Formalization and validation of a subset of the European Train Control System

Conference Paper *in* Proceedings - International Conference on Software Engineering · June 2010

DOI: 10.1145/1810295.1810312 · Source: IEEE Xplore

Formalization and Validation of a subset of the European Train Control System*

A. Chiappini
European Railway Agency
Valenciennes, France
angelo.chiappini@era.europa.eu

A. Cimatti
Fondazione Bruno Kessler
Trento, Italy
cimatti@fbk.eu

L. Macchi
Registro Italiano Navale
Genova, Italy
luca.macchi@rina.org

O. Rebollo
European Railway Agency
Valenciennes, France
oscar.rebollo@era.europa.eu

M. Roveri, A. Susi, S. Tonetta
Fondazione Bruno Kessler
Trento, Italy
{roveri,susi,tonettas}@fbk.eu

B. Vittorini
Registro Italiano Navale
Genova, Italy
berardino.vittorini@rina.org

ABSTRACT

The European Train Control System (ETCS) is a control system for the interoperability of the railways across Europe.

In this paper, we report on the activities of the EuRailCheck project, promoted by the European Railway Agency, for the development of a methodology and tools for the formalization and validation of the ETCS specifications. Within the project, we achieved three main results. First, we developed a methodology for the formalization and validation of the ETCS specifications. The methodology is based on a three-phases approach that goes from the informal analysis of the requirements, to their formalization and validation. Second, we developed a set of support tools, covering the various phases of the methodology. Third, we formalized a realistic subset of the specification in an industrial setting. The results of the project were positively evaluated by domain experts from different manufacturing and railway companies.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications—*Methodologies*

General Terms

Languages, Verification

Keywords

Requirements validation, formal methods, methodology

*The activities described in this paper have been funded by the European Railway Agency under the project EuRailCheck, service contract ERA/2007/ERTMS/02. S. Tonetta has been supported by the Provincia Autonoma di Trento (project ANACONDA).

ICSE '10, May 2-8 2010, Cape Town, South Africa

1. INTRODUCTION

The European Train Control System (ETCS) is a project supported by the European Union aiming at the implementation of a common train control system in all European countries. This has been judged as a prerequisite for the uninterrupted movement of train across the borders, being the existing train control system in different countries totally incompatible. ETCS is already installed in important railway lines in different European countries (like Spain, Italy, The Netherlands, Switzerland) and installations are in progress in other countries, such as Sweden, UK, France, Belgium and also non-European railways such as China, India, Turkey, Arabia, South Korea, Algeria and Mexico.

Since 2005, the European Commission decided to give the European Railway Agency (ERA) the role of system authority for ETCS, with the responsibility of managing the evolution of the specifications (change control management), ensuring their consistency, and guaranteeing the backwards compatibility of new versions with the old ones.

In 2007, ERA issued a call to tender for the development of a methodology complemented by a set of support tools, for the formalization and validation of the ETCS specifications. The activity posed many hard problems. First, the ETCS documents are written in natural language, and may thus contain a high degree of ambiguity. Second, the ETCS specifications are still in progress, and receive contribution by many people with different culture and background. Third, the ETCS comprises a huge set of documents, and comes with severe issues of scalability.

In this paper, we report on the activities of the EuRailCheck project, that originated from the successful response to the call to tender by the consortium composed by "Registro Italiano Navale (RINA)", a railway certifying body, "Fondazione Bruno Kessler - irst", a research center, and "Dr. Graband and Partners", a railway consultancy company. Within the project, we achieved three main results. First, we developed a methodology for the formalization and validation of the ETCS specifications. The methodology is based on a three-phases approach that goes from the informal analysis of the requirements, to their formalization and validation. Second, we developed a set of support tools, covering the various phases of the methodology, based on the integration of algorithmic formal verification techniques within traditional design tools. Third, a realistic subset of the specifi-

cation was formalized and validated applying the developed methodology and tools.

The results of the project were then further exploited and validated by domain experts external to the consortium. The evaluation was carried out in form of a workshop, followed by hands-on training courses. These events were attended by experts from manufacturing and railways companies, who provided positive feedback on the applicability in the large of the methodology.

This paper is structured as follows. In Section 2, we detail the goals of the project. In Section 3, we describe the methodology. In Section 4, we overview the support tools. In Section 5, we report the results of the project. In Section 6, we discuss the lessons learned in the experience. In Section 7, we discuss some related work. In Section 8, we draw some conclusions and outline the future work.

2. GOALS OF THE PROJECT

High-quality specifications are of paramount importance for the development process. Flaws in the requirements can lead to correct systems that do not do what they are supposed to. In the ETCS setting, manufacturers of ETCS components can use correct specifications to verify the compliance of the system being developed, and railway companies can use them for acceptance. The goal of the project is to provide a methodology and tools to analyze the ETCS specifications, and to eliminate various kinds of flaws. First, the methodology should support the identification and elimination of ambiguities, so that the requirements can be uniquely interpreted by the system designers. This is particularly important in the case of ETCS, since the requirements are to be applied in different cultural contexts such as the different national railway manufacturers. Second, it should be possible to detect inconsistencies in the specifications, that may invalidate the system verification, which is usually resource and time consuming. Finally, although ETCS is an available standard, its specification is in continuous progress for maintenance and evolution; thus, the methodology and tools should support early validation of extensions, modifications and maintenance of the current specifications.

In order to do this, we rely on formal methods, both on formal specification, which guarantees the absence of ambiguity, and on formal verification, in order to automate the analysis and the search for possible flaws. We remark that the project poses requirements that are significantly different from "more traditional" applications of formal methods. Traditionally, in formal verification, the entity under analysis is a design, that is to be verified against a set of requirements, assumed to be correct. Here, the purpose of the activity is the analysis of the ETCS requirements, independently from any system design. A standard check on formal requirements is to verify that they are consistent, but is not sufficient to assure that the requirements are correct, and that they specify exactly what the experts have in mind. Thus, the use of formal methods, traditionally oriented to design verification, has to be re-thought before being applicable.

