Virtualised Embedded Systems for Testing and Development (VIRTUES)

Författare: Karl Meinke (KTH Stockholm) , Andreas Rasmusson (Scania)
Datum: 2017-09-28
Projekt inom FFI delprogrammet Trafiksäkerhet och automatiserade fordon

FFI Fordonsstrategisk Forskning och Innovation

Innehållsförteckning

1 Sammanfattning

I fordonsbranchen används idag i stor omfattning s.k hardware-in-the-loop (HIL) testning för att på ett tillförlitligt sätt testa distribuerade elektronik-system, häri även innefattande de inbyggda systemens mjukvara. Dock är HIL-riggar en knapp och underhållskrävande resurs. Både mjukvaru-utveckling och mjukvaru-testning kan förenklas och effektiviseras genom att arbeta med en virtualiserad HIL-rigg (VHIL). Detta öppnar upp möjligheter att köra massivt parallella tester på data-servrar i molnet eller på arbetsstationer som utnyttjar multi-core teknologi.

Projektet VIRTUES, ett samarbete mellan Scania CV och KTH över 3.5 år, har utrett frågeställningar kring virtualiserad testning av embedded-mjukvara som kör på virtuell hårdvara. Inom projektet har det utvecklats mjukvaruutvecklingsverktyg för att använda maskin-inlärningsstödd testfallsgenererering och även en virtuell-hårdvaruplattform. Dessa verktyg har kopplats samman och då använts i fall-studier rörande testning utifrån formella krav samt i felinjiceringstester. Mjukvaran som testats är faktisk automotive-mjukvara från ECU:er.

Vi uppskattar att minst 90% av den mjukvarutestning som idag utförs på HIL-riggar kan, med lika god tillförlitlighet, överföras till en VHIL-rigg, och därigenom spara kostnader samt frigöra HIL-riggarna till mer renodlade utforskande elektriska experiment.

2 Executive summary in English

In the automotive industry, realistic tests of distributed electronic systems, including embedded software, are today executed using *hardware-in-the-loop (HIL) testing*. However, the necessary HIL rigs are a limited resource. Both testing and software development could be simplified by working with a virtualised HIL rig (VHIL). This could give testers the possibility to execute massively parallel test suites, using less expensive multicore technology.

The VIRTUES project has been conducted over 3.5 years between Scania CV and KTH to investigate this hypothesis. The project has developed new software engineering tools that support machine-learning assisted test case generation, as well as virtualised hardware emulation. These tools have been integrated together. The resulting platform has been succesfully applied to requirements and fault injection testing of automotive ECU applications.

We estimate that 90% of the activities carried out on a HIL-rig today could be transferred to this VHIL rig, if it were to be productized.

3 Bakgrund

In the automotive industry, realistic tests of distributed electronic systems, including embedded software, are today executed using *hardware-in-the-loop* (aka. HIL rig) testing. However, the necessary HIL rigs are a limited resource. Both testing and software development could be simplified by working with a virtualized emulation environment. This could give testers the possibility to execute massively parallel test suites, using less expensive multicore technology.

Large-scale parallel testing requires that the Swedish automotive industry supplement today's manually designed test suites with larger automatically generated test suites.

4 Syfte, forskningsfrågor och metod

Since 2009, KTH-CSC has carried out research into automated requirements-driven test case generation. Previous KTH-CSC collaboration with the automotive industry (Volvo) supports the hypothesis that our test technology would be highly suitable to automate large scale parallel testing within a virtualized environment.

New research has been necessary to arrive at four goals: (1) a multi-ECU simulation environment, (2) a virtualised HIL-rig, (3) an integrated automated parallel testing toolset, and (4) case studies of the costs/benefits of using the integrated virtualisation and testing platform for developing new and existing systems. The end-users will be embedded software developers and software testers in general.

5 Mål

In our original proposal, the project activities were divided into <u>3 main tracks:</u>

1. the ECU Simulation Platform,
2. the Virtualised HIL rig,
3. the Testing Platform.

Below we describe the main goals within each track.

Project Track 1: The ECU-simulation Platform

Project Track 1 considers the implementation and evaluation of the instruction set simulation platform itself. The simulation platform is required input to the HIL-virtualisation and testing tracks. To reduce technical risk in the project, this track will deliver increasingly capable simulators that the other tracks can use from early on.

