task.

• A well-designed library containing essential definitions is a crucial asset to the modeller. To structure the development of such a library and disconnect the design of the library from its use, OVADO could provide mechanisms for encapsulation, namespaces and modules. Unit testing could also be applied to the definitions in this library.

• When extracting a set of counter-examples for some requirement which represent physical objects on the tracks, OVADO could provide a way to visualize such objects on a map, therefore giving a direct insight of which physical objects may present a threat to safety with respect to this specific requirement.

• Besides static configuration data, OVADO could be used to analyse other kinds of data, such as execution logs produced by CBTC equipments.

• Finally, let us mention RailML (Nash et al., 2004), yet another data format which could be provided one day to OVADO.

12 Acknowledgements

The authors wish to thank all the members of the teams which worked on the Ouragan L13 data validation project: Philippe Ayraut, Anne-Charlotte Derode, Jean-Yves Friant, Julien Guilhot, Myriam Haddad, Mohamed Zalim, Christophe Métaayer, Frédéric Badeau, Aghiles Adjaz, Benoît Lucet, Mohamed Belarsi, Emmanuel Billaud, Matthieu Carlier, Gladys Gbedo, Brice Gombault, Matthias Güdemann, Minh-Thang Khue, Guillaume Phavorin, Roméo Courbis, Olivier Hervillard, Christophe Logerot, Morgan Prigent, Thomas Navas, Julien Vanderstraeten, Emmanuel Nedelec, Thomas Muller, Nicolas Beauger, Sébastien Juran, Josselin Dolhen, Laurent Voisin and (last but not least) Gaëlle Visentin. We are also specifically grateful to Laurent Voisin and Vincent Pouzo for their helpful insights and remarks during the redaction of the article.

13 References


CENELEC. (2011). EN 50128: Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems.