The main issues in achieving these goals are that ETCS is a huge complex system, with many functional modules, which need to be clearly separated between on-board and track-side functionalities, and is contributed by a vast set of people. Many ambiguities come from the use of the natural language, from corrections applied by different people, with different cultures, of different mother tongues.

Finally, the methodology should meet some user needs in order to facilitate the adoption of the methodology by non experts in formal methods. In particular, the following non-functional requirements are important to make the methodology applicable in practice.

- The methodology should use a "light" language for the formalization avoiding intricate formalisms.
- The methodology should hide the interaction with the underlying verification tool.

3. METHODOLOGY OVERVIEW

In this Section we recall the main phases of our methodology, whose complete characterization is described in [9].

3.1 The methodology at a glance

In order to formalize the requirements, our approach exploits a subset of the Unified Modeling Language version 2 [1] (referred as UML from now on), and on a constraint language, based on a subset of the Property Specification Language (PSL) [13]. The constraint language mixes Linear Time temporal Logic (LTL) [24], regular expressions [4], first-order logic and hybrid aspects such as constraints on the continuous evolution of real-time and physical entities. It combines mathematical and English expressions for a natural correspondence with the textual requirements, resulting into a Controlled Natural Language (CNL), although the subset of handled English is minimal. In the following, we will refer to the constraints and the language as the CNL constraints and the CNL language.

Our methodology consists of the the following three main steps (also depicted in Fig. 1):

M1 *Informal analysis phase.* It consists of the categorization and structuring of the informal requirement fragments described in the requirements document to produce categorized requirement fragments.

M2 *Formalization phase.* The categorized requirement fragments are described through the set of concepts and diagrams in UML, and through additional constraints in the defined CNL to produce formalized requirement fragments.

M3 *Formal validation phase.* It consists of the definition of a series of validation problems, their automatic validation check, and the analysis of the results.

Each step of the methodology is supported by a specific tool.

3.2 Informal analysis phase

In the informal analysis, the set of requirements is first categorized on the basis of their characteristics, and some dependencies are imposed among them. In particular, we recognize the following steps:

M1.1 *Isolation and Categorization of the informal requirement fragments* to identify a requirement fragment of the domain requirements document and to categorize it according to a given taxonomy.

M1.2 *Creation of the dependencies among the informal requirement fragments.*

M1.3 *Analysis of the informal requirement fragments based on standard inspection-based requirements engineering.*

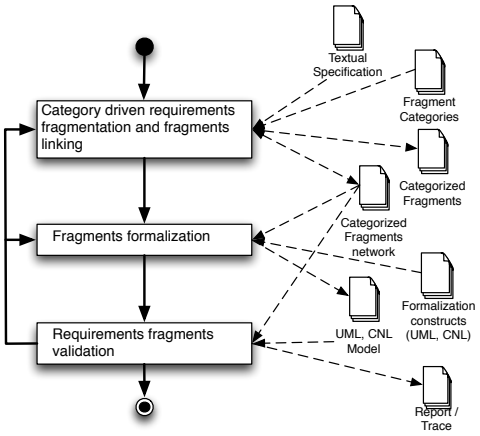

Figure 1: The methodology and the artifacts produced in the three phases.

Examples of the requirement fragments categories considered in step M1.1 are (see [9] for additional details):

- *Glossary*, to identify the fragments describing terms in the domain (e.g., the train, its position, the track);
- *Architecture* for the fragments describing particular parts of the ETCS domain (e.g., on-board subsystems, track-side subsystems);
- *Functional*, for the fragments mandating constraints on the authorized configurations and behaviors of the objects of the domain (e.g., the starting of the train);
- *Scenario*, for fragments describing an example of interaction among specific instances of the domain concepts (such as, an example of exchange of messages between two trains and one track-side radio device);
- *Property*, for the fragments describing a property which is expected to hold if the system fulfills the functional requirements (e.g, two trains cannot be in the same position at the same time).

The categories help the analyst in understanding the domain and are used in the next steps of the methodology to guide the formalization of the domain by suggesting the use of specific UML or CNL constructs. For the step M1.2 we identified three categories of dependencies:

- given two requirements we say that A *Strongly Depends* on B if A cannot exist without B;
- A *Weakly Depends* on B if A can exist without B but there is a link between them;
- A is a *Refinement* of B if A redefines some notions of B at a lower level of abstraction.

3.3 Formalization phase

In this phase, the categorized requirement fragments produced as artifact of the informal analysis phase are formalized. The formalization phase consists of the following steps:

M2.1 *Formalization of each requirement fragment* by specifying the corresponding UML concepts and diagrams, and/or the *CNL* constraints;

M2.2 *Linking the formal elements introduced in the previous step to the textual requirements*; the link is used for requirements traceability of the formalization against the informal textual requirements, and to select directly from the domain requirements document a categorized requirement fragment to be validated.

In particular, in the step M2.1, we adopted *classes* and *class diagrams*, to formalize the requirements that have been classified as Glossary, and CNL constraints to specify requirements that have been classified as Functional, Property, and Scenario, and to impose constraints on the class diagrams derived by the Glossary requirements. Moreover, we exploited *state machines* and *sequence diagrams* to model elements that in the specifications were represented by similar artifacts. In particular, state machines were used to formalize requirements classified as Functional, while *sequence diagrams* to represent those requirements classified as Scenario that describe the interaction among a set of objects. The selection of UML diagrams and concepts has been performed on the basis of the expressive power of the UML concepts and on the need related to the formalization of the UML constructs in a formal language. Details on the underlying object model extended with temporal constraints are given in [8].

