Impact of multicore platforms in hardware and software certification

Summary results of RECOMP WP4

Risto Nevalainen
Spinet Oy
Espoo, Finland
risto.nevalainen@spinet.fi

Oscar Slotosch
Validas AG
Munich, Germany
slotosch@validas.de

Dragos Truscan
Åbo Akademi University
Turku, Finland
dragos.truscan@abo.fi

Uwe Kremer
TÜV SÜD Rail GmbH
Munich, Germany
uwe.kremer@tuev-sued.de

Vicky Wong
Space Systems Finland Oy
Espoo, Finland
vicky.wong@ssf.fi

Abstract— **This article is a summary of the main results[1] of the RECOMP project in certification and qualification topics. Work package 4 in RECOMP has made contributions in amendments in safety standards, including multicore specific hardware and software requirements and addressed the need for additional methods and techniques. Cost-effective certification has also been a main topic in WP4 work.**

Current safety standards provide a well defined reference in most generic and domain-specific safety issues. However, they are not complete and accurate enough in certifying multicore platforms and applications. By definition, standards are somewhat behind of leading edge already at the moment when they are published. Multicore specific methods, techniques and tools are mainly additional details as certification criteria, missing in current safety standards.

Keywords—multicore, certification, safety standards

I. INTRODUCTION

Certification and qualification of safety-critical systems ensure that the systems are suitable for their intended use. These activities involve the provision of objective evidences to demonstrate with the level of confidence required by an applicable standard that the systems will behave as intended. The greater the potential consequences in the event of failure of the systems, the higher the required level of confidence becomes, which generally result in higher costs. Work Package 4 of the project RECOMP (Reduced Certification Costs Using Multicore Platforms) addresses certification and qualification aspects of safety-critical systems and software. Based on a subset of safety requirements from RECOMP WP1, WP4 has defined different approaches with the goal of reducing certification and re-certification costs of safety-critical systems, as well as enabling the qualification and certification of multicore systems. This paper enumerates the WP4 findings that target the automotive and industrial automation domain, which are described in more details in the two project deliverables: *Standards Amendment targeting multiple processor safety systems (SMP, AMP)* [1] and *Development process amendments to achieve a robust and safe development* [2]. The former is a set of results and findings about the need to include multicore specific requirements in IEC61508 and other safety standards, while the latter describes needs in development process itself and how methods and tools can support robust and cost-effective system development and certification.

[1] The methods and tools discussed in this article have been evaluated, improved, or developed by different RECOMP partners for addressing multicore issues, and not by the authors of this article, whose mere contribution was to briefly enumerate them.

II. TOWARDS NEW CERTIFICATION APPROACHES

A. Integrated use of safety standards in qualification and certification

Safety is presented in many ways in current standards. In some standards safety is only implicit, without being mentioned. It can be even avoided, so that the standard does not get any flavor from the safety domain. A good example is assurance case standard ISO/IEC15026, which can be directly applied for safety case. The safety community has much to learn from a large set of "non-safety" standards; how to use them and how to write reusable standards.

In some other standards, safety is the main focus area, and any other topic may be avoided (e.g. security). Good example is ISO26262 [5]. It is quite directly derived from the main functional safety standard IEC61508 [3], adding a lot of automotive specific details and requirements. Similar standards are developed for avionics, nuclear power, medical electronics and various other domains.

Safety can be evaluated using several basic approaches. The key output, the system and/or software product, can be assessed against a predefined set of safety requirements. Safety assessment approach covers both the product and the

process characteristics and properties. Process assessment focuses typically on the product development phase. All these approaches produce valuable information in building trust on the safety of the product. So far, harmonization of different approaches is missing in certification and qualification of safety-critical hardware and software.

RECOMP project has developed an integrated method, in which product evaluation, safety assessment and process assessment can be combined. This method is based on systematic collection and classification of evidences, so that they are collected only once but can be reused for different qualification and certification approaches. It saves costs in the typically laborious data collection phase. The main skeleton for integrated method is ISO/IEC15504 standard for process assessment [4]. It has also extension for safety management, safety engineering and tool qualification processes. Methods are an essential part of evidences in a typical certification case.

