Electronic Communications of the EASST
Volume 33 (2010)

Proceedings of the Fourth International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2010)

Security in Open Model Software with Hardware Virtualisation – The Railway Control System Perspective

Johannes Feuser and Jan Peleska

14 pages

Guest Editors: Luis S. Barbosa, Antonio Cerone, Siraj A. Shaikh
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Security in Open Model Software with Hardware Virtualisation – The Railway Control System Perspective

Johannes Feuser[1] **and Jan Peleska**[2]

[1] jfeuser@informatik.uni-bremen.de
[2] jp@informatik.uni-bremen.de
Department of Mathematics and Computer Science
University of Bremen, Germany
http://www.informatik.uni-bremen.de/agbs/index_e.html

Abstract: Using the openETCS initiative as a starting point, we describe how open software can be applied in combination with platform-specific, potentially closed-source extensions, in the development, verification, validation and certification of safety-critical railway control systems. We analyse the safety and security threats presented by this approach and discuss conventional operating system partitioning mechanisms, as well as virtualisation methods with respect to their potential to overcome these problems. Furthermore, we advocate a shift from open source to open models, in order to increase the development efficiency of combined open and proprietary solutions.

Keywords: openETCS, open source, open model, security, hardware virtualisation

1 Introduction

1.1 Background

By the end of 2009 German Railways initiated a discourse on the possible benefits of using *Free/Libre Open Source Software (FLOSS)* in railway control systems, with special focus on the *European Train Control System ETCS*. This initiative was labelled *openETCS* [Has09b, Has09a]. Reviewing evidence where security threats had been purposefully integrated into closed-source commercial software products, the author argued that open source software could be useful – perhaps even mandatory in the future – to ensure safety and security of railway control systems: even though the standards applicable for safety-critical systems software development in the railway domain [CEN01a, CEN99] require independent-party verification and validation, the complexity of the source code on the one hand and the limited budget available for V&V on the other hand can only mitigate the threat of safety and security vulnerabilities, but cannot guarantee to uncover all compromising code components inadvertently or purposefully injected into the code. As a consequence, in addition to the V&V efforts required by the standards, the broad peer-review enabled by publicly available software could really increase software safety and security[1]. German Railways indicated that also *open proofs* might be necessary to complement

[1] Following [Lev95] we agree that safety and also security are *emergent properties*, that is, they can only be attributed to complete systems, and not to software alone. When we use the terms *software safety* and *software security* in this paper, we mean *absence of software malfunctions that may lead to safety or security hazards on system level*.

the open source code, but they did not comment on the necessity to publish software models, specifications and on the potential of an open certification process.

Initially, the openETCS position statement stirred considerable interest, but has become somewhat quiet recently, at least on the public level. We suspect that this is due to the fact that railway suppliers are currently evaluating the impact of these requirements on their business models which are still based on closed software and supplier-specific solutions, in order to protect their intellectual property. In parallel German Railways will still be investigating the leverage it may already have or will gain in the future on its suppliers in order to enforce the open software idea[2]. Should German Railways – potentially supported by research communities investigating the potential of open source software in the safety-critical domain – succeed in promoting openETCS, this would automatically become an international European topic: since ETCS is a European effort to provide high-speed railway transport across borders, and since suppliers in many European countries contribute to ETCS systems and software development, success or failure of the openETCS initiative will eventually be established on European level, and not just nationally in Germany.

1.2 Objectives and Overview

This contribution is a combination of a position paper and an elaboration of solution approaches to the openETCS scenario. We argue in Section 2 that the underlying development, V&V and certification approaches enforced by the standards [CEN01a, CEN99] require that not only software, proofs (or semi-formal verification arguments) and verification tools should be published, but that the open-source paradigm should be lifted to an *open-model paradigm*, in combination with open code generators and V&V tools.