3.3.1 UML

In our framework, a *class* represents a concept in the domain and is associated with a set of class attributes, representing the set of characteristics of the concept, and a set of class methods, representing actions/procedures the class can perform. *Relationships* among classes represent the relations existing between domain concepts. In our context we allow only for the *association*, *aggregation* and *generalization* UML relationships. We use *state machines* to model the behavior of the methods of a class in the domain. We restricted the syntax of the UML state machines, and in particular of their states, allowing only the representation of "simple" *states*, each representing a generic state of the corresponding class method, *initial state*, to represent the entry point of the corresponding class method, *final states*, each representing the termination of the corresponding class method, and *conditional states*, each representing conditional branches in the execution of the corresponding method. UML *sequence diagrams* model the evolution of the specified objects as a sequence of exchange of messages, focusing on the representation of their interactions. We use sequence diagrams to model Scenario requirements that are requested/expected to happen in the domain in terms of exchange of messages. The UML notation for the sequence diagram has been restricted only to *messages* that correspond to method calls of the object itself or of another object. We also allow the use of some *interaction operators* defined in UML that specify particular configurations of messages, such as: *negation* of messages configurations, *alternative* configurations, *option*, *parallel* and *loop*.

3.3.2 Constraints

In order to allow the specification of constraints and temporal properties of the entities in the model, we extended the UML model with a constraint language. CNL languages are well-defined subsets of natural language whose grammar has been restricted in order to be automatically processable (cfr. for instance [16]). In our choice, the language includes tem-

poral operators as in [22]. We proposed a grammar, based on the subset of the PSL [13] that mixes LTL operators with Regular Expressions (RELTL). This choice is motivated by the fact that this fragment, being linear time, is suitable to express constraints on the evolution of observable events of the system. The language has been extended with constraints on the continuous evolution of real-time and physical entities [10]. The grammar has been defined to include enough syntactic sugar to be easily accepted and then used by non-experts in formal methods or software engineering.

We have classified the constraints we use to annotate the UML concepts and diagrams in the following five categories:

- *Initial*, defining constraints that are valid at the beginning of any system behavior;
- *Invariant*, defining a constraint expected to be always valid over time;
- *Behavior*, defining a generic constraint formalizing Functional requirements;
- *Scenario*, defining a generic constraint formalizing Scenario requirements;
- *Property*, defining a generic constraint formalizing Property requirements.

An example of such CNL constraints could be the specification of the Property that: "two trains cannot be in the same position on the track at the same time", that can be expressed as follows:

for all Train t1, t2, **such that** t1 != t2 **then**
never(t1.position = t2.position)

where **for all**, **such that**, **then** and **never** are keywords of the CNL grammar.

3.4 Formal validation phase

The validation of the formalized requirement fragments aims at improving the quality of the requirements. This goal is achieved by performing several analysis steps, based on the use of formal techniques, which may help to pinpoint flaws that are not trivial to detect in an informal setting.

The formal validation phase of the methodology is accomplished as follows:

M3.1 *Check the well-formedness of the formalized requirement fragments.* This activity aims at verifying that the formalized requirement fragments syntactically adhere to the formal language syntax, and that all the elements mentioned in them have been previously defined.

M3.2 *Narrowing of the formalized requirement fragments.* This phase aims at focusing the validation to a particular subset of interest of the formalized requirement fragments (e.g. to restrict the validation to the elements of a specific module). In this phase the domain expert defines a set of *validation problems* consisting of a set of objects per each class and, possibly, a set of scenario constraints or a property to be checked.

M3.3 *Formal validation of the identified formalized requirement fragments.* The validation problems defined in M3.2 are passed to the automatic verification engine (see details below), and the domain expert analyzes the results.

The validation problems defined in the phase M3.2 can be of three types: consistency checking, scenario compatibility, and property checking.

3.4.1 Consistency checking

The formal notion of logical consistency can be intuitively explained as "freedom from contradictions". It is possible that two formalized requirement fragments mandate mutually incompatible behaviors. This check aims at formally verifying the absence of logical contradictions in the considered formalized requirement fragments. Therefore, it does not require any domain knowledge. Consistency checking is carried out by checking the satisfiability of the conjunction of the involved constraints.

3.4.2 Scenario compatibility

This check aims at verifying whether a scenario is admitted given the constraints imposed by the considered formalized requirement fragments. The check can be reduced to the problem of checking the satisfiability of the conjunction of the set of considered formalized requirement fragments and the constraints describing the scenario. Thus, if the scenario is compatible, we obtain a behavior trace compatible with both the considered formalized requirement fragments and with the constraint describing the scenarios. Otherwise, we obtain a subset of the considered formalized requirement fragments that prevents the scenario to happen.

3.4.3 Property checking

This check aims at verifying whether an expected property is implied by the considered formalized requirement fragments. This check is similar in spirit to Model Checking [11], where a property is checked against a model. Here the considered set of formalized requirement fragment plays the role of the model against which the property must be verified. When the property is not implied by the specification, a counterexample is produced. A counterexample is a behavior witnessing the violation of the property, i.e. a trace that is compatible with the considered formalized requirement fragment, but does not satisfy the property being analyzed.

Property checking can be reduced to the problem of checking the satisfiability of the conjunction of the considered formalized requirement fragments with the negation of the property. If this set is consistent, then a witness behavior compatible with the considered formalized requirement fragments and satisfying the negation of the property is produced. This behavior is a counterexample for the property. If such witness does not exist then the property holds.

3.4.4 Problem verification

As described above, the validation problems are reduced to a satisfiability problem for the specification language. The satisfiability problem is solved with the following two automatic steps:

1. the specification (including constraints and diagrams) is translated into an symbolically represented, infinite-state transition system;
2. the transition system is analyzed to look for an accepting path by applying standard model checking techniques for infinite-state systems; the path has a one-to-one correspondence with a model for the original specification.

