NORTH - Non-intrusive Observation and RunTime verification of cyber-pHysical systems*

José Rufino, António Casimiro, Antónia Lopes

LASIGE, Faculdade de Ciências, Universidade de Lisboa, Portugal; email: {jmrufino, casim, malopes}@ciencias.ulisboa.pt

Frank Singhoff, Stéphane Rubini, Valérie-Anne Nicolas, Mounir Lallali, Mourad Dridi, Jalil Boukhobza, Lyes Allache

Lab-STICC UMR CNRS 6285, Université de Bretagne Occidentale, UBL, Brest, France; email: {singhoff, rubini, vnicolas, lallali, mourad.dridi, boukhobza}@univ-brest.fr

Abstract

The increase use of autonomous vehicles and other cyber-physical systems has motivated the adoption of Runtime Verification (RV) techniques for embedded systems. This stems from the criticality of such systems, which call for the assurance of correct operation, both on value and time domains. However, traditional RV techniques (mostly based on code instrumentation) may inevitably pose significant overheads, both in performance and timeliness, due to their inherent intrusiveness, which make them clearly unfit for critical systems.

This paper aims at advancing the state-of-art in RV techniques by presenting an innovative research observation and runtime verification methods, supported in non-intrusive monitoring machinery. The negative effects of traditional techniques (ranging from function call interception to source code annotation with observation points) are avoided, making this novel approach relevant to virtually all (critical) cyber-physical systems.

1 Introduction and Motivation

Autonomous vehicles are finding their way into more applications every day. For example, drones for surveillance. These vehicles include on-board computing systems that are based on embedded processing elements, performing the necessary control functions to perform the vehicle's mission.

Given that the interaction with the environment may have very high safety requirements, the correctness of the overall system is paramount, and should be ensured at all times. This may be envisaged as a general framework of the so-called Cyber-Physical Systems (CPS), which mechanisms controlled and/or monitored by computer-based techniques [1].

*This work was partially supported by FCT, through funding of LASIGE Research Unit, ref. UID/CEC/00408/2013, and by FCT/CAMPUS FRANCE (PHC PESSOA programme), through the transnational cooperation project 3732 (PT) / 37932TF (FR), Non-intrusive Observation and RunTime verification of cyber-pHysical systems (NORTH). This work integrates the activities of COST Action IC1402 - Runtime Verification beyond Monitoring (ARVI), supported by COST (European Cooperation in Science and Technology).

In order to satisfy the demanding timing, safety and security properties, some sort of correctness verification procedures are needed during the execution (runtime) stage of the system, assessing its state against a previously defined specification. The systematic and well-defined use of such procedures is called Runtime Verification (RV) [2].

Most of the current RV techniques require the modification of the application source-code. Although such code instrumentation is reasonable for larger systems, the timeliness requirements together with scarce resource availability that characterise autonomous and vehicular systems may pose an unsurpassable challenge for runtime verification in such kind of systems. Other techniques, such as system and/or function call interception, are also not free from intrusiveness [3, 4].

The goal of this work is to enable non-intrusive observation and runtime verification techniques in cyber-physical systems. This calls for advanced models, methodologies and mechanisms. Non-intrusive RV needs a novel approach based on accurate modelling of system components and embedded processing elements. These models will then be strengthened by the usage of formal temporal logics for verification rule definition, based on specifications and model properties. Then, rules are verified in runtime by independent (non-intrusive) hardware observation and monitoring mechanisms.

A comprehensive set of properties, ranging from timeliness to safety and security, shall be monitored. Timeliness and safety are crucial for the correctness of vehicle operation and for mission survivability. Security proprieties and resilience to intrusion attacks is mandatory in vulnerable systems subject to an increasing number of threats. Therefore novel and innovative RV techniques are in need to be applied to the CPS realm. To avoid the intrusiveness shortcomings of traditional RV techniques, the observation of the system must be made non-intrusive, a fundamental step for the NORTH project.

The paper is organized as follows. Section 2 introduces the NORTH work flow. Section 3 focus on the non-intrusive NORTH features while Section 4 discusses the evaluation of those features in NORTH-inspired systems. Section 5 describes the related work and, finally, Section 6 presents some concluding remarks and future research directions.