B. Pre-certified and reusable elements in systems

Safety functions are normally realised by subsystems. Typical safety function consists of the three subsystems sensor, logic unit and actor (see Fig. 1).

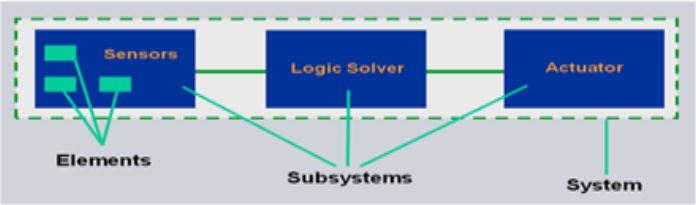

Figure 1. Fig. 1 Parts of a typical safety function

These subsystems consist of elements. An element is defined according to ISO26262-1:2011 as a system or part of a system including components, hardware, software, hardware parts and software units. The IEC61508-4:2010 defines an element as part of a subsystem comprising a single component or any group of components that performs one or more element safety function. The definitions are oriented on the behaviour of a functional unit without the definition of physical limitations. Therefore it is possible to define generic elements with generic safety functions participating on an overall application specific safety function.

Typical generic elements with a generic safety function are for example a microcontroller, an operating system, software libraries or power supply units. Generic elements should be qualified and certified for the use in safety relevant applications without knowledge of the final safety function. The terms compliant item according to IEC61508-4:2010 or safety element out of context according to ISO26262:2011 can and should be used.

In general, the development and design of generic safety elements has to follow the requirements of safety standards for avoidance and control of systematic faults as well as the robustness for random faults. There are some effects to be considered when safety elements are combined. In principal it is possible to reach a higher risk reduction by combining already pre-certified elements in parallel for a redundant operation. But this is only valid for the influence of random faults. For instance, if two homogenous elements are used in parallel a systematic fault will affect both elements at the same time. Therefore it is necessary to combine elements with a sufficient independence to be also robust against systematic faults.

The main benefit of using pre-certified safety elements is the simple reuse of these elements in different safety applications without a re-assessment of the basic element functionality. Furthermore, higher risk reduction levels could be reached by combining these elements. The main requirements for a pre-certification are to tailor the development process depending on the safety element and to provide the necessary information for the system integrator in the safety manual.

III. MULTICORE IN EMERGING SAFETY STANDARDS

The IEC61508 standard describes various safety requirements as well as fault avoidance and fault control measures that could be used during a safety-related system development process and in the system design to reduce the possibilities of random hardware failures and systematic failures that would compromise the functional safety of the system. However, the IEC61508 requirements are stated in a rather generic manner and do not sufficiently address multicore system development.

Isolation, separation and independence of safety and non-safety-related functions as promoted by the IEC61508 standard are areas in which the use of multicore technology in safety-related applications is experiencing the most difficulties. Multicore platforms provide the possibility to realise mixed-criticality applications on multicore platforms. However, the contention of shared resources inherent to multicore system may lead to concurrency and synchronization issues such as latency, jitter, throughput, lockout and deadlock [6], which could significantly undermine the determinism or integrity of the system design. Performance and control stability may also be compromised due to problems associated with cache coherency, cache sharing and shared data buses [7], [8]. Undermined determinism and performance in turn make it more difficult to achieve and demonstrate temporal and spatial separation between components in a multicore system. In other words, it becomes difficult to achieve and demonstrate that a higher criticality application is free from interference by a lower criticality application that resides on the same computational platform despite the execution of the two SW being handled by different cores. There could still be other shared resources such as memory, data buses or peripherals which would result in adverse interference such as the blocking of higher criticality task by lower criticality task that in turn could prevent the high criticality task from meeting its schedule and satisfying its performance requirements.