3.4.5 Validation Loop

The above validation steps can find problems such as inconsistency, incompatibility with a scenario, or failure of a property. The results must by analyzed to discover if the issue was due to a wrong formalization (of the requirements or of the scenario/property) or if it is present also in the original informal specification. The validation steps can be iterated arbitrarily, by correcting formalized requirement fragments and/or the corresponding categorized requirement fragments if necessary, creating new scenarios, new properties, and by analyzing different aspects of the requirements specification. The narrowing phase M3.2 allows the domain experts to focus only on a subset of the formalized requirement fragments by selecting specific modules and consider only some of the functions of the selected module thus enabling for a *modular validation* approach. It also allows them to perform several kinds of *what-if analysis*, in particular, it allows them to check which properties and scenarios remain valid after adding/removing new formalized requirement fragments. Moreover, in the narrowing phase we can ignore the requirements with low-level details and consider the requirements at a higher level of abstraction, thus enabling for a *hierarchical verification* approach. This process results in a validation loop where every check increases the confidence of the domain expert in the correctness of the formalized requirement fragments.

4. TOOL SUPPORT

Within the project, a tool supporting the methodology has been designed and developed. Several requirements were taken into account, such as easy of use, and openness.

The technological basis was identified in two tools provided by IBM: the RequisitePro suite was used as a front end for the management of the ETCS informal requirements; and, the Rational Software Architect (RSA) was used for the management of the formalization with UML and CNL of the ETCS requirements.

RSA was used for its openness in the manipulation of UML specification, and its customizability thanks to the embedded Eclipse platform it is built upon.

RSA was used as a gluing platform, and all the modules were developed as plug-ins for RSA. The main functionalities include RequisitePro custom tagging, annotation of UML diagrams with CNL (syntax checking, completion), support for the instantiation to finite domains, control of the validation procedure. Moreover, we also developed, relying on the API provided by RequisitePro and on the Eclipse platform, the traceability links among the informal requirements classified in RequisitePro and their formal counterpart in UML and CNL inside RSA.

The verification back-end is based on an extended version of the NuSMV [7] model checker, able to deal with continuous variables, and to analyze temporally complex expressions in RELTL [13, 8, 10].

5. THE METHODOLOGY IN PRACTICE

This section reports the three phases of the validation and use of the methodology. A first phase has been carried on to refine the proposed methodology concepts and process and to validate them internally to the consortium. A second phase has been the use of the methodology and the support tool by the certification body to formalize and validate a large set of requirements from the ETCS specifications. A third phase has been that of training experts from other European railway organizations and companies to the exploitation of the methodology.

5.1 Internal validation

An internal validation of the methodology has been carried out within the project.

The validation relied on the complementarity of the consortium. The choices proposed by the methodology experts have been analyzed by the railway experts. The methodology has thus been iteratively refined, polishing the concepts and process. This was achieved by formalizing and validating an increasing number of relevant requirement fragments, explicitly selected in order to assess ease of modeling and scalability of the methodology. We considered several parts of the ETCS specifications in order to exercise all the constructs defined in the methodology. We categorized and formalized a set of 90 requirements. The number of artifacts produced out of this effort has been around 150 classes and 350 different CNL constraints.

The model that has been produced and refined during the first three months of the project, allowed to refine the methodology concepts and process, while, after the third month the size of the model has been increased in order to verify the methodology and tool scalability.

5.2 Validation in an industrial setting

5.2.1 Reference ETCS subset

The methodology has been used to formalize and validate large pieces of the System Requirements Specification, subset 26 of the ETCS specifications [2]. This activity has been carried out by project partners working in notified bodies of the railway sector, experts in the domain of ETCS. The purpose was also to test the feasibility of the formalization of the whole specification. Thus, a representative set of requirements has been identified. In particular, the parts of the ETCS specifications analyzed by the study are described in the following.

- *Data exchange between On-Board Subsystem (OBSS) and Trackside Subsystem (TSS).* It has been modeled using a generic channel, with a buffer to contain the messages and a messages management strategy (e.g. FIFO), that can be specialized in each of the communication channels foreseen in the ETCS specifications (e.g. EuroBalise channel or Euroradio Channel).

- *Balise linking information.* The linking mechanism has been introduced as an interesting case study of the interaction between OBSS and TSS that can bring, when a fault occurs, to a OBSS reaction (e.g. emergency brake).

- *Movement Authority Management.* The movement authority is the authorization to move given by the TSS to the OBSS on the basis of the information exchanged.

- *RBC/RBC Handover.* The handover between two RBCs (Radio Block Centre), which means the sharing of the Movement Authority supervision and then its transfer from one controller (RBC1, which supervises the area

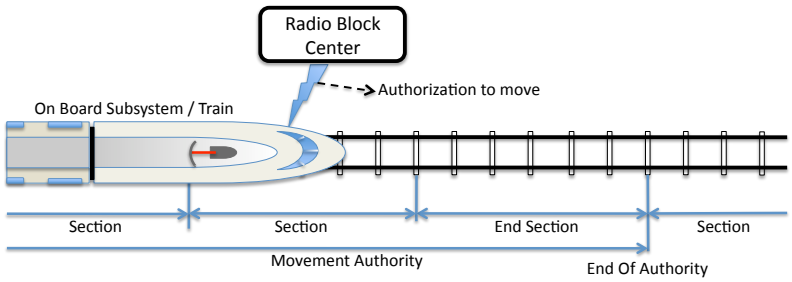

Figure 2: A scenario with some of the relevant concepts described in the ETCS specifications.

the train is leaving) to another one (RBC2, which manages an adjacent area to the one of RBC1, in which the train is entering).