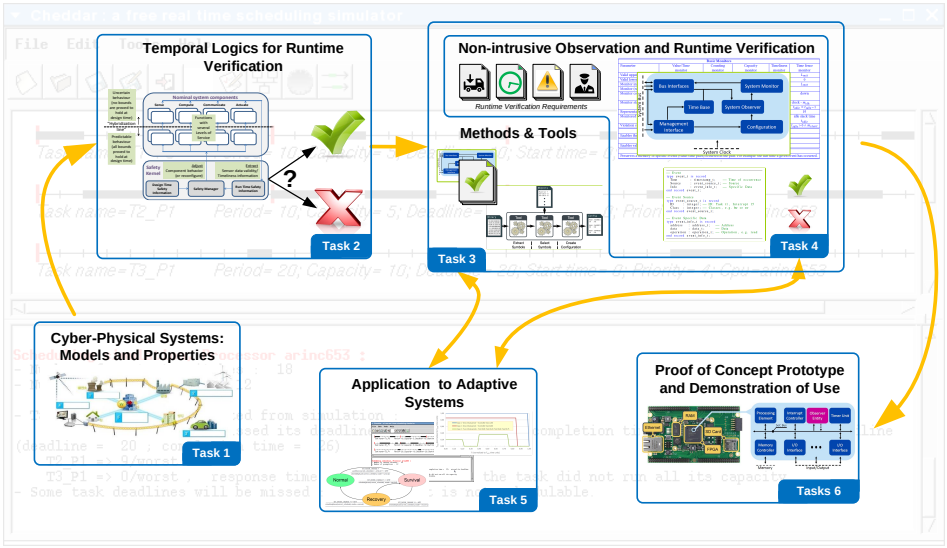

Figure 1: The NORTH project work flow.

2 NORTH Project Work Flow

Understanding and formalising the properties of a cyber-physical system and how temporal logics can be used to verify those properties, is one key point of research. The gained knowledge will then be consolidated in a methodology and in the architecture of a tool for assisting in the definition and expression of runtime verification mechanisms. These should be combined with mechanisms for self-adaptability, allowing the dynamic definition of new sets of observation points. A proof of concept demonstrator, merges the outcomes of the activities, sketched in the diagram of Figure 1.

T1. CPS Component Modelling and Property Extraction this task aims at modelling the several components in a cyber-physical system, both hardware and software, identifying/extracting the properties which can be subjected to runtime verification activities. The hardware may include not only the computing platform but also a relevant set of sensors/actuators. The software part of the system includes naturally the control components. However, it may include also properties of data processing (validity and fusion), (self-)awareness, (self-)adaptability, perception, collaboration, among others. On the environment level, dealing with uncertainly is a must.

T2. Temporal Logics for Runtime Verification of CPS this task aims at exploring the applicability of Temporal Logics to the runtime verification of cyber-physical systems, and therefore study how the properties extracted in the previous task can be verified in runtime by such logics. Properties that should be verified in runtime may include: load bounds, timeliness, safety and security.

T3. Methods and Tools for Runtime Verification of CPS this task aims at defining the architecture and flow for the integration of the results stemming from the previous task into the specification of runtime verification clauses. The definition and design of special-purpose tools, augmenting and extending the scope of existing standard tools (e.g., the GNU binutils), may be required for adequate assistance in this process. This task defines a set of points of interest that should be non-intrusive observed and verified: variables and data-types; dynamically allocated memory (if exists); system or function calls; exception handling.

T4. Non-intrusive Observation and Runtime Verification this task aims at designing a runtime verification system, using non-intrusive observation and monitoring machinery, implemented in hardware, as a basis for supporting runtime verification. The results from the previous tasks are used to map the temporal logic formulas into sets of data and/or event observation points, to be configured in the hardware machinery, and into runtime verification assertions, to be checked on the monitoring activities, which may then take a decision on system correctness. The RV system may be restricted to use a few fundamental set of monitors complemented with some additional functional blocks.

T5. Adaptive Non-intrusive Observation and RV of CPS this task aims at designing self-contained hardware-based mechanisms allowing the dynamic definition of new sets of observation points, due to normal changes in the operational conditions of the system, while maintaining the same runtime verification assertions. For example: a running program makes a function call (either recursive or not) that creates a new stack frame; runtime verification needs to define a new set of observations points for the recently created stack frame but the runtime verification assertions (e.g., non-violation of stack frame boundaries) are invariant.

T6. System Prototype for RV and Demonstration of Use this task aims at creating a prototype merging both the toolset architecture and flow to demonstrate the usage of effective runtime verification for cyber-physical systems.

3 Non-intrusive Runtime Verification tools

To experiment the approach proposed in NORTH, we have developed two tools. The first one monitors a target system at runtime and triggers the collection of a trace of scheduling events whenever some condition on the sequence of events is not respected. The second tool can verify on line more complex temporal scheduling properties from the execution traces.