AP 1.1 Development of new instruction set single-ECU simulation platform

In order to go beyond today's focus on application level functional requirements, it is necessary to be able to more closely simulate the effects of executing on a particular microcontroller. We will investigate how to exploit state-of-the-art instruction-set-simulation technology and if it is feasible to run a completely unmodified binary version of the real production software from a real ECU.

We will survey the current state-of-the-art of ready-to-use instruction-set-simulation technologies with sufficient support for the hardware we intend to virtualise, and select one platform for further development of hardware models. We will investigate strategies for how to incrementally develop the required custom hardware models.

AP 1.2 Development of new multi-ECU simulation platform

To test multi-ECU scenarios, a simulation comprised of several, simultaneously executing ECUs will be setup. Since the simulation is more complex, consideration must be given to if and how the simulation can be partitioned over several cores or several computers.

We will identify some real-life scenarios where a function on the truck is realised by several cooperating ECUs and will set up a virtual platform where these ECUs are executing as a single simulation.

We intend to create both a functional level multi-ECU simulation as well as an instruction-set-simulation based variant of it.

The latter variant was not completed since we deemed that it would not enable enough interesting experiments for the effort needed. In the multi-ECU experiments, below, we have used instead, either the single-ECU instruction set simulator or the functional-level multi-ECU simulator.

Project Track 2: Using the Virtualised HIL-rig

Using the virtualized hardware developed in Track 1 we will modify the test-framework used to configure the HIL-rigs and run HIL-tests to be able to use virtual hardware instead of HIL-rigs. This opens up the possibility for test-developers to have a "HIL-rig" on their desktop for parallel test case development and also for parallel regression testing of existing test cases.

AP 2.1 Virtualisation of HIL-rig test frameworks with complex environment models and auxiliary hardware

For a fully virtualized solution, we need to carefully investigate which parts of the HIL-rig software that are affected and where to modify them. Many of the issues/design consequences discovered here will be relevant for the general task of virtualising hardware rigs and/or when designing new hardware rigs with later virtualization in mind.

AP 2.2. Extending a Virtualised HIL-rig with Capabilities beyond the HIL-rig

Compared to functional-level simulations, the instruction-set simulator developed in Track 1 lets us control and observe more aspects of the execution, e.g. fault-injection, resource usage and (to some extent) execution time. Now, certain non-functional behaviours that could previously only be observed in the rig will be reproducible. Even some behaviours that are not testable in the hardware rig will be possible to verify using the simulator.

We will provide new ways for the test-framework to control the simulation and we will investigate how our testing standards and practices can be improved by using these new methods.

Project Track 3: The Testing Platform

AP 3.1. Requirements Testing of ECUs and Systems

As a part of automated requirements testing, verdict construction necessitates that user requirements must be precisely modeled using a requirements modeling language. KTH will collaborate with Scania engineers to understand how, when and why to make precise user requirements models.

AP 3.2 Integration of Test Automation with Test and Emulation Platforms

This work-package deals with technical integration of the KTH test toolset into the Virtual HIL-rig and the new simulation environment, exploiting new observability options.

AP 3.3. Case Studies in Unit, Integration and System Testing

Two central questions of research track 2 are: (i) how to apply automated requirements testing within Scania, and (ii) what are the costs and benefits of this? We feel these questions are best answered empirically by performing testing studies on many significant components and subsystems. We need to understand how cost/benefit varies between different products.

AP 3.4 Test Tool Optimisation

It will be important to understand the current limitations of the KTH test toolset and how these can be overcome. Machine learning and model checking (the core technologies in the KTH test toolset) are rapidly evolving technologies that we need to exploit. Optimisation of the tools will be focused on meeting the performance needs indicated by automotive case studies (AP 3.3).

AP 3.5. Parallel Testing on Multicore Platforms

One of the main motivations for moving to emulator technology for the automotive industry is the possibility to create multiple virtual test platforms using inexpensive multicore computing hardware. This opens up the possibility to perform large-scale testing in parallel. We anticipate up to 200 processors in our own research. Multiprocessor technology seems also necessary in the future to perform multi-vehicle simulations needed for testing systems-of-systems, e.g. platooning functions.

6 Resultat och måluppfyllelse

Project Track 1: The ECU-simulation Platform

AP 1.1 Development of new instruction set single-ECU simulation platform

We surveyed the state-of-the-art in instruction set simulation suitable for our hardware and use-case. As a result we chose to base the implementation of our platform on the open source project QEmu to maximize our ability to experiment with parallel and distributed execution and avoid licensing and code-availability bottlenecks.