Enabling technology for using multicore platforms for mixed-criticality applications involves partitioning mechanism, which generally introduces additional abstraction layers (e.g. separation microkernel and

hypervisor) into the system. This tends to violate the simplicity in the implementation of safety functions as favored by the safety standard. Interactions between the different layers and components become more complex and more difficult to be visualized without proper tool support. Shared resource contention problems could manifest in different system levels and could also be solved on the different levels. However, these solutions could pose additional constraints on other design levels and must be taken into consideration during system design. One example is the stripping down of services, e.g. interrupts, provided by operating systems due to data consistency or scheduling scheme executed by the separation microkernel.

As a result, developers of components in a multicore system cannot work in isolation. For instance, a common memory model and a standard communication protocol would have to be established and adopted by all the parties responsible in the system development. However, this could be difficult at times as the development timelines of the system components are not always aligned, particularly when a safety-compliant subsystem is procured. The subsystem might contain complex mechanisms, which might have impact on the functionality of the system but are for various reasons not documented. Such omission might be more detrimental in multicore systems than in single core ones as the information could be fundamental for properly handling task parallelism. Inaccessibility to such technical information could also hinder the design of proper diagnostics measures as well as test planning.

In essence, multicore system development is not much different from system development using a network of single-core processors but issues are exacerbated due to shared resource problem and the complex interactions between elements. It becomes more difficult to demonstrate sufficient independence and deterministic performance in the design and through verification. Techniques and methods currently in use are geared more towards uniprocessor systems and might not be applicable for multicore systems. Current worst case execution time (WCET) analysis methods, for instance, are in some cases incapable in providing a reasonable bound on the execution time as they do not account for multicore specific issues such as cache interference and overhead induced by cache flush operations during context switching.

Due to the intricate interactions between elements, it becomes more difficult to demonstrate freedom of interference in a multicore system, i.e. ensuring that low criticality applications do not interfere with high criticality applications, in terms of peripheral usage, input/output interfaces, communication and memory, through functional testing. Functional testing becomes unmanageable and at times insufficient. Other verification methods such as model-based or statistical analysis, formal methods and qualified synthesis methods would have to be used in conjunction to support functional testing.

IV. SELECTED METHODS TO REDUCE CERTIFICATION COSTS

Different safety standards such as IEC61508 and ISO26262 regulate the design of mixed-criticality systems using the concept of independence. It requires the functions in a system to be developed according to the highest relevant SIL among all these functions, unless there is sufficient independence of implementation between them.

The independence between functions of different safety-criticality levels is to be proven by either demonstrating both spatial and temporal independence or justifying that any violation of independence is controlled (IEC61508 - 7.4.2.9). Furthermore, safety-standards require the justification and documentation of independence, which implies the need for corresponding verification methods.

RECOMP partners have suggested different methods and tools to reduce certification costs and risks in achieving adequate safety for mixed-criticality, multicore systems. These contributions have been described in more details in [2]. The contributions of the RECOMP project can be roughly split into five different categories, although the border between these categories is often rather thin:

- demonstrating spatial and temporal independence
- validation & verification methods
- monitoring approaches
- specification methods
- tool chain

The following sections briefly summarize contributions by RECOMP partners in each category.

A. Demonstrating spatial and temporal independence

1) Spatial independence

Suggestions concerning spatial independence include traditional measures such as hardware memory protection or operating system support for virtual memory [2].

In [15] an architecture is proposed that involves a centralized memory protection unit (MPU) in addition to CPU local protection units to guard the use of shared memory. This module receives all requests, compares the address ranges, the source of the access and the access type with its configuration and accepts or rejects the access. This simplifies the certification of multicore systems running multiple safety-critical applications or even mixed-critical workloads. Once it has been shown that the separation of the applications works and fulfills all requirements, each application can be evaluated individually.

2) Temporal independence

IEC61508 recommends two main methods for achieving temporal independence of two software functions: scheduling techniques and timing analysis.