3.1 Scheduling monitoring tool

More precisely, the aim of the first tool, which is called the health monitor, is to verify, at runtime, the conformity of a cyber-physical real-time system with the specification of its task model. The task models may be defined during the early steps of a design process, and specifies the execution sequence and duration of the software tasks. The engineers use this model to verify the schedulability of their design before execution, by the mean of scheduling simulation tools such as Cheddar [5] for instance.

A scheduling simulation result constitutes a deterministic reference that can configure the health monitor. We propose a hardware implementation of a monitor which integrates a micro-sequencer in charge of verifying the occurrence of a timed sequence of scheduling events on the target system. The hardware also includes a module for the event capture, an event recorder, a timestamp generator and some communication logic to transfer the event traces. In case of erroneous behaviour, with respect to the expected task scheduling, the event trace is sent to a supervision station for further analysis.

3.2 On-line verification tool

The second tool is a verification tool which performs the temporal scheduling properties verification on system execution traces. It has to be embedded as a component of the health monitor, and to be executed in line during the system execution. Therefore, its execution speed has to be compliant with the system requirements and its memory footprint must stay as low as possible. In addition, the monitored systems may have non finite executions or long finite executions. During execution, the verification tool should not consider as input the whole trace, but only a finite fixed size slice of it (by using a transition buffer filled by the monitor hardware part). The verification tool execution time on one slice must be lower than the system execution time corresponding to the next trace slice, otherwise some trace events may be lost. For all these reasons, the verification algorithm performs only one pass through the trace.

The verification tool adopts the same system model and trace model used in the Cheddar tool. Its verification algorithm is based on a representation of the system state (including task and resource states), and starting from an inactive initial state (built from the system model), and simulates the execution represented by the trace, event by event. At the same time, and depending on the properties to verify, some checks are done on event occurrences or periodically at the end of each same time sequence. Periodic checks concern the tasks reaching the end of their period, and are needed to cope with possible missing events in the trace, such as missing task activation events. It also allows us to complete the detection of undue locked resources, or task missed deadline.

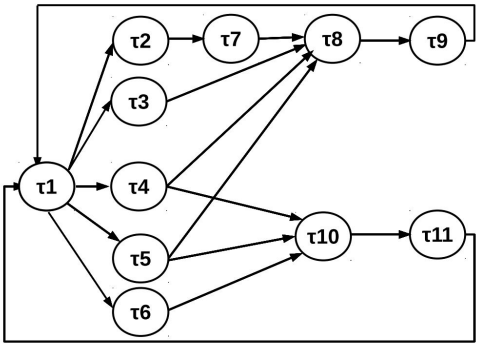

Figure 2: Modeling of the ROSACE application for Cheddar

4 Evaluation

The tools described in the previous section have been validated by several experiments.

The monitoring tool has been implemented on a Xilinx System-On-Chip (SoC) Zynq7000. The hardware monitor has been synthesized from a VHDL[2] model and the resulting design occupies less than 5% of the FGPA resources.

A simple experiment composed of two tasks has been successfully run to demonstrate the ability of the monitor to record the corresponding scheduling event on a Real-Time Executive for Multiprocessor Systems (RTEMS) target [6].

The online verification tool has been implemented in C and applied on several case examples produced from Cheddar. From those examples, four properties have been identified: missed deadlines, deadlocks, priority inversion and lock resources.

Larger experiments based on a mixed-criticality systems case study are also planned during the project.

As any mixed-criticality system, the case study is composed of several applications with different levels of criticality [7]. In this context, the different levels of criticalities are different levels of guarantee on the application deadlines. As an example, a two level mixed-criticality system may contain high-criticality applications on which the deadlines must be met and low-criticality applications on which the deadlines are allowed to be missed.

The case study of the NORTH project is a drone system with 3 criticality levels [8]. The highest criticality level is a flight controller software called ROSACE [9]. ROSACE is a data flow oriented application composed of a set of periodic dependents tasks and requires that all task deadlines are met. Figure 2 shows a model of ROSACE made for Cheddar. Each task is defined by the classical periodic task parameters, i.e. WCET (Worst Case Execution Time), period, priority, release time and deadline.

[2] **V**ery **H**igh-Speed Integrated Circuit **D**escription **L**anguage.

The dependencies between tasks are expressed by a specific priority assignment. The middle criticality level application is a path planning algorithm computing online the path of the drone according to its environment.

Finally, the lowest criticality application is a video application with a high computation need which simulates a video surveillance system embedded in the drone. All the applications run on a POSIX RTEMS [6] target and are written in C.