QEmu has one CPU-model that is sufficiently similar to our hardware, but no models of the peripheral hardware.There exist a handful commercial products who provide models of not just the CPU, but the entire microcontroller-circuit (MCU). However, for our proprietary ECU, a custom model of its additional peripheral hardware, will eventually have to be created anyway.

Since the ECU-software we intended to run was already developed and its assumptions about the execution environment are fixed, we could settle for implementing models of only the hardware actually used by the ECU-software.

In addition to implementation of hardware models we had to make a few modifications to the base platform QEmu. We corrected a few bugs that we reported back to the Open source project. We also added support for a new input format that we could base on work done elsewhere in the QEmu open source community.

Throughout the implementation we could use the simulator's built in debugger support to selectively bypass parts of the application whose hardware interaction we had not yet modelled. This was quite helpful since we could then implement the hardware model in increments.

Incrementally more capable versions of the platform were provided for use in the case studies in AP3.3. Feedback of the needs of AP3.3 guided how the priorities were set for the incremental deliveries.

We used the debugger as the interface for interacting with the ECU-software in a way that did not require any inspection support in the ECU-software itself. This is crucial in order to provide a virtualization platform that does not interfere with the ability to re-architect the ECU-software.

The integration of the platform with LBTest in AP3.2 is based on the scripting via the debugger-interface. This method gives the integrator great power when, in the wrapper code, the meaning of the formal requirements is to be translated into stimuli to and response from the platform. Advanced debugger features, such as conditional breakpoints and breakpoints that trigger side-effects can be used.

We estimate that 90% of the activities carried out on a HIL-rig today could be transferred to this VHIL rig, if it was productized. In a real world software development use-case, where the ECU-software evolves over time, it is necessary to modify the simulator platform and base it on a commercial product with a model of the entire microcontroller. Arguably, it would make most sense if the circuit-manufacturer provides or recommends a model of the circuits it manufactures.

1.2 Development of new multi-ECU simulation platform

The ECU:s on the truck are relatively loosely coupled in that they (mostly) communicate via message passing on a CAN-bus. This is an opportunity for parallelizing the multi-ECU simulation, at the expense of the risk of introducing non-determinism and imperfect reproducibility. In our use-case we did not need to consider more tightly coupled simulations of e.g multi-core cpu:s or parallel simulation of tightly coupled dynamical systems.

We have implemented a synchronous multi-ECU simulation that is used internally as Scania and it is sufficient to reproduce many behaviours we need to test. It is not as fast as a parallel version could be, in theory.

A Master's Thesis was started in order to investigate techniques for parallelization of simulations, e.g. DynamicTimeWarping algorithms. The this work was not finished, and did only produce tentative results.

We have cooperated with the project "Virtual Truck & Bus"(Energimyndigheten, Dnr: 2014-007465) at Scania. We can use their multi-ECU simulation platform and we have implemented components for that platform. It makes sense to cooperate on their platform for more coarse grained simulation of large scale system behaviours.

This platform was experimented on in AP3.3 has not yet resulted in any publishable results, in part due to resource-availability, and side-channel issues as discussed in AP2.1 below.

Simulation of the detailed behaviour at instruction-set level is crucial to reproduce behaviours from scenarios concerning e.g. multi-core or hardware fault-injection, and is where we subsequently focused our efforts in AP3.3. Hence, in this AP, we did not pursue the implementation of a multi-ECU version based on co-simulated instruction-set simulators.

Project Track 2: Using the Virtualised HIL-rig

AP 2.1 Virtualisation of HIL-rig test frameworks with complex environment models and auxiliary hardware

In order to build a complete virtual replica of a HIL-rig, we need to virtualise also the environment-components of the HIL-rig. Here we investigated issues we ran into when we attempted to virtualise a HIL-rig in use at Scania.

Depending on use-case, the ECU's external environment in a HIL-rig can be quite complex, not only its behaviour, but also in the mechanisms and shortcuts used to build the HIL-rig. We selected a use-case where we avoided tightly coupled dynamical models (i.e. control-loops), but still there are a few interesting observations we can make.

In a HIL-rig, it is in some cases convenient to simply use a hardware component instead of a model its behaviour. For example, the ECU might interact cryptographically with 3rd party hardware. Also, one may want to encourage ad-hoc testing by providing a real steering wheel etc.