A sketch of the scenario is illustrated in Figure 2.

5.2.2 Results

We have been considering 79 requirements in the whole activity. During the informal analysis phase, 83 requirement fragments have been identified, of which 34 were categorized as Glossary, 30 as Behavioral, 11 as Architecture, 5 as Scenarios and 3 were annotations. For example, the requirement 3.8.1.1 lists the elements that compose a Movement Authority (MA) starting from the End of Authority. In particular, the requirement 3.8.1.1.a says

> 3.8.1.1.a: "The End Of Authority (EOA) is the location to which the train is authorized to move."

We split it into the two requirement fragments

> 3.8.1.1.a_EoA: "The End Of Authority (EOA) is the location"
> 3.8.1.1.a_authorized: "to which the train is authorized to move."

Both have been categorized as Glossary. A Strong dependency link has been added to relate 3.8.1.1.a_authorized to 3.8.1.1.a_EoA, and another one to relate 3.8.1.1.a_EoA to the requirement fragment defining the MA.

Totally, the formalization produced 113 classes, 3 state machines, 43 CNL constraints, of which 24 of type Invariant and 19 of type Behavior. For example the requirement fragment 3.8.1.1.a_EoA has been formalized by adding the attribute "eoa" of type Real to the class "Movement_Authority" (that was already present in the model, see Figure 3).

The requirement fragment 3.8.1.1.a_authorized, instead, has been formalized with a invariant CNL constraint attached to the class "Movement_Authority" stating

> on_board_ss.location > eoa **implies not**
> on_board_ss.authorized_to_move

where "on_board_ss" is the role of an association linking the classes "Movement_Authority" and "On_Board_SS", while "location" and "authorized_to_move" are two attributes of the class "On_Board_SS", respectively of type Real and Boolean.

5.2.3 Model validation results

In the validation, we concentrated on the requirements that describe the *Movement Authority Management*, and we selected the corresponding formalized fragments. We successfully checked the consistency and different scenarios of

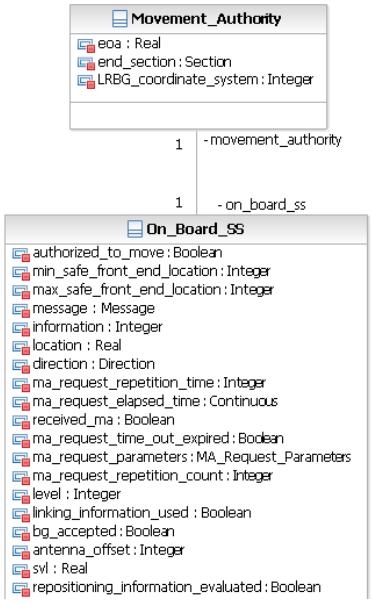

Figure 3: Excerpt from class diagram.

increasing complexity. The scenarios first forced a train to move from a starting location for 50 kilometers. Then, we forced the train to pass over a set of specified balises. As responses to these scenarios we obtained traces describing not admissible behaviors in real settings; for example, given the specifications, a balise can move along the track, or a balise can transmit also to trains that are not passing over it. In all the runs, we obtained several traces, whose length has a maximum of 18 passes, and containing a maximum of 454 variable assignments. On the bases of these output, we added several constraints to strengthen the assumptions of the scenario in order to obtain the expected trace: for example, we forced the balises not to move, we constrained the balises to send messages when the train was passing over, or the train to proceed always forward. Such additional constraints and modelling elements can be exploited by the expert during the refinement and/or modification of the ETCS requirements specifications.

5.2.4 Feedback

Based on the application of the methodology, the domain experts identified the following main advantages.

Traceability. The application of the methodology yields a high degree of traceability between the various parts of the

specification, the different parts of the formal model and verification artifacts. Thus, all the validation results can be directly related to the set of requirements under analysis.

Incremental approach. The overall ETCS model is started up from the most general definition of the System Architecture. Some components have been only modeled at their highest level of abstraction while others, more fitted to the topic the scenario referred to, have been worked out more and more deeply as long as new sentences and definitions were elaborated. This opens up the possibility to parallelize the modeling activity, and thus to enhance scalability.

Validation effectiveness the formalization and analysis of the specifications allowed for an early discovery of lacks of definitions and minor inconsistencies that could lead to misunderstanding. This information can contribute, in conjunction to the traceability features, to revisions of further versions of the specifications.

5.3 Training other industrial users

After the project, the results were presented for validation to domain experts, in the form of a workshop and training sessions, attended by national railway companies and industrial experts. In the following we briefly describe these experiences and the qualitative and quantitative feedback obtained by the experts.

5.3.1 The workshop

The first stage of the training to industrial users was a two-day workshop (`http://es.fbk.eu/events/formal-etcs/`), where ERA invited potential users coming from different railway organizations including International Union of Railways (UIC), Deutsche Bahn, Rete Ferroviaria Italiana (RFI), Gestionnaire des infrastructure Ferroviaires (RFF), Ansaldo, Alstom, CEDEX, German Aerospace Center (DLR). The discussion was open and we received a general positive feedback on the work done in the project. Particular emphasis was shown on the new aspects of the formal verification techniques that target the requirements rather than the design of the system. We also received constructive suggestions such as, e.g., to limit the formal language to a core removing the diversity of some UML constructs.

5.3.2 Training setting

The second stage were training courses organized in form of three different sessions of one week. The courses were attended by 22 railway experts. The attendees were employes of national railways-related administrations and railways industries from eight different European countries: Spain, France, Germany, Belgium, Czech Republic, Switzerland, Netherlands, and Italy. Their background was mainly related to the definition and management of railway signaling systems. Some of the experts are directly involved in the definition and maintenance of the ETCS specifications. Only few of them have been previously exposed to the use of formal methods and model checking techniques during their activities of verification of railways requirements specifications. The program of each course included two days of theoretical lectures, with exercises related to a small number of requirements extracted from the ETCS specifications, followed by three days of hands-on training exploiting the methodology with the supporting tool to formalize and validate 18 ETCS requirements extracted from one of the ETCS documents sections. The experts worked in team of two persons each in order to stimulate the internal discussions. The work on the requirements went through the whole methodology.