Our expectation is that – due to functional extensions, adaptations to specific hardware and national rules with impact on railway control algorithms – the open source software will nearly always have to be modified and/or enhanced by platform-specific code (Section 3). These adaptations may still be closed software or – even if made publicly available – not be of sufficient general interest to stimulate a public peer reviewing process. As a consequence we envision a scenario where future railway control systems are developed as enhancements and refinements of open models where a portion of the code has been certified according to the OpenCert paradigm and will remain unchanged in most applications, but this *re-usable core* is complemented by less trustworthy additions. Analysing the remaining safety and security threats of this scenario, we show that it can be compared to the *grey-channel paradigm* where safety-critical dependable distributed applications have to communicate over potentially unsafe channels. This situation is nowadays standard practice in distributed railway control applications and the standard [CEN01c] defines how to ensure safety and security of the resulting system, at the potential risk of reducing the availability of the system, due to fail-safe blocking of further operation.

Based on the grey-channel scenario we discuss in Section 4 how conventional operating systems mechanisms may help to reduce the safety and security risks presented by this scenario. As a final step (Section 5) we advocate the utilisation of virtualisation in order to further reduce

[2] Needless to say that, due to the possibility to re-use FLOSS, German Railways also expect a decrease of software development costs by the openETCS initiative, because suppliers would not need to re-implement major portions of the publicly available railway control algorithms.

these risks: trusted core software and target-specific adaptations run in different virtual machines, communicating according to the grey channel paradigm as if distributed over a network. We discuss the impact of this approach on the future development of virtual machines, hypervisors and communication interfaces.

Section 6 contains the conclusion.

1.3 Related Work

Our work is motivated by the challenges formulated by German Railways and the openETCS initiative [Has09b, Has09a], and uses the development and railway application scenarios presented there as a starting point. Certification issues of safety-critical systems in general are described in [Sto96]; the work presented here is specialised on the railway domain where the standards [CEN01a, CEN03, CEN99, CEN01b, CEN01c] apply. While – as described in [SC09] – quality and certification issues concerning open software in general still leave many open questions to be tackled, the railway control systems scenario described in this paper relies on certification according to the rules defined in the standards listed above. The only differences to today's standard procedure are that (1) the certified code and its associated documentation are made publicly available and (2) it may be necessary to re-certify the software as soon as adaptations and extensions have been made for a concrete system implementation.

The model-driven approach advocated in this paper is based on domain-specific modelling as decribed in [KT08] because it is well-known that the utilisation of domain-specific description formalisms and associated automated code generation and mechanised model-based testing and verification has high potential in the railway domain [HP03, HPK09, Mew09]. It has to be emphasised, however, that the open-model approach and the security analyses presented in this paper only rely on the availability of an arbitrary specification formalism that is suitable for formal verification and automated code generation. Even conventional UML2 [OMG03a, OMG03b] (and potential augmentations by means of the profile mechanism) are suitable if a well-defined model-to-text (i. e. code) transformation is used to associate a transformational semantics with the semi-formal UML model [BBHP06].

2 From Open Source to Open Model Software

The terms open source software (OSS) and free/libre open source software (FLOSS) refer to source code. Certifiable train control systems software, on the other hand, has to be complemented by a collection of additional artifacts contributing to the *safety case*, that is, the comprehensive and structured evidence justifying that the resulting system will guarantee safe operation. Among others (for details see [CEN01a, CEN99]), the list of these artifacts comprises software specification and design models and complete records of all V&V measures taken to ensure software code compliance with its applicable specifications, as well as evidence showing how all functional and structural aspects of the software have been throughly tested and verified[3]. It is well known that for systems of highest assurance level[4] the effort for elaboration of the safety

[3] The term *verification* comprises formal mathematical analyses, as well as semi-formal reviews and inspections.
[4] The so-called *system integrity level SIL-4* and the associated *software safety assurance level SSAS-4*.

case is frequently higher than the proper software development effort. As a consequence, just source code without the additional artifacts mentioned above would be nearly worthless. Additionally, as soon as FLOSS code has to be adapted, this will become quite hard and often invalidate previous V&V results if these adaptations have not been guided by a systematic approach, preferably based on a software model giving indications how to modify the software in an admissible way.

Due to these considerations we are convinced that OSS/FLOSS can only be applied successfully in the railway control systems domain if code is accompanied by or – even better – embedded in free/libre open *models*. Following the principles of object-oriented modelling, the description formalism should be based on a meta-meta model that is publicly available as in the case of the OMG meta object facility [SVE07] or in the case of the *Graph, Object, Property, Role and Relationship (GOPRR)* meta-meta model introduced in [KT08] for the design of domain-specific languages, so that the model could be unambiguously interpreted and processed by various development and V&V tools. Additionally, these models should clearly indicate where platform-specific or application-specific changes are admissible by means of class inheritance, overriding and overloading of operations or by means of adding components with admissible interfaces.

