Opening up the Verification and Validation of Safety-Critical Software*

Hardi Hungar and Marc Behrens
Institute of Transportation Systems
DLR, Braunschweig, Germany
{hardi.hungar,marc.behrens}@dlr.de

Abstract:

Smooth cross-border rail traffic is of important interest to commercial realizations of ETCS[1]. Starting from the hypothesis that the traditional way of developing software for safety-critical systems might be an obstacle to standardizing rail traffic, the ITEA 2 project openETCS has set out to pursue the idea of transferring an open-source development style to this domain, taking the EVC[2] as a target. The goal is to formalize the requirements in a functional model, derive, via design models, an implementation, and demonstrate how the verification and validation activities necessary for certifying a resulting product could be performed. All of this is to be done as an open-source project, employing only open-source tools. One of the main motives behind the approach is to use the potential of an open community to detect design and implementation flaws much earlier than the resource-limited inspection in a traditional development setting.

This papers discusses the challenges this new approach faces from the legal requirement of adhering to the standards, mainly the EN 50128 in this case, particularly with respect to verification and validation. This comprises the interpretation and application of the standard throughout all lifecycle phases for a open-source model-based development and qualification issues for personnel and tools.

1 Motivation

Despite ongoing efforts and investments, cross-border interoperability has not yet been achieved by the ERTMS (European Rail Traffic Management System) initiative. Two main issues are costs and diverting functional implementations. As part of the ERTMS, ETCS (European Train Control System) is the on-board system for signaling, control and train protection. Within the train, its central component is the EVC (European Vital Computer).

ETCS OBUs[3] are currently expensive if compared to existing older systems. To give specific figures, the cost for equipping 50 ICE-T in 2010 was 365 t€ per unit, with additional 100 t€ per unit for estimated software maintenance over the next 15 years [Has12].

*This research has in part been funded by the German Federal Ministry of Education and Research as part of the ITEA 2 initiative. The responsibility for the content lies with the authors.

[1]European Train Control System

[2]The European Vital Computer is the central constituent of the assembly On-Board

[3]On Board Unit, composed of the EVC and its peripherals

For full interoperability, the ETCS on-board equipment has to be backwards compatible with all ETCS track-side equipments, regardless of the manufacturer, year of build, release version, etc. This is not yet the case for all EVCs. One of the reasons is that most requirements are not formally specified and thus open to being interpreted differently by certifying authorities. To prove instances of interoperability, extensive tests are required. If they fail, it is difficult to identify the responsible component (if the cause is not that two manufacturers have interpreted (completed) the requirement specification in consistent yet different ways)[4].

The two issues cost and interoperability are accompanied by further phenomena commonly observed in complex electronic equipment in rail applications. Software generally is error-prone, thus complex software will seldom be completely free of bugs (unless development methods are drastically changed). Thus, it will need continuous service. Typical software is experienced to contain 1 to 10 bugs per KLOC (thousand lines of code)[5] High quality, mature software may have 0.5 bugs per KLOC. Achieving a rate of less than one bug per 10 KLOC will certainly be very expensive. With the ETCS kernel having an estimated 100 to 500 KLOCs one may expect a number in the order of 50 to 1000 bugs. This does not mean that the EVC will be unsafe or "wrong". In fact, most of the bugs will never affect the system at all. But it means that there is a high probability that some issues will arise sooner or later.

Given that, and taking into account the long operation periods of rail equipment, it is to be expected that the control software will need bug fixing and certainly updates for other reasons for a long period. This results in several risks, from obsolescence of development tools to vendors going out of business. In the world of proprietary, closed systems this entails high risks (i. e. high expected lifetime costs) for the railway undertaking.

The ITEA 2 project openETCS [ITE12] approaches these problems by:

Standardization reducing the diversity of equipments and promoting one common kernel for the EVC,

Formalization of requirements removing ambiguities,

Open development starting from requirements down to the software, and including all activities within the open ecosystem.

Our paper is concerned mainly with this third step: What does V&V within an open development mean, and how could it be implemented in the regulated domain of safety-critical railway equipment?

An essential ingredient of the approach is to move into making the system *open proof* [opee]: A software or a system is "open proof" if all of the following are FLOSS (Free and/or Libre and/or Open-Source Software):

1. the entire implementation (requirements, design, code, required documentation for use/maintenance, etc.)

[4]One indication of the diversity is the high amount of change requests set to be designer's choice

[5]We give commonplace figures for motivation purposes only. The topic of software and bug metrics is too complex to be discussed here.

2. automatically-verifiable proofs of at least one key criterion, and
3. all required tools needed for use and modification of the above (e. g., compilers, editors, proof-checkers, proof assistants, etc.)