In a HIL-rig the emphasis is mainly on being able to produce as many behaviours as possible. This is in contrast to being able to exactly reproduce each of the (possibly fewer) behaviours in a setup with models instead of hardware as the environment. Thus, being able to observe something e.g. "occasionally, on the average once a day" is counted as a "win" for the HIL-rig testers, although maybe not so much so for the engineers subsequently trying to pin-point the bug.

If we require complete reproducibility there needs to be a single clock responsible for stepping the system. In our HIL-rig some of the configuration tools interact with the HIL-rig on a side-channel parallel to the simulation. All such side channels need to be engineered away.

The tools interacting with a HIL-rig might also use components that happen to limit how many simulations that can be run in parallel. Some tools may have licensing restrictions or they may use limited resources in the operating system kernel, e.g. resource limited device drivers.

As a concrete proof-of-concept, we set up a simulation, where the environment-model software from a HIL-rig was connected to the virtual ECU to produce a richer execution environment for the virtual ECU. Actually solving most of the technical hurdles observed above will have to be done in subsequent projects.

AP 2.2. Extending a Virtualised HIL-rig with Capabilities beyond the HIL-rig

In an instruction set simulation we have many new ways to monitor and interact with the software. We can chose to interact with the simulation on two levels; the guest-level which is the execution of the code on the simulated CPU, and on the host-level, which is the level were the hardware-models are implemented and which runs on the workstation's CPU.

Instead of designing a fixed API for interacting with the simulation, we instead interact programatically via a debugger-interface at either guest- or host-level (or both!). This gives us essentially total control and observability of the simulation, and also ensures that the guest program in no way has to provide architectural support for testing or debugging.

Compared to the functional specification expressed in source code, we take as input the true binary realisation of the ECU-software. For this reason we can faithfully reproduce behaviours that depend on memory-layout, e.g. bit-flip effects, stack-overflows, wild-pointers, interrupts. This faithfulness was the basis for a fault-injection use-case AP3.3. However, keep in mind that the binary realisation will lack representation of design invariants from the functional level, e.g. data types. So both levels have distinct useful aspects.

On the other end, compared to real hardware we are able to monitor things that the real hardware cannot inform us about, e.g. that the software configures the hardware in a correct manner according to some external safety guidelines.

We are also not subject to resource limitations in the hardware's debugging support, e.g. how many simultaneous break-points that can be active or data-transfer-rates between target-hardware and the debugger.

There are commercial ISS-platforms, (not QEmu) where the simulation can be run backwards. A fault can then be located, quite conveniently, by simply starting out from its observed failure and then execute backwards.

Some feedback from the wrapper-program development of AP3.3 was that, naturally, the more we instrument the simulation, the more the simulation will be slowed down. This is not a big problem in our main use-case where tests against the simulations is done as batch-jobs on server farms, but it will affect if and how the simulation is useful for ad-hoc tests, for test-development and for interaction with physical hardware.

Having a systematic way to switch between different simulation granularities at different times should be required in a platform, if it is to be used also interactively by humans. QEmu has no such infrastructure in place.

Project Track 3: The Testing Platform

AP 3.1. Requirements Testing of ECUs and Systems

A thorough literature survey of formal modeling languages for safety requirements modeling of real-time embedded automotive applications was performed very early on in the project. In particular, previous research results from Scania and Robert Bosch Gmbh in this field were useful. Potential relevance to emerging ISO 26262 issues, such as traceability of safety cases, were also considered.

This literature survey arrived at a number of pattern languages and visual requirements modeling languages (state transition diagrams). The advantages and disadvantages of these from a testing perspective have been described in the conference publication [2]. This requirements modeling research has continued within our new KTH research projects such as EU ECSEL project SafeCOP and KTH project STaRT.

AP 3.2 Integration of Test Automation with Test and Emulation Platforms

LBTest was integrated with the Scania test execution platform WinComP very early in the project. This tool integration then supported many early case studies in unit testing (see 3.3 below). LBTest was also successfully integrated with QEMU and GDB by a PhD student, and could be used to perform case studies in virtualized hardware fault injection. The combined virtualized fault injection test platform is described in the technical report [10] that will be submitted for conference publication. A pre-study of communication fault injection for distributed software architectures using LBTest is described in the conference publication [5].