5 Related Work

The application of non-intrusive runtime monitoring to embedded systems has been discussed in [3, 4] and, more specifically, in safety critical environments [10].

Configurable minimally intrusive event-based frameworks for dynamically runtime monitoring have been developed [11]. Additionally, the RV concept has been applied to autonomous systems [12] and to a AUTOSAR-like RTOS, aiming the automotive domain [13]. A runtime monitoring approach for autonomous vehicle systems requiring no code instrumentation by observing the network state is described in [14].

However, to the extent of our knowledge, no such techniques have been applied to aerospace systems, specially if critical avionic applications are combined with other non-critical applications.

6 Conclusion

In this paper we have presented NORTH. The NORTH project is a collaborative project between the LASIGE/Univ. Lisboa and the Lab-STICC/Univ. Bretagne Occidentale aiming to investigate and experiment a runtime verification platform for embedded real-time systems.

The NORTH project addresses the study of Non-intrusive Observation and Runtime Verification in a comprehensive way, from conceptual tasks such as component modelling and property extraction to implementation and prototyping, passing through methods and tools for building a (self-)adaptive RV architecture.

Those propositions led to the development, up to the moment, of two tools: a hardware scheduling monitor implementing on a Field Programmable Gate Array (FPGA) board and running a RTEMS target, and a runtime analysis tool written in C and allowing on line detection of task scheduling errors.

In the next steps of the project, the tools will be experimented on a drone case study running a mix-criticality system. A mix-criticality system is both composed of high and low criticality tasks and cannot be designed by resource reservation only. Thus monitoring and verification for online resources management is expected to be an interesting approach in this context.

References

[1] J. Sztipanovits, X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and S. Wang, "Toward a science of cyber-physical system integration," *Proc. of the IEEE*, vol. 100, Jan. 2012.

[2] M. Leucker and C. Schallhart, "A brief account of runtime verification," *The Journal of Logic and Algebric Programming*, vol. 78, pp. 293–303, May-Jun 2009.

[3] C. Watterson and D. Heffernan, "Runtime verification and monitoring of embedded systems," *Software, IET*, vol. 1, Oct. 2007.

[4] T. Reinbacher, M. Fugger, and J. Brauer, "Runtime verification of embedded real-time systems," *Formal Methods in System Design*, vol. 24, no. 3, pp. 203–239, 2014.

[5] F. Singhoff, J. Legrand, L. N. Tchamnda, and L. Marcé, "Cheddar: a flexible real time scheduling framework," *ACM Ada Letters journal*, vol. 24, pp. 1–8, 2004.

[6] *Real-Time Executive for Multiprocessor Systems - RTEMS POSIX Application Programmimg Interface Guide*, release 5.0.0 ed., 2017. https://www.rtems.org.

[7] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Proceedings of the* 28^{th} *Int. Real-Time Systems Symposium (RTSS)*, IEEE, Dec 2007.

[8] L. Allache, "Mise en oeuvre d'un benchmark drone/mix-criticality," in *Master thesis, Master 2 LSE*, 2018.

[9] C. Pagetti, D. Saussié, R. Gratia, E. Noulard, and P. Siron, "The ROSACE case study: From simulink specification to multi/many-core execution," in *IEEE 20th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 309–318, IEEE, 2014.

[10] A. Kane, *Runtime Monitoring for Safety-Critical Embedded Systems*. PhD thesis, Carnegie Mellon University, USA, Feb. 2015.

[11] J. C. Lee and R. Lysecky, "System-level observation framework for non-intrusive runtime monitoring of embedded systems," *ACM Transactions on Design Automation of Electronic Systems*, vol. 20, no. 42, 2015.

[12] G. Callow, G. Watson, and R. Kalawsky, "System modelling for run-time verification and validation of autonomous systems," in *Proc. 5th Int. Conf. on System of Systems Engineering*, (Loughborough, UK), June 2010.

[13] S. Cotard, S. Faucou, J.-L. Bechennec, A. Queudet, and Y. Trinquet, "A data flow monitoring service based on runtime verification for AUTOSAR," in *Proceedings of the 14th Int. Conf. on High Performance Computing and Communications*, (Liverpol, UK), IEEE, June 2012.

[14] A. Kane, O. Chowdhury, A. Datta, and P. Koopman, "A case study on runtime monitoring of an autonomous research vehicle (ARV) system," in *Proc. 15th Int. Conf. on Runtime Verification*, (Vienna, Austria), Sept. 2015.