The ADMORPH approach for Adaptively Morphing Embedded Systems

A. Pimentel, C. Grelck, L. Miedema, D. Sapra
Informatics Institute, University of Amsterdam; email: {a.d.pimentel, c.u.grelck, p.l.miedema, d.sapra}@uva.nl

M. Völp, F. Lucchetti, A. Matovic
SnT - Université du Luxembourg; email: {marcus.voelp, federico.lucchetti, aleksandar.matovic}@uni.lu

M. Maggio, N. Vreman
Department of Automatic Control, Lund University; email: {martina.maggio, nils.vreman}@control.lth.se

S. Altmeyer, F. Haas
Augsburg University; email: {altmeyer, haas}@es-augsburg.de

A. Casimiro, J. Cecílio, G. Jäger, A. Espindola
LASIGE, University of Lisbon; email: {casim, jmcecilio, gjager, adespindola}@ciencias.ulisboa.pt

S. Skalistis
Collins Aerospace; email: stefanos.skalistis@collins.com

J. Kouwer, G. de Lange
Thales Nederland B.V.; email: Jeroen.Kouwer@nl.thalesgroup.com

J. Almeida, H. Blasum, M. Brotz, S. Wagner
SYSGO SAS and SYSGO GmbH; email: {jose.almeida, holger.blasum, mario.brotz, stephan.wagner}@sysgo.com

P. Novobílský
Q-media; email: pno@qma.cz

Abstract

Due to the increasing performance demands of mission- and safety-critical Cyber-Physical Systems (of Systems), these systems exhibit a rapidly growing complexity, manifested by an increasing number of (distributed) computational cores and application components connected via complex networks. However, with these systems' growing complexity and interconnectivity, the chances of hardware failures and disruptions due to cyber-attacks will also quickly increase.

In the ADMORPH project we explore system adaptivity, in terms of dynamically remapping application components to processing cores, to fuse fault- and intrusion tolerance with the increasing performance requirements of mission- and safety-critical CPS(oS). This paper describes the overall ADMORPH architecture and provides an overview of the developed methodologies, methods and tools for the specification, design, analysis and runtime deployment of adaptive mission- and safety-critical CPS(oS) that are robust against both component failures and cyber-attacks.

Keywords: Cyber-Physical Systems, Adaptation, Resilient control, Design Space Exploration

1 Introduction

Cyber-Physical Systems (CPS) form a crucial information-technology domain worldwide, that covers many industrial sectors, including: health industries, industrial automation, avionics, and space. CPS often consists of heterogeneous, multi- or many-core systems that are distributed and connected via complex networks, creating what is known as Cyber-Physical Systems of Systems (CPSoS).

Designing CPS(oS) systems is challenging due to the stringent and often conflicting extra-functional design requirements they must meet. A single task in the system that misses its computational deadlines can have severe – sometimes even life-threatening – consequences for mission- or safety-critical CPS(oS). Safety-critical CPS(oS) demand ultra-high levels of dependability, which is becoming even more important as the levels of system autonomy rise.

To ensure reliability, availability, and safety, mission- and safety-critical CPS(oS) must be able to handle various disruptive events caused by hardware component failures or cyber-attacks like Denial-of-Service (DoS) attacks aimed at disrupting the system or compromising critical system functionality.

System adaptation offers a promising technique to maintain

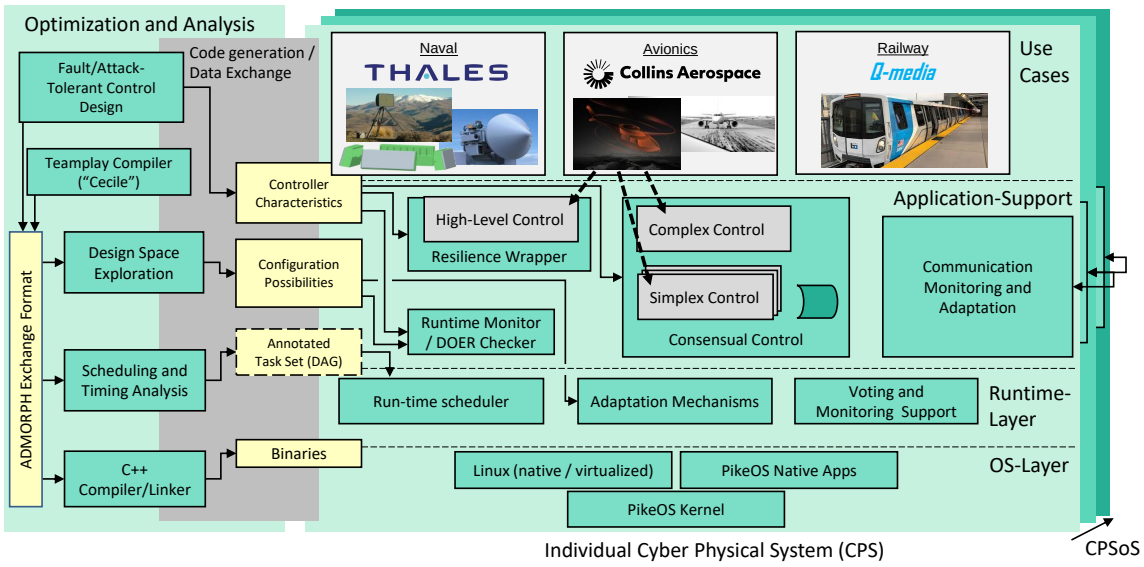

Figure 1: Global perspective of the ADMORPH architecture.