Scheduling techniques try to solve the problem either statically or dynamically. The former manages resource access in a static way, such that temporal interference between functions can be excluded by design, for example by defining exclusive (i.e. non-overlapping) time slots for

each function. The drawback of these approaches is that they may cause higher base latencies, lower processor utilization, as well as limited flexibility in case of unexpected events, such as sporadic inputs or random errors [2]. Dynamic scheduling, in turn, allows for calculating the priorities at execution time, providing lower base latencies and a greater flexibility in case of unexpected events.

In this context, support for modular certification was provided via the AUTOFOCUS3 framework [13], which was used for modeling and analyzing the behaviour and structure of multicore systems. The framework allows for task and message scheduling synthesis for shared memory multicore architectures using satisfiability solving techniques.

Timing analysis techniques justify the extent of temporal interference using formal analysis. These formal analysis techniques use an underlying mathematical model which can be used to infer and formally prove different properties of the system. For instance, one can determine the worst case for the system timing and to check if given constraints like task or process latencies or data path latencies are met. If all constraints are met in the worst case scenario, they will also be met in other cases. This kind of techniques imply additional effort in performing scheduling analysis in order to prove that each function satisfies its own requirements independently from the others.

Several tools for formal timing analysis have been used (e.g., TimesTool [18], Cheddar [19]) or extended (e.g., SymTA/S [20]) in the context of RECOMP. Although timing analysis tools such as TimesTool and Cheddar target single core implementations, multicore implementations can be modeled, designed and realised by these same instruments. For instance, one can start the design with building over the conception of real-time components, then proceed to modeling and timing analysis, and finalize with the implementation. Different algorithms can be used in the multicore-context to provide an off-line partitioning scheme, after which timing analysis can be performed as in single-core case. The SymTA/S tool has been extended to enable the design, optimization and verification of multicore scheduling, by introducing new communication overhead analyses and heuristics for schedule generation w.r.t safety and efficiency requirements.

A crucial issue in the context of real-time performance is the occurrence of random hardware errors, which can cause time-consuming re-execution of parts of the software according to a non-deterministic pattern. This makes predicting or bounding without uncertainty the execution time difficult. In the context of RECOMP, new methods have been introduced for estimating the real-time capabilities of individual functions if random hardware errors occur.

Different safety analysis techniques have been also employed as a means to reduce the effort of certifying multicore platforms. Such techniques include analysis of dependent failures, safety analysis at system level via inductive and deductive analysis and calculation of safety metrics. For instance, applying some of these analysis techniques at system level, instead of going to subsystem level, reduces the effort and cost of using multicore platforms.

B. Validation and Verification approaches

Several approaches for validation and verification of multicore-based mixed criticality systems have been studied in RECOMP.

At hardware level, techniques for verification of digital circuits such as simulation, verification via automated test generation and execution (using for instance, SystemVerilog), or assertion based specification (temporal logic statements inserted into hardware specifications) have been suggested [15].

At software level, automatic white-box testing using concolic execution has been proposed [16]. This technique generates a test suite automatically and the coverage of the test suite itself can be verified directly without being dependent on the correctness of the test generation algorithm. Thus, any problems during test generation are detectable by noticing that the generated test suite does not fully achieve the required coverage of the source code being tested.

C. Diagnostic and monitoring techniques

Different diagnosis methods (e.g. power-on self tests, run-time tests, etc.) and monitoring techniques including traditional and more recent (model-based diagnosis) have been suggested in the RECOMP project, especially in the context of the automotive domain [2].

In the context of multicore, monitoring and diagnostic techniques have been suggested both at hardware and software level. Among those suggested at hardware level we enumerate: lock step processing (2 cores run same code and evaluate the result), external voltage and quartz monitoring, monitoring system I/O or shared memory protection unit [2]. Other monitoring methods have been designed for the AX32 ARM based platform using commercial-off-the-shelf components, in order to avoid the generation of large numbers of interrupt requests to critical cores from faulty peripherals [17].