As sketched in Figure 1 we suggest the terms *open meta metamodel*, *open metamodel*, and *open model* for the higher-level abstractions required "above" the open software. Figure 1.

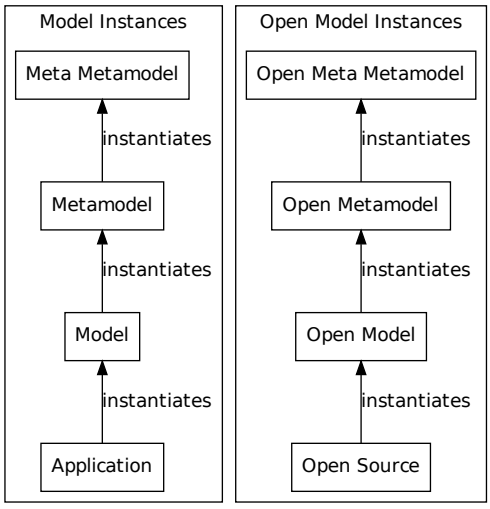

Figure 1: Denomination for open domain-specific modelling (DSM).

A typical benefit from this approach would be that certification credit for module verification could be re-used for all software methods or functions that have not been changed for the platform-specific adaptation. On the other hand, these adaptations may require extensive new V&V activities and associated re-certification for the complete system, if their impact on the re-usable components is not clearly visible. This problem will be analysed more closely in the

sections below.

Re-usable certifiable open model software also requires a specification of the admissible tool chain to be used for model-to-text transformations, compilation and linking and V&V regression activities, because otherwise it could not be guaranteed that the software build process would be performed correctly and the V&V process would lead to trustworthy results. These tool aspects, however, are beyond the scope of this paper.

3 Security Analysis for The Open Model Software Scenario

Figure 2 shows a very general scenario how a platform-specific adaption of an open model and associated FLOSS could compromise the resulting system. This example has one model implementation which is directly generated from the open model, and therefore gets certification credit by means of re-use for all component-specific V&V artifacts. Suppose that sub-models 2 and 3 had to be newly developed for the platform-specific solution, resulting in supplier implementations 1 and 2. It is obvious that component-specific V&V measures have to be performed for these new implementations. We are interested in the question whether some certification credit could be re-used for model implementation 1 on software integration level, for example, the V&V measures previously taken to show that this implementation cooperates correctly with other components directly generated from the open model.

Unfortunately, this is not true without further restrictions: if implementation 2 is malicious it may compromise both model implementation 1 and supplier implementation 1, either by sending corrupted data through their designated interfaces or through covert channels which were not intended to be utilised according to the model[5], or by means of unintended resource usage creating denial of service attacks.

As a consequence no certification credit can be re-used for model implementation 1 on software integration level: All corresponding V&V artifacts have to be re-produced in order to justify that none of the platform-specific implementations can compromise the resulting system through any of the other implementations. In the two following sections we will analyse suitable measures to counter the threat presented by such malicious implementations.

4 Partitioning

As seen in the previous section, the creation of faulty or malicious supplier implementations in an open model scenario cannot be completely avoided, but their impact on other software components should be minimised. Modern operating systems offer a number of standard mechanisms to cope with these situations. All of these mechanisms may be summarised under the keyword *partitioning*, which has to be enforced in the resource domain and in the time domain.

In the resource domain partitioning means that faulty or malicious components cannot interfere with the (legal) access of another software component to the resource and cannot access any resource without proper authorisation. Typical resources in the embedded systems domain

[5] E. g., by writing to illegal memory addresses if all implementations run as operations or threads in the same address space.