the system's operation at the required level of Quality of Service (QoS) or achieve graceful degradation when disruptive events hit the system. Allowing dynamic task relocation or replication between different processors, processor types, or even across hardware/software boundaries yields morphing systems. These enable applications to execute using a variety of different and dynamically interchangeable system configurations over heterogeneous system resources. Morphing systems are the key to providing a real breakthrough in establishing robustness against unpredictable, disruptive events that affect the dependability of systems, prolonging their life and maximizing their efficiency during this lifetime.

The absence of systematic methodologies to design and runtime-manage complex, adaptive embedded systems is holding back their progress. To address this issue, the ADMORPH project[1] is investigating holistic approaches to systematically designing, analyzing, and managing embedded computer systems in mission- or safety-critical CPS(oS). The approach uses the concept of system adaptation in the form of dynamic task-to-resource allocation to achieve fault and intrusion tolerance.

Adaptation is crucial for coping with faults over the long run, provided the system can tolerate faults long enough for adaptation to become effective. Moreover, it allows us to optimize both functional and non-functional properties of these systems. In ADMORPH, we design and develop adaptation strategies (1) to evade faults by relocating services to a different set of resources, (2) to improve resilience by including more resources in the tolerance of faults, and (3) to match the systems' resilience to the perceived threat by allocating more or less resources to the defense.

This paper presents an overview of the ADMORPH architecture and its key components (Figure 1). The architecture encompasses Optimization and Analysis components (including design space exploration, a coordination language, and scheduling and time analysis), Application Support components (providing various adaptation strategies, with the support of monitors), and considers Runtime and Operating System layers that are designed to provide real-time support for applications, for different adaptability requirements. All these can be used in multiple application scenarios, and in ADMORPH we consider three specific Use Cases to validate the several components of the architecture.

2 Optimization and Analysis

The embedded computer systems in CPS(oS) are often composed of multicore systems built for a specific application, operating a combination of homogeneous and heterogeneous cores on a single chip. Modelling and exploring embedded multicore systems is a time-consuming and a complex task. Moreover, these devices can be deployed for a long term and therefore system lifetime reliability is an important consideration while designing them. However, it is imperative that over long periods of deployment time, some of the cores in such systems will start to deteriorate owing to the ageing process and will eventually fail. In principle, placing extra cores on the chip can increase the lifetime reliability albeit at the cost of increased power consumption and chip area.

2.1 Design Space Exploration

In ADMORPH, we presented a framework [1] to explore the design space of platform architectures and their floorplans, with the contradictory objectives of increasing lifetime reliability and lowering the average power consumption. The exploration algorithm in this framework, based on a Genetic Algorithm, returns the *Pareto Set* from the population upon convergence. The design points in the *Pareto Set* exhibit the trade-off between two objectives and one cannot be considered better over the other w.r.t both the objectives [2]. The framework employs a high level simulator to calculate the Mean Time to Failure (MTTF) of the chip. A higher MTTF of a chip represents the fact that the chip will fail after longer duration and hence is predicted to operate for longer duration

[1] https://www.admorph.eu

after initial deployment. In this context, MTTF can be interpreted as an estimate of active lifetime of the chip and thus its reliability.

The simulator runs multiple times, also called Monte Carlo simulation, to take the averages of both failure times and power usage. The high number of simulations required makes the framework compute intensive. We therefore also proposed variations of the exploration methodology to reduce the total number of simulations needed for a faster convergence of the main algorithm.

2.2 TeamPlay Coordination Language

The TeamPlay Coordination Language [3] supports the specification of CPS(oS) at a very high level of abstraction. TeamPlay adopts the principle of *exogenous coordination* as a key design choice and achieves a complete separation of concerns between the specification of stateless components (aka tasks) and their orderly interaction in a streaming network. Components are characterized by a set of functional as well as non-functional contracts. The functional contracts consist of a set of typed input and output ports as well as a set of state ports, i.e. output ports that are short circuited to corresponding input ports, and hence allow us to mimic state in an otherwise stateless world. The non-functional contracts span from average and worst-case execution time (on a given hardware unit) to average and worst-case energy consumption (on a given hardware unit) to fault-tolerance execution regimes, such as dual or triple modular redundancy.

Components may have multiple versions that behave identically with respect to the functional contracts, but typically expose different behaviour with respect to the non-functional contracts, even when run on the same hardware. As a coordination language TeamPlay focuses on the specification and interaction of components and leaves the implementation of components to lower-level languages, usually C or C++ in the domain of cyber-physical systems.

This design permits TeamPlay to adapt CPS(oS) applications to the available hardware, usually commodity-off-the-shelf high-performance embedded systems, under varying objectives such as meeting deadlines, energy budgets or robustness requirements [4]. To this effect we have proposed various scheduling algorithms [5, 6], as well as the adaptive runtime environment YASMIN [7] and investigated the best use of constrained resources for fault-tolerance under a weakly-hard real-time regime [8], among others.