	Questions
Q1	*Effectiveness of the informal analysis of the specification (and supporting tool)*
Q2	*Effectiveness of the formalization phase of the specification (and supporting tool)*
Q3	*Effectiveness and the usability of formal validation phase of the previously defined model (and supporting tool)*
Q4	*Effectiveness of the overall proposed methodology*
Q5	*Effectiveness of the tool in the overall modeling process*
Q6	*Usability of the tools*
Q7	*Clarity of the output traces produced by the tool*

Table 1: Questions proposed to the training experts.

5.3.3 Training operative results

During the categorization phase the experts extracted an average of 28 requirement fragments (mainly related to Glossary and Behavioral requirements due to the nature of the selected fragments).

The formalization phase led to the specification of several class diagrams all related to the same model that on average contained 16 classes. State machines have been specified to describe the behavioral requirements. In the same phase each team specified about 12 CNL constraints to translate behavioral fragments, to impose constraints on the classes and relationships of the model, and to refine the definition of the Glossary requirements. The validation phase allowed the teams to reason about the traces produced by the tool, and to discuss several possible refinements and modifications to the piece of specifications analyzed.

5.3.4 Feedback and discussion

At the end of the course we collected the feedback, both quantitative and qualitative. A questionnaire was filled in by 18 participants anonymously.

The questions, in Table 1, focused on the methodology and the supporting tool and in particular on their *effectiveness* (questions Q1-Q5), *usability* (Q6) and *clarity* (Q7). Each question was rated in a standard scale, with 5 grades (1: "very poor"; 2: "poor"; 3: "sufficient"; 4: "good"; 5: "very good"; 0: "unsure"). In all the questions, the "good" answer is prevalent, and it is also the median value (see Figure 4). The standard deviation goes from 1.14 for Q4, for which the mean is 3.61 and the mode is 5, to 0.607 for Q2, for which the mean is 3.88 and the mode is 4.

Focusing on the analysis of critical values, some experts gave a poor judgment to the overall methodology (Q4), but a positive judgment to all three phases of the methodology (Q1-3). This incoherence could come from the different ways of perceiving and adopting the ETCS specifications and validation in the different organizations. This has been reflected in doubts related to the need of their methodological validation proposed here. Moreover, many of the experts are used to adopt model based techniques in their daily activity of validation, verification and application of systems specifications. A property based approach, like the one proposed here, determines an heavy shift of paradigm that needs more time to be completely accepted.

Comments have also been requested to the user in sup-

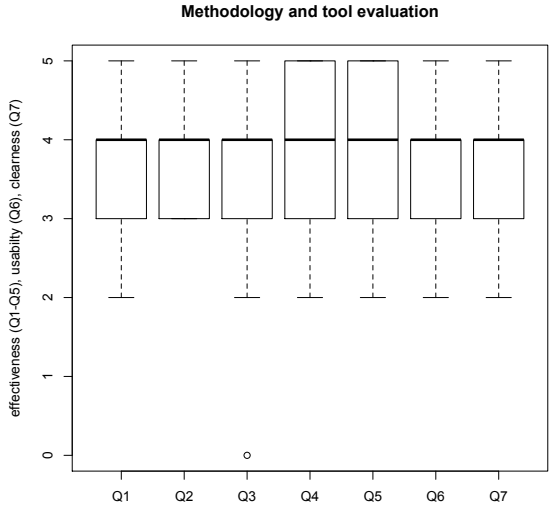

Figure 4: Boxplots of the results of the evaluation questionnaire: methodology and tool effectiveness (Q1-Q5), tool usabilty (Q6), methodology output clarity (Q7). The scale is: 1 "very poor"; 2 "poor"; 3 "sufficient"; 4 "good"; 5 "very good"; 0 "unsure".

port of the quantitative judgment; both positive and negative important issues emerged. Some experts judged the methodology to be a good candidate also in the phase of the requirements specifications, but also pointed out that applying the methodology to real-world specifications may be complex and time consuming. Finally, an issue emerged from one of the experts with respect to the criticality, in a real project, of having a design validation more than a specifications validation, as in our case. Perhaps, as claimed by another expert, both these validation phases are needed together with a clear traceability between them.

In our experience we identified some threats to consider in future structured empirical studies [28]. Threats concern the relationship between theory and observation (construct validity). *Usability*, *clarity* and *effectiveness* are subjective measures; a means to objectively weight them has been that of evaluating the number of requirements the experts effectively formalized and validated during the three days. Other threats concern the generalization of the findings (external Validity). Since the selected subjects represent a population with good knowledge on requirements specifications and strong industrial experience, and the requirements we exploited are real world artifacts, we expect similar results for other industrial experts. Only further studies with different types of domains, higher number of requirements and subjects can confirm the results. However, the small number of requirements in the training session (18), has its counterpart in the industrial experience of formalization and validation we described in Section 5.2 where a higher number of requirements have been managed.

6. LESSONS LEARNED

On the bases of the three different exploitation experiences, we discuss some interesting positive and negative emerged lessons, which should be considered in the next releases of the methodology and supporting tool and in the exploitation of the methodology in other industrial settings.

6.1 Use of the methodology

6.1.1 Advantages

The domain experts identified in the industrial exploitation, as well as in the training courses, the following main advantages: traceability from the informal requirements to the formalized requirements to the trace that exercised those requirements, possibility to apply an incremental approach to the model building, and early detection of lacks and contradictions in the requirements. Based on these observations, it seems that the property-based approach used in the methodology and in the verification engine, which encodes each requirements fragments in a distinct piece of formalism, is able to support the incremental and modular approaches to the formalization and enabled for reasoning at different level of abstraction in the specification.