Given an open proof implementation of the software kernel, it can freely be used, studied, modified and redistributed. So bugs can be identified and corrected in an early development state. This sounds very attractive, though it is by no means already an answer to the problems faced by railways and manufacturers since e. g. the specific verification obligations and the clear separation of roles required by the EN 50128 are not reflected.

[Whe06] makes a case for the superiority of FLOSS development, which he summarizes in the sentences:

> "Normal" mathematicians publish their proofs, and then depend on worldwide peer review to find the errors and weaknesses in their proofs. And for good reason; it turns out that many formally published math articles (which went through expert peer review before publication) have had flaws discovered later, and had to be corrected later or withdrawn. Only through lengthy, public worldwide review have these problems surfaced. If those who dedicate their lives to mathematics often make mistakes, it's only reasonable to suspect that software developers who hide their code and proofs from others are far more likely to get it wrong.

Again, the argument sounds good while its range is somewhat limited. For a successful application in the rail domain, the "open" approach must be made consistent with current regulations, in our case the CENELEC standards, most prominently the EN 50128 describing how software for safety-critical rail systems is to be developed.

The standards are geared towards producing a system of proven high quality, not one that is to be expected to be changed often. This contrasts with the observations made on present-day software-intensive systems, where patches and new versions are not that rare. This manifests itself in exemplifying the development in the form of a waterfall process or in form of the V model. The process has of course provisions for maintenance updates, but does not emphasize this aspect.

And it is geared towards a company or fixed organization performing the development, with requirements on the organization and personnel structure having a clear separation in roles.

2 open Approach Versus CENELEC

In this section, we discuss problems and options for reconciling an open development approach with the existing legal framework. We present the current view we have on that, it is not an agreed position of the project consortium.

2.1 Overview of the CENELEC Requirements

The main goal of the CENELEC standards EN 50126, EN 50128 and EN 50129 is to control the safety risks caused by information-processing railway applications. Risk is defined as the product of the damage resulting from some event and its occurrence probability. As it is extremely difficult to compute the risk *caused* by some equipment, the regulations address the development process to reduce the probability of unsafe or critically erroneous applications being produced. Concerning the questions related to the suitability of an open development, mainly three different categories of regulations are relevant. We concentrate here on the software regulations (EN 50128).

Management and organization The equipment is built by an entity, called the *supplier*. Personnel responsibility for different roles in the development must be assigned, and the persons must be demonstrably qualified for their respective tasks. There are restrictions on the organizational structure, e. g. which tasks must be performed by different persons or organizations. All of the development must be planned and documented in detail.

System design The design shall be documented in a top-down form, with clearly defined steps and interfaces between those steps. As a general principle, it must be verified that each step achieves its goals, and the end result must be validated against the original requirements. Verification is a general term not to be confused with proof in the mathematical meaning of the word, neither the one of formal logic with (in principle) machine-checkable results. A particular instance of the V model illustrates an acceptable lifecycle, any deviating approach shall be established as being equally suitable. Activities to be performed in the steps and documentation to be produced are defined in detail. The adopted lifecycle must make provisions for iterations, with the general idea of achieving a consistent and consistently documented development through impact analyses and change management.

Design means Tools that are used in the design are classified according to their role according to the impact they may have on the safety of the resulting software. T1 tools like requirements editors have no direct contact to the code produced. T2 tools may be involved in checking or testing software but not produce or modify code — their failure may lead to errors not being revealed but not to create errors. T3 tools finally produce or modify code or data to be used by the code. All tools used in the design must be qualified with respect to their role, T3 tools in particular have to be validated or their results extensively checked. Furthermore, the tool set used in development must be coherent. Further restrictions concern the verification activities to be performed, e. g. structure-dependent tests. Again, a consistent set of measures must be selected, where the standard defines some accepted combinations and lists a lot of (other) techniques to chose from.

2.2 Matching openETCS

One of the goals of moving to "open" is to have the development artifacts and results inspected by a large set of people, to discover errors or things which could be solved in a better way. And then, of course, have corrections or improvements performed by the "community". We discuss where this approach collides with the process prescribed (or at least recommended) by the standard.

General The EN 50128 makes provisions for changes to a software after deployment, and does so in a sensible way (impact analysis, review of only the impacted things). But it remains in the responsibility of the supplier to organize the development so that a change can be made without too much effort. This general observation becomes more virulent when revisions are envisioned to become more frequent and come from different parties as in the open approach.