At software level, the two-step degradation concept was introduced to allow a multicore platform to switch from a normal state to a safe state when non-critical errors are detected, or to an isolation state whenever critical errors are detected. Employing memory and time partitioning as well as the corresponding monitoring mechanisms allows one to present ‘non-interference’ arguments and furthermore, individual partitions can be modified without affecting the safety-critical design of other partitions. Additional useful techniques used were monitoring, error detection and error correction for bus communication, as well as control of the inter-core communication.

D. Specification methods

1) Formal specifications

Current safety standards do not include formal methods such as formal specification, refinement and model checking

as the main suggested quality assurance methods, but instead they heavily rely on traditional methods such as high coverage test suites. However, tools like NuSMV and Uppaal have been used successfully within RECOMP for performing model-checking and respectively for timing analysis of safety-critical systems [21].

Additionally studied methods are Event-B [22] and the B-method [23], two refinement-based formal methods for correct-by-construction development of software intensive systems. These methods can be used to derive provably correct systems from abstract specifications. The methods are mainly concerned with functional correctness, where schedulability and other timing properties need to be checked separately [2]. Formal modeling can be used to analyze models on high level of abstraction and thereby find problems in the design early. This is envisioned to be the main benefit in order to make certification easier. This will also bring down certification costs. Abstract formal models can help in re-certification when only the implementation of a component changes, but the specification stays the same. Correctness arguments for the complete system can then be preserved, provided that the specification captures the necessary information. The tools for the B-method and Event-B are not qualified according to any safety standard used in RECOMP. Nevertheless, the B-method was successfully used in the past in the railway industry [24].

SIMULINK [25] is another framework which uses formal specifications for the development of mixed-criticality systems. Using formal proof techniques, one can verify different properties of the system such as non-occurrence of feared events, the availability of the system, correctness and temporal properties, robustness (i.e. safe state preservation), as well as verifying different pre/post-conditions and correctness of functions.

SIMULINK can be used in conjunction with other MATLAB based tools for supporting a model-based development process including code generation. Additions developed within RECOMP allow one to prove not only that the model is correct but also that the generated code fulfills certain safety requirements [2].

2) Modular safety cases

A safety case is a requirement in many safety standards. Since modular certification helps in reducing costs of using multicores, creating modular safety cases has been one of the topics of interests in this project. However, one of the main challenges that modular certification has in contrast with modular design is that certification must consider not only the regular operation but also abnormal operation and malfunctioning components. To address these issues an approach was proposed to establish a modular and compositional construction for safety cases that has a correspondence with modular structure of the underlying architecture. As with system architecture, it would require to be possible to establish interfaces between the modular elements of the safety justification such that safety case elements may be safely composed, removed and replaced. Similarly as with system architecture, it will be necessary to establish the safety argument infrastructure required in order to support modular reasoning. The strategy is to define a set of claims related to platform level hazards. The claims are decomposed using sub-claims, evidence and other argumentation elements that are represented using the Goal Structuring Notation (GSN) [12][26] .

E. Reducing Tool Qualification Costs

The new standards for the development of safety critical systems such as ISO26262, DO-178C / DO-330 [10] and IEC61508 require analyzing all tools that are used within the development process of the software. This includes also the integration and verification of the software. All these standards have a three phase approach for using tools safely:

- Classification: the tools are classified into classes that describe the confidence (certification credit) they require in the development process of the system. The classification is based on the analysis of potential errors in the tool and their detection or prevention probability within the process.
- Qualification: Tools that require confidence in the analyzed processes have to be qualified. Qualification might be restricted to the identified use cases and to show the absence of critical errors. In the ISO26262 there are many qualification methods suggested (proven in use, process assessment, validation and development according to a safety standard).
- Usage: The tools can be used according to the known or found restrictions in the development process. There should be a documentation that contains the constraints from the process that have been considered in the analysis phase and workarounds for all restrictions found during tool qualification.