From the methodology exploitation perspective, the main interesting point has been the good acceptance of the methodology by experts in signalling systems with few knowledge about formal methods. Another interesting observation raises from the different expertises in the team both during the industrial exploitation and during the training. The team members can be divided in three categories: the requirements definition experts, in particular the group of experts who defined the ETCS specifications; the analysis and validation experts, from industry and institutional organization; and the certification experts, from the certified body. All these three categories found the methodology useful from their own perspective.

6.1.2 Disadvantages

During the application of the methodology all the experts highlighted the need of more structured guidelines during all the phases. The experts identified the fragmentation and categorization of the requirements phase of the methodology as the one where the need of more structured guidelines would be more important than the other phases because of the possible implications in the effectiveness of the methodology. Some of the experts asked for a more detailed set of indications related to the use of natural language syntactic categories, such as names or verbs, as indicators of the different classification categories the requirements belongs to. In our application experience our policy has been to initially leave free the experts in choosing their own way of categorizing the requirements, and then to gather feedback from this initial work to specify precise guidelines if needed.

6.2 Proposed specification language

6.2.1 Advantages

As a general observation, the methodology addresses the key problem of the interaction between natural and formal languages. The natural language cannot be substituted in the description of the ETCS, but it is possible to structure and annotate it by means of the Rational RequisitePro tool. The formal languages are made more accessible for the domain expert with the use of UML diagrams; the restrictions imposed on the UML diagrams guarantee the formality of the approach. The experts seem to have addressed this point with few problems and in particular during the training courses they seem to have quickly acquired the expertise, having the possibility to specify also complex constraints in CNL after the two days learning phase. Another interesting

point in this line has been that the formalization task has been useful to reason on the exact semantics of each ETCS requirement since users had to pass from a rich but ambiguous language (natural language) to a (formal) limited one. This has been recognized by the experts as a result per se.

6.2.2 Disadvantages

The mixing of visual and textual paradigms for the specification of the domain concepts and of the constraints respectively, in the initial phases of the working sessions resulted in some doubts. This effect decreased in the following sessions. In any case we observed that it is useful to provide the analysts with the possibility to specifying entities in alternative ways, e.g., allowing them to specify the ontological part of the domain specification not only via class diagrams but also exploiting textual specifications in a form similar to the CNL.

As for the expressive power of the language, there may be some aspects of requirements different from the considered ones, which cannot be represented in the proposed language; for example, probabilistic aspects or deontic modalities that can be found in the case of privacy, security or legal requirements. This fact can lead to the extension of the formalism with logic construct for example from modal logics.

Anyway, although from an expressive point of view the language considered so far is quite powerful, a better level of usability could lead to extend the set of the logic constructs we exploited in the project to better formalize certain aspects such as knowledge representation.

6.3 Validation process

6.3.1 Advantages

The full automation of the validation checks has been appreciated by the people who used the tool and that had previous experience with formal methods. The generated traces have been found particularly useful to understand the informal and formal specifications. The time required by the verification engine to check the validation problems was not an issue, also because the size of the problems created during the training courses was not excessive.

6.3.2 Disadvantages

Two main aspects of the interaction with the underlying verification engine emerged to be possible issues for the usability of the tool.

A first issue is the knowledge that the user needs in order to understand the traces generated by the underlying verification engine. In the nominal case, the traces are intuitive and easy to understand, but when unexpected traces are generated it may be difficult to understand why the underlying verification engine produces them.

Second, the underlying verification engine showed to be particularly effective in picking the models that the user did not expect. This is particularly effective to disprove some properties, but it may result in a long process of refinement of the assumptions and/or of the properties to check if desired behaviors were indeed compatible with the requirements.

7. RELATED WORK

The problem of formalizing and analyzing a requirement specification is one of the main challenges in requirements engineering. Many methodologies have been proposed to solve different aspects, which are related to the management, the elicitation, the representation, the analysis, and the validation of the requirements. The works that encompass formal methods for requirement engineering can be divided into two categories: on one hand, there are many research papers which propose expressive formal languages to represent the requirements written in natural language, but they usually lack of techniques to analyze the produced formal artifacts; on the other hand, there are papers with languages, methodologies and tool to elicit, represent, and analyze the requirements also with formal techniques, but they do not focus on the formalization of requirements written in natural language, and thus they do not fill the gap between existing documents and the new produced requirements.

In the first category, we can list works such as [14] and [5], which aim at extracting automatically from a natural language description a formal model to be analyzed. However, on one hand, their target formal languages cannot express temporal constraints over object models; on the other hand, they miss a methodology for an adequate formal analysis of the requirements. Other works such as [17, 6] provided expressive formal languages to represent the requirements. Although, the proposed languages have some similarities with ours such as the adoption of first-order temporal logic, they do not allow specification of hybrid aspects which are necessary for safety-critical applications. Also these works miss a methodology for the analysis of the formal requirements and the verification algorithms are perform either with interactive theorem proving or with model checking restricted to propositional sub-cases.

In the second category, we can list the framework of Heitmeyer [19], which proposes to express the requirements in the Software Cost Reduction tabular notation. This aims at detecting specification problems such as type errors, missing cases, circular definitions and non-determinism. Although this work has many related points to our approach, the proposed language is not suitable to formalize functional requirements which describe relational constraints of the system at high level of abstraction, with temporal assumptions on the environment. Tropos [26] and KAOS [12, 27] are goal-oriented software development methodologies that provide a visual modeling language that can be used to define an informal specification. The visual modeling language is supported with annotations that characterize the valid behaviors of the model, expressed in a first-order temporal logic. Also for these specification languages, the expressive power is not sufficient because they are restricted to finite domains.