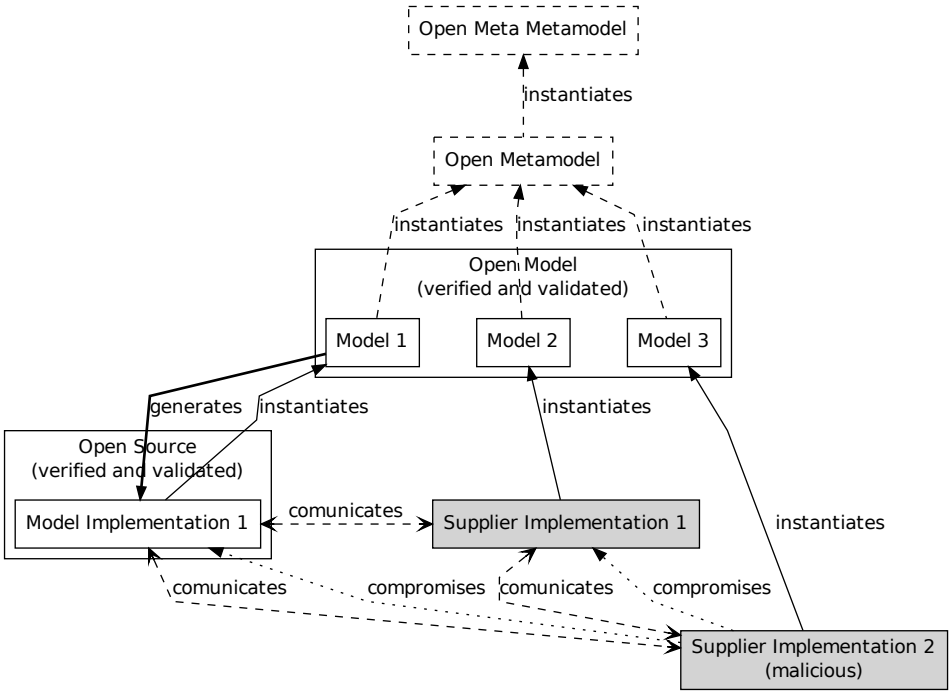

Figure 2: Possible security threats in open model software combined with platform-specific adaptations.

are CPU cores, memory, hardware and software interfaces and operating system resources like semaphores, message queues and others. The traditional way of implementing resource partitioning is through different privilege levels for application and operating system layer, virtual address spaces supported by memory management units, encapsulation of resource access by means of system calls and kernel access mechanisms and access control mechanisms enforced by the operating system [Sta08a, Sta08b]. Currently, resource partitioning is typically static for safety-critical embedded systems, since the dynamic allocation and de-allocation during system operation is hard to verify, or – as in the case of dynamic memory partitioning with paging – unsuitable for the embedded domain as long as suitable solid-state disks are not available.

In the time domain partitioning implies that corrupt components may not access any resource – in particular, the CPU and the communication interfaces – for an undue amount of time, thereby creating denial of service attacks. Time partitioning is typically enforced by means of schedulers; prominent examples are partition (process) schedulers complying with the ARINC specification 653P1-2 [ARI05] defining a distributed operating system as used in modern avionics (e. g. Airbus A380)[6]. On the interface level, the Avionics Full-Duplex Switched Ethernet Network guarantees fixed communication bandwidths for different communication links by means of an on-board scheduler for package transmission [ARI09] (also used in the Airbus A380 and in modern Boeing aircrafts). Alternatively, the Time Triggered Protocol (TTP) [TTP10] assigns temporal communication slots to processes.

5 Hardware Virtualisation with Open Models

Para Virtualisation. The conventional mechanisms enforcing partitioning described in the previous section have the draw back that they require all software components to run under the regime of a single operating system. At least in the current situation, where several on-board train controllers are required in order to cope with national boundary conditions, this is disadvantageous for openETCS, because of the diversity of supplier hardware and associated operating systems. This problem also has implications on the open model approach: if the code generated from these models relies on the availability of specific operating system mechanisms (for example, a certain scheduling policy), this code may only run on platforms whose operating systems support these mechanisms. This impairs the potential re-use advantages of the open model approach in a considerable way.