Management and organization Partly implicitly, but also explicitly, the standards assume a fixed organization carrying out the development. All plans for all activities have to be fixed, and changes to a plan entail a number of regulated steps. Personnel may of course change in the course of a project, but responsibilities have always to be clearly assigned. An open community from the very idea does not satisfy these requirements, so an organizational answer to this demand has to be found.

System design openETCS goes for a model-based development of the software. Though the 2011 version of the EN 50128 includes model-based design as a highly recommended approach for SIL-4 software and lists appropriate techniques, the decision how this can be instantiated is largely left to the entity performing the development. In particular the role models may take as or in artifacts is not defined. An explorative design style, which is very common in model-based design, with parts of the system being modeled in detail while other remain only rudimentarily considered is ad-hoc not conformant. The artifacts corresponding to the phases of the proposed (illustrative) process of the standard would have to be constructed later, when all system parts have been detailed sufficiently.

Design means Model-based development comes usually with a higher degree of formalization than traditional styles. This is in accordance with the standard, which would also accept rigid (mathematical or formal logic) proofs for verification steps. The state of the art is albeit still quite far from what Wheeler advocates as "open proof". In particular, the tools themselves while complying to open proof are far from being readily usable. Among other issues, frequent updates—what is more the rule than the exception in an open community—entail many difficulties from qualification to repeatability.

2.3 Detailing the "Open" Concept

The "Open" concept implies a high degree of freedom in terms of possibilities to contribute and review given by the hosting platform Github [opeb]. Within this freedom openETCS sets out to have publicly available proofs for the source code licensed under FLOSS[6].

General Within openETCS, the source code of the EVC is categorized as "Source Code", "Qualified Source Code", "openETCS Source Code" and "Approved Source Code". Starting out with the initial openETCS partners [opec] providing qualified trusted developers, an initial contribution of "Qualified Source Code" is deployed into the openETCS repository, making it "openETCS Source Code". This deployed contribution becomes public after an incubation phase and is then available to the public and can be picked up by a development community, which itself can contribute "Source Code" to the group of qualified trusted developers. The "Source Code", in case of approval by the qualified trusted developers, becomes the "Qualified Source Code" as part of the openETCS trusted repository and after further approval by verification and validation at the end of the development cycle becomes the "Approved Source Code" and thus ready for integration by the suppliers.

Organization The "openETCS Eco System", one of the projects running inside the openETCS container [oped], describes the development and release process and gives detailed instructions on how to participate. Two roles, the *contributor* and the *committer* are defined. While the contributor can change his forked repository and apply for his changes to be inserted into the openETCS repository by a pull request, the committer directly writing to the repository is responsible to take the decision on accepting or rejecting the pull request. As a gatekeeper to the intellectual property container on the repository, the committer needs to be aware of intellectual property rights clearance.

Process Anticipating ways to profit from the potentials of an open community, the following central characteristics for the process were identified:

Modularity of Approval by enabling small and independent proofs for verification and validation, to lower the complexity and ease review.

Executability by having automatically executable tests, to speed up testing procedures and achieve high test throughput.

Formalized Impact Analysis with reporting for each occurring issue, vital for channeling and tracing the changes to the software during the full process.

Traceability up to the requirements, mandatory for SIL-4 software development and specifically important in an open project, as one needs to trace V&V verdicts to trigger decisions for changes that have a direct impact on committed software.

[6]free-libre open source software

Means Concerning V&V, the first step is to work out the verification and validation plan covering the following central topics:

Executive Summary giving an overview of the major elements from all sections.

Problem Statement describing the challenges to be answered by V&V as well as the decisions to be taken based on the V&V results as well as how to cope with potentially faulty output. It further describes the accreditation scope based on the risk assessment done on V&V-level.

V&V Requirements Traceability Matrix links every V&V artifact back to the requirements to measure e. g. test coverage and to directly link V&V results to the requirements.

Acceptability Criteria, as the direct translation of the requirements into metrics to measure success, are used e. g. for burndown charts within the process.

Assumptions that are identified during the design of the verification and validation strategy and how these assumptions have an impact on the verdict by listing capabilities and limitations.

Risks and Impacts that come across the execution of V&V tasks together with the impacts foreseen.

V&V Design states how the V&V process builds up including data preparation, execution and evaluation.

V&V Methodologies giving a step-by-step walkthrough of all possible V&V activities including the assumptions, and verdict-relevant limitations and criteria for, e. g., model verification, model-to-code verification, unit testing, integration testing and final validation (according to the standard, this involves running the software on the target hardware).

V&V Issues describing unsolved V&V issues and their impact on the affected proof or verdict.

Peer Reviews going into details on how the community can take part and how official bodies and partners are integrated into the development and review process.

Test Plan Definition going into the details of testing by describing among other things:

  Title as a unique identifier to the test plan.