Several other formal specification languages such as Abstract State Machines [18], Z [25], B [3], Alloy [20], and OCL [23] have been proposed for formal model-based specification. Similarly, the requirement engineering phase of the DENTUM project [15] proposes to formalize the requirements with some variants of state machine. These model-based approaches are not suitable to formalize functional requirements which constrain the temporal evolution of the system (e.g., fairness) and assumptions on the physical environment. Indeed, we claim that such requirements found a more natural and traceable formalization with a temporal logic.

A different approach is proposed in [21], within the VeriSoft XT project, which validates the requirements

against the design under verification. This approach may lead to a vacuous validation of the requirements, if the design is not correct. Our techniques aim at validating the requirements before the design is created.

8. CONCLUSIONS AND FUTURE WORK

In this paper we described the EuRailCheck project, where we developed an end-to-end methodology for the analysis of requirements. The property-based approach guarantees traceability, by allowing for a direct correspondence between the components of the informal specification and their formalized counterparts. The methodology is supported by tools that integrate, within a commercial environment for traditional requirements management and model-based design, advanced techniques for formal validation. The validation techniques allow the user to check consistency, entailment of required properties, and possibility of desirable scenarios.

Within the project, we formalized a realistic subset of the specification. The results were positively evaluated by domain experts and potential end users external to the consortium.

In the future, we will pursue the following lines of activity. First, we will investigate the application of automated techniques for Natural Language Processing (e.g. automated tag extraction, discourse representation theory), in order to increase the automation of the first phase of the methodology. Second, we will explore extensions to the expressiveness of the formalism, the relative scalability issues of the verification tools, and optimization such as the ones described in [8, 10].

9. REFERENCES

[1] UML Version 2.1.2. `http://www.omg.org/spec/UML/2.1.2/`.

[2] System Requirements Specification - ETCS Subset 026 v230, 2006.

[3] J.-R. Abrial. *The B-book: assigning programs to meanings*. Cambridge University Press, 1996.

[4] A. V. Aho, R. Sethi, and J. D. Ullman. *Compilers - Principles, techniques and tools*. A.-W., 1986.

[5] V. Ambriola and V. Gervasi. On the Systematic Analysis of Natural Language Requirements with CIRCE. *Autom. Softw. Eng.*, 13(1):107–167, 2006.

[6] P. D. Bois, E. Dubois, and J.-M. Zeippen. On the Use of a Formal R. E. Language - The Generalized Railroad Crossing Problem. In *RE*, pages 128–, 1997.

[7] A. Cimatti, E. M. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: A new symbolic model checker. *STTT*, 2(4):410–425, 2000.

[8] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta. Formalizing requirements with object models and temporal constraints. *Journal of Software and Systems Modeling (SoSyM)*. DOI 10.1007/s10270-009-0130-7.

[9] A. Cimatti, M. Roveri, A. Susi, and S. Tonetta. From Informal Requirements to Property-Driven Formal Validation. In *FMICS 2008*, volume 5596 of *LNCS*. Springer, 2008.

[10] A. Cimatti, M. Roveri, and S. Tonetta. Requirements Validation for Hybrid Systems. In *CAV 2009*, volume 5643 of *LNCS*, pages 188–203. Springer, 2009.

[11] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.

[12] R. Darimont, E. Delor, P. Massonet, and A. van Lamsweerde. GRAIL/KAOS: an environment for goal-driven requirements engineering. In *ICSE'97*, pages 612–613. ACM, 1997.

[13] C. Eisner and D. Fisman. *A Practical Introduction to PSL*. Springer-Verlag, 2006.

[14] A. Fantechi, S. Gnesi, G. Ristori, M. Carenini, M. Vanocchi, and P. Moreschini. Assisting Requirement Formalization by Means of Natural Language Translation. *Formal Methods in System Design*, 4(3):243–263, 1994.

[15] M. Feilkas, A. Fleischmann, F. Hölzl, C. Pfaller, K. Scheidemann, M. Spichkova, and D. Trachtenherz. A Top-Down Methodology for the Development of Automotive Software. Technical report, Technische Universität München, 2009.

[16] N. Fuchs, U. Schwertel, and R. Schwitter. Attempto Controlled English - Not Just Another Logic Specification Language. In *LOPSTR*, number 1559 in LNCS. Springer, 1999.

[17] C. Ghezzi, D. Mandrioli, and A. Morzenti. Trio: A logic language for executable specifications of real-time systems. *Journal of Systems and Software*, 12(2):107–123, 1990.

[18] Y. Gurevich. Evolving Algebras 1993: Lipari Guide, 1995.

[19] C. Heitmeyer. Formal Methods for Specifying, Validating, and Verifying Requirements. *J. UCS*, 13(5):607–618, 2007.

[20] D. Jackson. Alloy: a lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.*, 11(2):256–290, 2002.

[21] U. Kühne, D. Große, and R. Drechsler. Property analysis and design understanding. In *DATE*, pages 1246–1249, 2009.

[22] R. Nelken and N. Francez. Automatic Translation of Natural Language System Specifications. In *CAV 1996*, volume 1102 of *LNCS*, pages 360–371, 1996.

[23] OMG. *Object Constraint Language: OMG available specification Version 2.0*, 2006.

[24] A. Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.

[25] J. M. Spivey. *The Z Notation: a reference manual, 2nd edition*. Prentice Hall, 1992.

[26] A. Susi, A. Perini, P. Giorgini, and J. Mylopoulos. The Tropos Metamodel and its Use. *Informatica*, 29(4):401–408, 2005.

[27] A. van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, 2009.

[28] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering - An Introduction*. Kluwer Academic Publishers, 2000.