As a solution to this problem we suggest *hardware virtualisation*, where – controlled by a *hypervisor* and a host operating system – several *guest* operating systems may run simultaneously in so-called *virtual machines (VM)* on the same hardware [vmw07]. A hypervisor works as a *virtual machine monitor (VMM)* which either dispatches sensitive instructions issued by a guest operating system that require kernel privileges to the hardware or emulates these instructions by means of interaction with the host operating system. In the latter case the hypervisor may have the capabilities of a micro kernel in its own right and may even render an additional host operating system superfluous. This is the case when so-called *para virtualisation* is applied:

[6] Standard [ARI05] only requires to assign guaranteed time slices to partitions in round-robin manner. This does not guarantee that applications will meet their deadlines. In [MHG+09], a more sophisticated approach based on *earliest deadline first scheduling* is described

Here the sensitive actions of guest operating systems are not dealt with on machine instruction level, but instead the guest utilises a pre-defined hypervisor API providing hardware access on a higher level of abstraction, thereby considerably improving the performance of applications running in virtual machines (see [Tan08, pp. 568] for a more detailed overview).

The most important micro kernel capabilities that we suggest for hypervisors supporting para virtualisation are

- a preemptive round robin scheduler enforcing fixed execution time windows for each virtual machine, similar to the inter-partition scheduling requirements of [ARI05],
- driver management for hardware interface access with explicit assignment of interface visibility to selected virtual machines,
- control of the memory management unit to enforce memory partitioning and assign either fixed memory portions to virtual machines or limit each VM's amount of dynamically allocated memory,
- communication mechanisms supporting message-based inter-VM and remote communication.

Open Model Scenario With Virtualisation. In this virtualisation scenario, the code portions generated directly from the model without platform-specific adaptations would run in one virtual machine, and platform-specific adaptations would run in separate virtual machines. Since each virtual machine mimics a complete computer with its local operating system, platform hardware and peripherals, resource partitioning is easily enforced: hardware interfaces that should not be accessed by a group of software components are simply not visible in their "virtual computer hardware". The utilisation of main memory could be limited by the hypervisor, and the separation of memory address spaces is already enforced on virtual machine level. Communication between virtual machines can be performed, for example, by means of a socket interface.

The effect of virtualisation is similar to several distributed application programs cooperating by means of remote communication. The impact of a malicious or otherwise faulty component is reduced to corrupt communication behaviour on the intended interfaces: it is impossible to influence the outside world by other interfaces but the ones configured for the virtual machine. Since from the viewpoint of the receiver it cannot be distinguished whether the sender or the communication channel is corrupt, this situation is already well understood in today's distributed railway control applications communicating over public networks known as *grey channels*: the safety-relevant components have to be developed on the basis that any type of error may occur on the grey channel, because this is a communication medium whose hardware and software has not been developed with the same assurance level as the safety-critical application itself. As a consequence, the safety-relevant component has to cope with repetition, deletion, insertion, resequence, corruption and delay of messages and guarantee fail-safe behaviour in presence of these faults. The defence mechanisms against these types of faults or attacks have to comply with the standard [CEN01c].

Applying the concept of hardware virtualisation to the initial open model scenario in Figure 2 leads to the revised scenario depicted in Figure 3. It also contains the generated model implementation and the two supplier implementations. In contrast to Figure 2 all supplier implementations