AP 3.3. Case Studies in Unit, Integration and System Testing

There was a strong emphasis in the project on unit testing. However, this was supplemented with external case studies (from current and previous KTH research projects) on integration and system testing, even up to the level of cyber-physical systems-of-systems (vehicle platoons). For unit testing, three case studies were considered: an engine start application (ESTA), a dual circuit steering application (DCS), and a fuel level display application (FLD). All three were actual production components, and collectively were representative of many testing situations and issues at Scania. Especially FLD had a strong pedagogical aspect, since the case study was previously shared among other Scania projects.

2 Masters and 1 PhD student carried out these case studies, supervised by KTH and Scania staff. The KPIs focused on to measure the performance of LBTest were mainly: (1) the difficulty of identifying and modeling product safety requirements, (2) the difficulty of performing full coverage testing with LBTest, (3) the success rate in identifying bugs and (4) a comparison of the effectiveness of LBTest with an in-house product testing environment. The successful performance of LBTest under all these headings was reported in a conference publication [2] and an extended journal publication is currently under revision [9].

For integration and system testing a brake-by-wire application and a platooning simulator were considered. LBTest proved itself to be scalable to much larger problems when tool optimization and improvements had been completed (see 3.4 and 3.5 below). The results of this work appear in the conference publication [1].

AP 3.4 Test Tool Optimisation

The test tool LBTest has been optimized in a variety of ways to improve both performance and usability. For better usability, a command line version of the tool was produced for seamless integration with QEMU and GDB. To improve performance, we have introduced learning algorithms for non-deterministic automata that give better model compression and higher test coverage. We have introduced learning algorithms that can exploit powerful multi-core architectures (see AP 3.5 below). We have considered a tightly integrated explicit-state model checker to give faster more scalable performance on complex testing tasks, and extended bug discovery. Arising from observed future needs, to consider environmental and road dynamic modeling, we have recently initiated a new KTH project STaRT (joint with KTH-CSC Robotics department) to study model checking for spatio-temporal logics. Some of these improvements have been described with benchmarking results in publication [6].

AP 3.5. Parallel Testing on Multicore Platforms

A parallelized machine-learning algorithm has been developed which is able to exploit the power of a multicore test execution platform. This was benchmarked on a high-latency multi-ECU testing problem (a platooning simulator). On a 4-core machine, the algorithm gave about 3-times speed-up compared with a sequential learning algorithm. Total processor usage level could exceed 98%, which showed that effective use of multicore processing power was possible, and that our solution could usefully scale to larger core numbers. The results have been published at EPEW-2017 [1].

7 Spridning och publicering

7.1 Kunskaps- och resultatspridning

Hur har/planeras projektresultatet att användas och spridas?	Markera med X	Kommentar
Öka kunskapen inom området	X	(1) KTH has organised Schloss Dagstuhl Workshop 16172, *Machine Learning for Dynamic Software Analysis: Potentials and Limits*, with Open University (Bennaceur), TU Darmstadt (Hähnle) and NASA Ames (Giannakopoulou). This leads to the first published textbook on this subject. (2) KTH has organised the special track *Machine-Learning in Software Products and Learning-Based Analysis of Software Systems* with Uni Dortmund (Howar) and Clausthal University of Technology (Rausch) at ISoLA 2016 conference (3) KTH has a STINT joint Japan – Sweden collaboration proposal in *Software Construction with Machine Learning* (pending) (4) KTH has presented VIRTUES results at international conferences and workshops including EPEW, IMBSA, ISoLA, SEFM. (5) KTH and Scania have presented results at national conferences such as: EiF 2016, Swedsoft STEW 2017, KTH-ICES annual conference 2016, Vehicle ICT Arena Innovation Bazaar 2017, (6) KTH has presented VIRTUES results at 2 Scania Research Open Days, and 1 Scania-KTH ICES workshop (ReVamp) (7) PhD student H. Khosrowjerdi is preparing his Licentiate thesis at KTH.
Föras vidare till andra avancerade tekniska utvecklingsprojekt	X	KTH continues VIRTUES research ideas in 2 projects (1) KTH-ICT project *Spatio-Temporal Planning at Run Time* (STaRT) (2) EU ECSEL project *Safe Co-operating Cyber Physical Systems using Wireless Communication* (SafeCOP). KTH and Scania plan for an FFI continuation project.
Föras vidare till produktutvecklingsprojekt		
Introduceras på marknaden		KTH (Meinke) is in discussions with KTH Innovation regarding possible commercialisation of the LBTest tool
Användas i utredningar/regelverk/ tillståndsärenden/ politiska beslut		