Obviously all three steps (classification, qualification, usage) cause costs. While classification of a tool (within a given tool chain) is not so expensive (experience is that this can be done with two to five days using Tool Chain Analysis (TCA) tool which is developed within the research project RECOMP), the qualification of tools can be very much effort, especially if there are no adequate qualification kits available that cover the relevant use cases. Furthermore also the process costs, for detecting or avoiding errors in unqualified tools or unqualified functions of tools can be costly (for example using a tool redundancy strategy).

The TCA (see Fig. 2 and [11]) allows one to model a tool chain structure in terms of tools having use cases which are reading or writing artifacts. The TCA also allows to compute the confidence level for every tool and use-case from the model that consists of tools, artifacts, use cases, features, potential errors, checks and restrictions as shown above together with their error detection probability which is the basis for computing the confidence needs.

The TCA also offers lots of plausibility checks for the tool chain and confidence model. For example if an detection measure from Tool B is assigned to a potential failure of tool A then there must be a data flow in terms of input/output artifacts from tool A to tool B, otherwise the assignment of this detection measure is invalid.

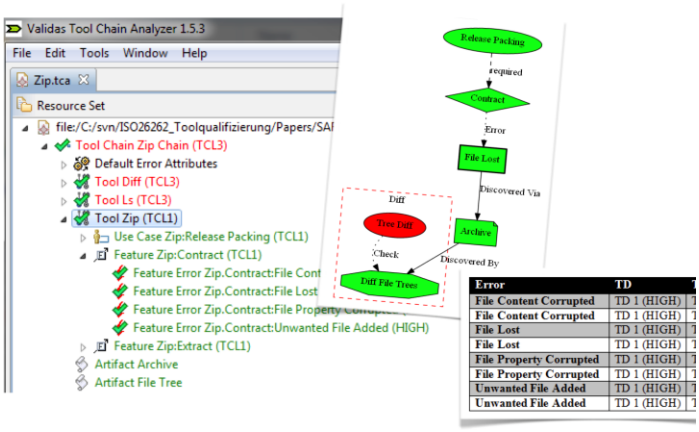

Fig. 2. Tool Chain Analyzer with generated figures and tables (MS Word)

The TCA can also generate a MS Word report, which contains detailed tables and figures for each identified potential tool failure, such that the computed TCL becomes plausible, comprehensible and checkable by review. The structure of this word report is designed such that it can be directly used as a part for the tool criteria evaluation report required by ISO26262.

As an example, the tool chain analysis method has proven to reduce the tool qualification needs, in one example by over 90% [9], so it has a big part in the success that the certification costs could be reduced using the methods developed in the RECOMP project.

V. Conclusions

Development of the next generation of main safety standards (IEC61508, ISO26262) starts in the coming years, and the publication phase is approximately in 2020. RECOMP results can be significant contribution especially for IEC61508, because they are current State-of-the-Art. Multicore platforms and their methods and tools provide a consistent additional set in current methods and tools. Also many current methods become obsolete during this decade and can be replaced with more effective, modern methods.

Acknowledgment

This work has been partially funded by the National Authorities involved in RECOMP and the Advanced Research & Technology for Embedded Intelligence and Systems (ARTEMIS) within the project RECOMP under Grant agreement no. 100202. Any opinions, findings and conclusions or recommendations expressed in this article are those of the authors and do not necessarily reflect the views of funding agencies. The authors would like to thank all partners and WP leaders for constructive collaboration in creating project results.

References

[1] Standards Amendment targeting multiple processor safety systems (SMP, AMP). Deliverable D4.2a.1 in RECOMP.

[2] Development process amendments to achieve a robust and safe development. Deliverable D4.2a.2 in RECOMP.

[3] International Electrotechnical Commission, IEC61508, Functional safety of electrical/electronic/programmable electronic safety-related systmes, Edition 2.0, Apr 2010 (7 parts).

[4] ISO/IEC15504-5, Information technology – Process assessment - Part 5: An Exemplar Process Assessment Model. 2012.

[5] International Organization for Standardization. ISO26262 Road Vehicles –Functional safety–. 1st Edition, 2011-11-15 (10 parts).