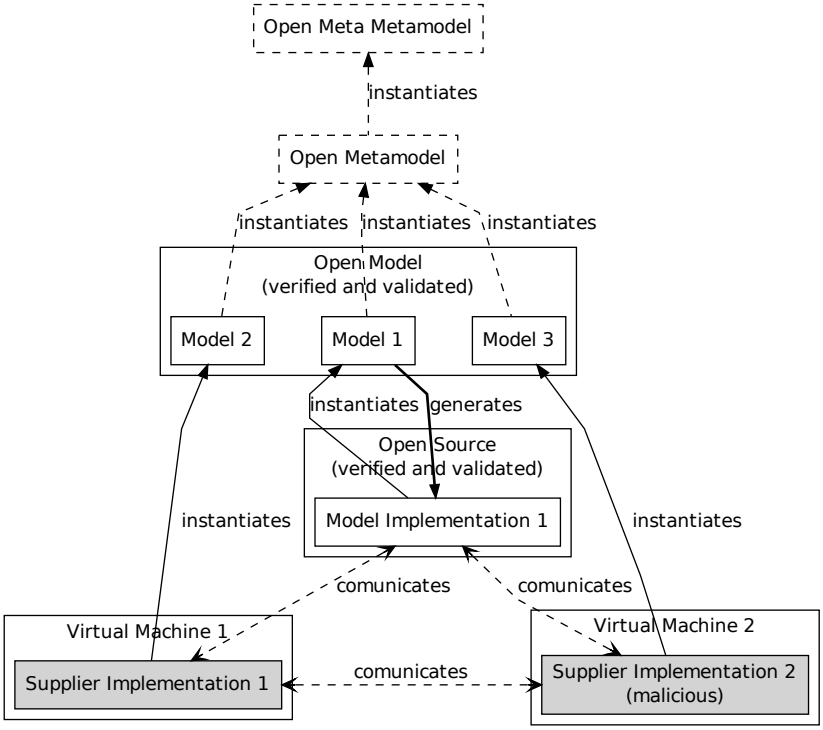

Figure 3: Hardware virtualisation for open models

are now locked in their own virtual machines. This ensures that the malicious implementation cannot compromise any other part of the software through covert channels or abuse of resources, while a communication of legal or corrupted data over the intended channels is possible.

The prevention of undue bandwidth consumption on hardware interfaces can be handled by means of scheduled I/O as described in Section 4. As a consequence, the need of certified high-integrity hypervisors or host operating systems arises. The effort to develop, verify and certify these is justified as soon as the hardware platform can be re-used in different application scenarios, so that hypervisor or host operating system would be re-used as well.

Certification Issues. We advocate the following development, validation and certification approach in the open-model scenario with virtualisation as described above:

- The hardware platforms for railway control systems should be equipped with a hypervisor possessing the micro kernel qualities listed above.
- This hypervisor should be open source and fully certified according to the aforementioned standards and according to the highest assurance level SSAS-4, because all further assurance considerations depend on the trustworthiness of this component.
- The re-usable core of the open-model software should be developed and fully validated with respect to one suitable operating system. In particular, the safe behaviour in presence of corrupt interface data received over a grey channel can be checked once and for all.
- Platform-specific or other functional adaptations should only be admissible as model derivations that may run in separate virtual machines which do not host the re-usable core software[7].
- The adaptations are again validated according to the applicable railway standards, running in an operating system possibly differing from the one hosting the re-usable core.
- Both the re-usable core and the adaptations use a remote communication paradigm to communicate with each other and integrate the required protection mechanisms for grey channel communication.
- The admissible operating systems for re-usable core and adaptations have to comply with the hypervisor API according to the para virtualisation paradigm.
- For an integrated HW/SW system consisting of several virtual machines with guest operating systems hosting the re-usable core and one or more adaptations, certification credit for the local validation activities[8] already performed can be granted.
- For certification of the integrated HW/SW system it remains to validate the following structural, functional and non-functional system properties:

[7] So, for example, simple overloading of some operations in a class belonging to the re-usable core would not be allowed.

[8] Such as module tests and SW integration tests, or partitioning properties for different processes that belong to the same adaptation, and will therefore run in the same VM on the target system.

- Correct communication among virtual machines and between VMs and interfaces.
- Correct functional behaviour of the integrated system: To this end, only functional requirements involving two or more virtual machines have to be tested.
- Performance and robustness in avalanche (stress) situations.

Proof of Concept. We intend to substantiate the advantages of open model openETCS advocated so far by means of a case study. For this purpose the ETCS would be particularly well-suited because the existing open standard [ETC07] may serve as an informal specification, to be formalised as a model conforming to our domain-specific meta model which is currently under development (see Section 2). Typically, the model holds objects directly corresponding to hardware elements like sensors or actuators, e.g. a reader device for Balises [ETC06]. Such elements are often subject to supplier-specific implementations.

To compare conventional operating system methods like process scheduling and memory management with the usage of hardware virtualisation, effects on the rest of the software have to be measured for both cases, in presence of one or more malicious supplier implementations. Therefore we will purposefully generate "supplier" implementations showing the relevant types of malicious behaviours, based on a formal threat model. Examples for these threats are:

- denial of Service attacks on
  - CPU bandwidth,
  - network interface bandwidth,
  - software interfaces of other objects,
- injection of false data to software interfaces of other objects,
- infinite blocking of calls by other objects.