  Description of the test and the test-item giving information about version and revision.

  Features to be tested and not to be tested in combination are listed together with information background.

  Entry Criteria which have to be met by the EVC before a test can be started, e. g. that the EVC has to be in level 3 limited supervision with the order to switch to level 2.

  Suspension criteria and resumption requirements are the central key to a smooth automation of the tests covering topics like *when exiting this test before step*

10, which entry criteria does it comply to or which resumption sequence has to be executed to continue testing.

  - **Walkthrough** covering a step-by-step approach of the test plan.
  - **Environmental requirements** going into the details of what is needed concerning the test environment, e. g. tools, adapter, data preparation.

- **Discrepancy Reports** identifying the defects.
- **Key Participants** describing the assignment and task for each role involved.
  - **Accreditation of Participants** is a crucial point inside the open community since the EN 50128 requires e. g. the assessor to be outside of the project or organization. Thus the role of the assessor of an open community still has to be defined since per definition everybody can participate.
  - **V&V Participants** listing the partners participating in V&V activities,
  - **Other participants** including other interest groups such as reviewer by affiliate partners[7].
- **Timeline** giving the timeline for the baselines as input to the V&V process and identifying when each artifact should be created.

To ease contribution from different parties and to reduce the testing effort, a high degree of formalization is targeted.

In realizing V&V, the selection, qualification and improvement of tools will be a central topic for research in the project. It will draw on outside results like the TOPCASED [Top] or the Project P [opea], and partners of the project will contribute some of there own tools to the openETCS pool for V&V. It will be considered to which extent the second criterion of "open proof", automatically verifiable proof, can be achieved with those means. There are mainly two ways to verify automated proving: Either by achieving trust in the prover, that is in effect verifying the tool, or by checking the evidence. The former is obviously very hard for tools which are continuously enhanced, which happens usually in open communities. And it will also be difficult for heavy weight tools like model checkers with high performance requirements which employ advanced algorithms and use sophisticated data structures.

Independently verifying the results seems more attractive. An example would be a proof engine like Coq [INR] which has a small "certification engine" to check proofs. Also approaches for model checkers [ZM03, Nam01] have been proposed, where the checkers produce evidence for verification (which might consist of huge amounts of data) on demand. An application scenario might use the checkers in the ordinary way for all usual development work, and turn on evidence generation and subsequent independent checking just for producing the safety case. Testing activities itself lend themselves much better to independent verification. For instance, one may measure test coverage with a small, validated tools. A demonstration of the viability of such approaches will have an important impact on the success of the openETCS project.

[7]affiliate partners are non-funded companies who signed the project cooperation agreement and with it get read access to the repositories starting from incubation phase to contribute e. g. by reviewing

3 Conclusion

Reconciling the idea of an open development with the strictly regulated world of safety-critical rail systems is an ambitious endeavor, and its concepts are not yet detailed. Yet in view of its potential, it is certainly worthwhile to try. Both from a conceptual as well as an economic point of view, making it real seems highly desirable. In this paper we sketched the approach taken by the openETCS project.

The next steps in contributing to its realization are detailing the verification and validation plan for the EVC software and selecting tools and methods, and also outlining what needs to be done to make them suitable for being used for developing part a system which in the end has to be certified.

Acknowledgment We acknowledge the comments of the anonymous referees which helped to improve the paper.

References

[Has12] Klaus-Rüdiger Hase. openETCS Workshop. Presentation, March 2012.

[INR] INRIA. http://inria.coq.fr/.

[ITE12] ITEA, 2012. http://www.itea2.org/project/index/view/?project=10135/.

[Nam01] K. S. Namjoshi. Certifying Model Checkers. In G. Berry, H.Comon, and A. Finkel, editors, *CAV 2001*, LNCS 2102, pages 02–13. Springer, 2001.

[opea] openDO. http://www.open-do.org/projects/p/.

[opeb] openETCS. https://github.com/openETCS.

[opec] openETCS. http://www.itea2.org/project/index/view/?project=10135/.

[oped] openETCS. http://openetcs.org/projects/.

[opee] openProofs. http://www.openproofs.org.

[Top] Topcased. http://www.topcased.org.

[Whe06] David A Wheeler. High Assurance (for Security or Safety) and Free-Libre / Open Source Software (FLOSS)... with Lots on Formal Methods / Software Verification. http://www.dwheeler.com/essays/high-assurance-floss.html, 2006. updated 2012.

[ZM03] L. Zhang and S. Malik. Validating SAT Solvers Using an Independent Resolution-Based Checker: Practical Implementations and Other Applications. In *DATE*, pages 10880–10885. IEEE, 2003.