[6] B. Baert and S. Luys. Application of multicore CPUs in a safety-critical context. Internet: http://www.barco.com/barcoview/downloads/multicore_CPUs_in_safety-critical_applications.pdf, 21 March 2011. [Accessed: 6 December 2012].

[7] L.M. Kinnan. Use of multicore processors in avionics systems and its potential impact on implementation and certification, Digital Avionics Systems conference (DASC '09), Orlando, USA, 2009.

[8] R. Fuchesen. How to address certification for multicore based IMA platforms: Current status and potential solutions, Digital Avionics Systems Conference (DASC '10), Salt Lake City, USA, 2010.

[9] M. Wildmoser et al. *ISO26262 - Tool Chain Analysis Reduces Tool Qualification Costs.* In SAFECOMP 2012.

[10] RTCA. DO-330: Software Tool Qualification Considerations 1st Edition 2011-12-13.

[11] Tool Chain Analyzer Tool, can be downloaded from www.validas.de/TCA.html [Accessed: 6 December 2012].

[12] Origin Consulting GSN Community Standard Version 1 (2011)

[13] S. Voss and B. Schätz, *Scheduling shared memory multicore architectures in AUTOFOCUS 3 using Satisfiability Modulo Theories*, MBEES 2012 Daghstuhl Workshop, 2012.

[14] A. Hattendorf, A. Raabe, and A. Knoll: Shared memory protection for spatial separation in multicore architectures. SIES 2012: 299-302.

[15] M. Šimková, O. Lengál, M. Kajan: HAVEN: An Open Framework for FPGA-Accelerated Functional Verification of Hardware, FIT-TR-2011-05, Brno, CZ, FIT BUT, p. 16, 2011.

[16] K. Kähkönen et al. LCT: An Open Source Concolic Testing Tool for Java Programs. In Proceedings of the 6th Workshop on Bytecode Semantics, Verification, Analysis and Transformation (BYTECODE'2011), pages 75-80, Saarbrucken, Germany, Mar 2011.

[17] J. Strnadel.: Proposal of Flexible Monitoring-Driven HW/SW Interrupt Management for Embedded COTS-Based Event-Triggered Real-Time Systems, In: Proceedings of the Work-in-Progress Session of the 32nd IEEE Real-Time Systems Symposium, Vienna, AT, TUV, 2011, s. 29-32

[18] Uppsala University. "Times Tool", Internet: http://www.timestool.com/, [Accessed: 6 December 2012].

[19] F. Singhoff. "The Cheddar project: a free real time scheduling analyzer", Internet: http://beru.univ-brest.fr/~singhoff/cheddar/, Sep. 2011. [Accessed: 6 December 2012].

[20] Symtavision, "SymTA/S", Internet: www.symtavision.com. [Accessed: 6 December 2012].

[21] X. Gan, J. Dubrovin and K. Heljanko: A Symbolic Model Checking Approach to Verifying Satellite Onboard Software. In Proceedings of the 11th International Workshop on Automated Verification of Critical Systems (AVoCS 2011), Newcastle, UK, September 2011.

[22] J.-R. Abrial. Modeling in Event-B: System and Software Engineering, Cambridge University Press, 2010

[23] J.-R. Abrial. *The B-Book: Assigning Programs to Meanings*, Cambridge University Press, 1996.

[24] P. Behm, P. Desforges and J. M. Meynadier. Météor: An Industrial Success in Formal Development. In *B'98: Recent Advances in the Development and Use of the B Method*, LNCS 1393, Springer, 1998.

[25] MathWorks. "SIMULINK". Internet: http://www.mathworks.com/products/simulink/. [Accessed: 6 December 2012].

[26] A. Ruiz, I. Habli, and H. Espinoza: Towards a Case-Based Reasoning Approach for Safety Assurance Reuse. In Proceedings of the 2012 International Conference on Computer Safety, Reliability and Security, SAFECOMP'12, p.22-35, Magdeburg, Germany, 2012.