The results of these tests with and without hardware virtualisation could be directly compared and would lead to a conclusion about the efficiency of the virtualisation approach.

It is obvious that hardware virtualisation cannot prevent all of the above mentioned attacks from affecting other software components. Therefore the fault tolerant behaviour of software implementations is highly relevant. A possible solution would be the utilisation of a standardised interface library, e.g. CORBA [HV99], providing methods to handle time-outs and other related problems. CORBA is not needed to be included in the metamodel, but only in the code generator [KT08]. Therefore, this approach would not add additional complexity to the metamodel and its model instances.

The distribution of the software as open models is another aspect of the concepts proposed here. To attract a community of substantial size and adequate competence it is crucial to provide a comprehensive tool chain with the open models. Obviously, the editors and compilers sufficient for open source distribution have now to be complemented with meta-modelling tools, modelling tools and code generators under open source licenses. Moreover, the tool set should be extended by a simulation and visualisation platform so that different solutions could be tried out without the availability of real-world railway infrastructure.

6 Conclusion

We have described an approach for combining open source software and proprietary system-specific code for the development of certifiable railway control systems. Following the certification requirements of applicable standards in the railway control systems domain, this approach requires not only code, but also models and V&V artefacts to be made publicly available. For ensuring the safety of a mixed open/closed source system, we have analysed the support mechanisms offered by today's operating systems in order to prevent software components of minor trustworthiness to corrupt the behaviour of the trusted safety-critical core. In particular, we advocate virtualisation mechanisms to encapsulate components of different assurance levels for achieving fault containment. Virtual machines running components of different assurance levels may communicate according to the grey channel paradigm which is already well understood in today's distributed railway control applications.

Our contribution was intended as a position statement and an indication of promising solutions. The justification of these claims is currently elaborated by means of case studies based on the ETCS specification [ETC07, ETC06] and the *Positive Train Control (PTC)* system concept [PTC10]. To this end, a domain-specific description formalism specialised on the railway control system domain and following the concepts explained in [HP03, HPK09, Mew09] will be used.

The technical effort for substantiating a proof of concept should be accompanied by an open discussion about how to attract an open source community of sufficient size to the openETCS idea: only if the number of actively contributing members is big enough, the desired effect of quality improvement by peer-review, test or analysis can be expected. We believe that such numbers can be reached because – due to the complexity of control objectives on the one-hand and to their illustrative meaning on the other hand – this application domain has always attracted practitioners and researchers in the safety-critical systems and formal methods domains.

Though this paper focused on the railway domain, we expect that the approach described here will be valuable for other safety-relevant domains as well, in particular for avionic systems. Our work has been influenced by the experience of the second author with validation of safety-critical railway control and avionic systems.

Acknowledgements. The first author has been supported by Siemens AG through a research grant of the Graduate School on Embedded Systems GESy at the University of Bremen (`http://www.informatik.uni-bremen.de/gesy`).

References

[ARI05] *Avionics Application Software Interface, Part 1, Required Services*. AERONAUTICAL RADIO, INC., 2551 Riva Road, Annapolis, Maryland 21401-7435, 12 2005.

[ARI09] *Aircraft Data Network, Part 7, Avionics Full-Duplex Switched Ethernet Network*. AERONAUTICAL RADIO, INC., 2551 Riva Road, Annapolis, Maryland 21401-7435, 09 2009.

[BBHP06] K. Berkenkötter, S. Bisanz, U. Hannemann, J. Peleska. The HybridUML Profile for UML 2.0. *International Journal on Software Tools for Technology Transfer (STTT)* 8(2):167–176, January 2006. Special Section on Specification and Validation of Models of Real Time and Embedded Systems with UML.

[CEN99] CENELEC. *EN 50126 - Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)*. CENELEC European Committee for Electrotechnical Standardization, Central Secretariat: rue de Stassart 35, B - 1050 Brussels, 09 1999.

[CEN01a] CENELEC. *EN 50128 - Railway applications - Communications, signalling and processing systems - Software for railway control and protection systems*. CENELEC European Committee for Electrotechnical Standardization, Central Secretariat: rue de Stassart 35, B - 1050 Brussels, 03 2001.