7.2 Publikationer

1. K. Meinke: *Learning-Based Testing of Cyber-Physical Systems-of-Systems: A Platooning Study*, Proc. EPEW 2017, pp135-151, LNCS 10497, Springer Verlag, 2017.
2. H. Khosrowjerdi, K. Meinke, A Rasmusson: *Learning-Based Testing for Safety Critical Automotive Applications*, Proc. IMBSA 2017, pp 197-211, LNCS 10437, Springer Verlag, 2017.
3. Falk Howar, Karl Meinke, Andreas Rausch: *Learning Systems: Machine-Learning in Software Products and Learning-Based Analysis of Software Systems*, Special Track at ISoLA 2016. ISoLA (2) 2016: pp 651-654, LNCS 9953, Springer Verlag, 2016.
4. Amel Bennaceur, Dimitra Giannakopoulou, Reiner Hähnle, Karl Meinke: *Machine Learning for Dynamic Software Analysis: Potentials and Limits*, (Dagstuhl Seminar 16172). Dagstuhl Reports 6(4): pp 161-173 (2016)
5. K. Meinke, P. Nycander: *Learning-Based Testing of Distributed Microservice Architectures: Correctness and Fault Injection*. Proc. SEFM Workshops 2015: pp 3-10, LNCS 9509, Springer Verlag, 2015.

Unpublished material in preparation
6. K. Meinke, *Recent Progress in Learning-based Testing*, to appear in: A. Bennaceur, R. Hähnle, K. Meinke (eds), *Machine Learning for Dynamic Software Analysis: Potentials and Limits*, Springer Verlag.
7. A. Bennaceur, K. Meinke, *Introduction to Machine Learning for Software Analysis*, to appear in: A. Bennaceur, R. Hähnle, K. Meinke (eds), *Machine Learning for Dynamic Software Analysis: Potentials and Limits*, Springer Verlag.
8. Proceedings of Dagstuhl Workshop 16172 *Machine Learning for Dynamic Software Analysis: Potentials and Limits*, A. Bennaceur, R. Hähnle, K. Meinke (editors) Springer Verlag, to appear
9. *Automated Behavioral Requirements Testing for Automotive ECU Applications*" submitted to the Journal of Software Testing, Verification and Reliability.
10. H. Khosrowjerdi, K. Meinke, A Rasmusson, *Virtualized-Fault Injection with Learning-based Requirements Testing*, in preparation.

.

8 Slutsatser och fortsatt forskning

The VIRTUES project has delivered validated tools and use-cases that convincingly demonstrate the advantages of automated requirements testing and fault injection testing in an automotive context.

The advantages of the VIRTUES approach have been demonstrated, such as lowered cost of equipment, flexibility and agility of approach, reliable and rigorous test results, and higher test coverage. Furthermore, we have shown that our techniques scale up to significant problem sizes, although further work is possible to extend scalability even further.

The advantages of our approach all contribute towards reduced development costs, higher product reliability, and shorter product development times. In the context of emerging autonomous driving software, we feel these advantages will be attractive for commercial exploitation. The more software-related behaviour we can simulate virtually, the more will the use of HIL-rig testing change from detecting faults in the ECU-software into simply validating that the hardware-models faithfully describe the actual hardware. In the limit (ideally) the reliance on HIL-rigs for software development will be greatly decreased.

With all new possibilities to observe and interact with the system under test, the more important will it be to produce test-cases intelligently, in order not to create a huge, unmaintainable test-suite. Test generation from more abstract descriptions, as exemplified by LBTest, seems to be a quite promising way forward here. For example, the quite high-level requirement from AP3.3 that "*a bit-flip in a check-sum protected memory area shall area lead to a system restart*" can be implemented quite concisely and still produce surprising behaviours, induced by the concrete memory layout.

The results of the VIRTUES project are already being exploited within the EU projects SafeCOP (ECSEL) and TESTOMAT (ITEA), and we anticipate that a continuation project between KTH and Scania will emerge in the future. By building on the execution platform produced in this project, there are many interesting next steps that we might explore. It may also be possible to combine this future work with other related Scania research projects such as ReVamp and Virtual Truck and Bus.

9 Deltagande parter och kontaktpersoner

Professor Karl Meinke (BSc, PhD) , School of Computer Science and Communications (CSC) KTH Stockholm, karlm@kth.se.

Andreas Rasmusson (MSc, Senior Engineer), Scania CV AB, andreas.rasmusson@scania.com