[CEN01b] CENELEC. EN 50159-1. Railway applications -Communication, signalling and processing systems Part 1: Safety-related communication in closed transmission systems. 2001.

[CEN01c] CENELEC. EN 50159-2. Railway applications -Communication, signalling and processing systems Part 2: Safety related communication in open transmission systems. 2001.

[CEN03] CENELEC. *EN 50129 - Railway applications - Communication, signalling and processing systems - Safety related electronic systems for signalling*. CENELEC European Committee for Electrotechnical Standardization, Central Secretariat: rue de Stassart 35, B - 1050 Brussels, 02 2003.

[ETC06] ERTMS/ETCS - Class 1 System Requirements Specification. 24-02 2006. Issue 2.3.0.

[ETC07] ERTMS/ETCS Functional Requirements Specification FRS. 21-07 2007. Version 5.0.

[Has09a] K. R. Hase. openETCS - Ein Vorschlag zur Kostensenkung und Beschleunigung der ETCS-Migration. *SIGNAL +DRAHT* 10, 10 2009.

[Has09b] K. R. Hase. openETCS - Open Source Software für ETCS-Fahrzeugausrüstung. *SIGNAL +DRAHT* 12, 12 2009.

[HP03] A. E. Haxthausen, J. Peleska. Generation of Executable Railway Control Components from Domain-Specific Descriptions. In *Proceedings of the Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'2003), Budapest/Hungary*. Pp. 83–90. L'Harmattan Hongrie, May 15-16 2003.

[HPK09] A. E. Haxthausen, J. Peleska, S. Kinder. A formal approach for the construction and verification of railway control systems. *Formal Aspects of Computing* 17, December 2009. DOI: 10.1007/s00165-009-0143-6.

[HV99] M. Henning, S. Vinoski. *Advanced CORBA Programming with C++*. Addision-Wesley Publishing Company, 1999.

[KT08] S. Kelly, J.-P. Tolvanen. *Domain-Specific Modeling*. JOHN WILEY & SONS, INC., 2008.

[Lev95] N. G. Leveson. *Safeware*. Addison-Wesley, 1995.

[Mew09] K. Mewes. *Domain-specific modelling of railway control systems with integrated verification and validation*. PhD thesis, University of Bremen, 2009.

[MHG+09] A. Mancina, J. Herder, B. Gras, A. Tanenbaum, G. Lipari. Enhancing a Dependable Multiserver Operating System with Temporal Protection via Resource Reservation. *Real-Time Systems* 43:177–210, 2009.

[OMG03a] OMG. UML 2.0 Infrastructure Specification, OMG Adopted Specification. http://www.omg.org/cgi-bin/apps/doc?ptc/03-09-15.pdf, September 2003.

[OMG03b] OMG. UML 2.0 Superstructure Specification, OMG Adopted Specification. http://www.omg.org/cgi-bin/apps/doc?ptc/03-08-02.pdf, August 2003.

[PTC10] Positive Train Control - Wikipedia. URL, http://en.wikipedia.org/wiki/Positive_Train_Control, 2010.

[SC09] S. A. Shaikh, A. Cerone. Towards a metric for Open Source Software Quality. In *Proceedings of the Third International Workshop on Foundations and Techniques for Open Source Software Certification (OpenCert 2009)*. Volume 20. 2009.

[Sta08a] W. Stallings. *Operating systems: internals and design principles*. In [Sta08b], chapter 7 - 8, pp. 353 – 453, 2008.

[Sta08b] W. Stallings. *Operating systems: internals and design principles*. Prentice Hall, 2008.

[Sto96] N. Storey. *Safety critical computer systems*. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA, 1996.

[SVE07] T. Stahl, M. Völter, S. Efftinge (eds.). *Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management*. Volume 2, chapter 5, pp. 64–71. dpunkt-Verl., 2007.

[Tan08] A. S. Tanenbaum. *Modern Operating Systems*. Pearson, 2008.

[TTP10] Real-Time Systems Research Group: The TTP Protocols. URL, http://www.vmars.tuwien.ac.at/projects/ttp/ttpmain.html, 06 2010.

[vmw07] vmware. Understanding Full Virtualization, Paravirtualization, and Hardware Assist. 08 2007. white paper.