2.3 Scheduling and Time Analysis

For the scheduling, we assume an input AFC-file that models an application as a directed acyclic graph, divided into section with varying redundancy levels. This means that redundancy levels can change dynamically in between section boundaries. The level of redundancy at runtime depends on the number of available processing elements and the current fault rate. To enable the dynamic adaptation depending on the number of cores and fault-rate, we use a set of pre-computed schedules for the different sections that can be loaded and executed at runtime. For the computation of the various schedules, we use a modified version of the Heterogeneous Earliest Finish Time (HEFT) scheduler that ensures a suitable mapping of redundant tasks on the processing elements. Based on this scheduler, we have developed *faktum*, a scheduling and analysis tool that serves two purposes: Firstly, it computes offline a set of fault-tolerant schedules, using the HEFT scheduler as described above, that can be fed to the design-space exploration. Secondly, it serves as a scheduling verification and analysis tool, which estimates the feasibility of design candidates during the design-space exploration. After the design-space exploration, it derives the final verdict on the suitability of the chosen candidate architecture.

2.4 Analysis of the Impact of Deadline Misses on Control Systems

Feedback control is a central enabling technology in a wide range of applications. Control systems are at the core of energy distribution infrastructures, regulate the behaviour of engines in vehicles, and are embedded in household appliances like washing machines. Control is centred around the feedback mechanism. Sensors provide information about the current state of the physical environment. This is used to compute suitable control actions to fulfil performance requirements, that are then implemented by actuators. For example, adaptive cruise control systems use measurements from a range of sensors to determine how to adjust the throttle to automatically regulate the vehicle's speed, while maintaining a safe distance from vehicles ahead.

Control actions are often calculated using hardware and software. Hence, the computation of the new control signals is subject to accidental faults, systematic issues, and software bugs. In practice, these computational problems are often ignored. But when can this be done safely? In ADMORPH we introduce a framework for analyzing the behaviour of control software subject to computational problems. We started the analysis with the evaluation of the stability [9] and performance [10] of control systems subject to consecutive deadline misses. We then worked on generalising the analysis to all the other weakly-hard [11] task models [12, 13] and creating an experimental toolchain [14]. We also analysed the latency of complex pipelines in which tasks in a chain of dependent computations experience deadline misses [15] and discussed recovery strategies for control systems [16].

3 Adaptation Methods

Applications quite naturally adapt their functionality or performance to changing demands and environmental situations. For example, planes transition through modes for taking-off, flying, landing and taxiing from the runway to the parking position at the terminal. While the triggers for these changes are expected or at least well predictable, one cannot equally well predict when the system has to adapt to faults. In ADMORPH, we focus on exactly that prediction and on the adaptation of the fault and intrusion tolerance mechanisms that support such adaptation. Figure 2 shows the control architecture of ADMORPH.

In various scenarios, high-level controllers play a crucial role in guiding the behavior of lower-level control loops, which operate at a much higher frequency and prioritize the system's stability. Functional adaptation, involving transitions

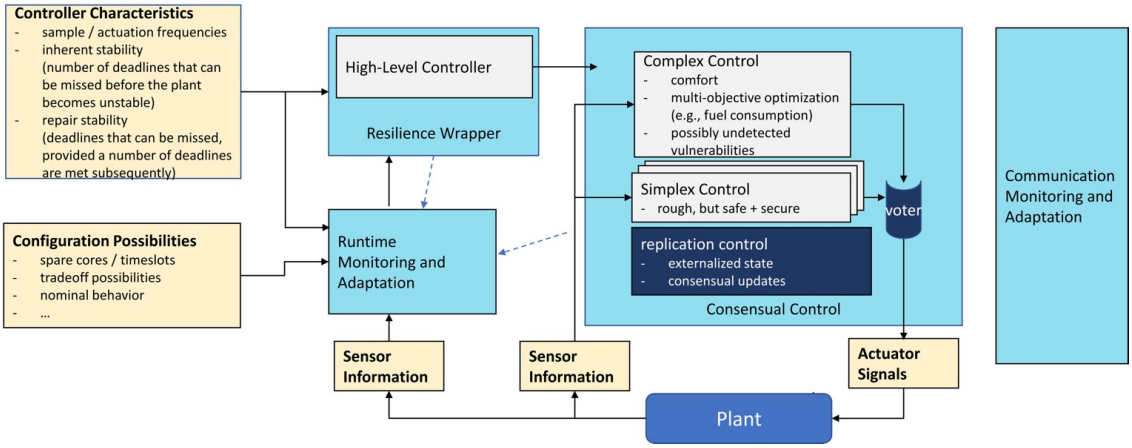

Figure 2: ADMORPH's control architecture.

between multiple high-level controllers, typically occurs during pre-defined configuration changes coordinated by the coordination language, its compiler, and toolchain. Threat-related adaptations, such as adjusting the internal resilience of components in response to perceived threat levels, require a combination of proactive resource planning for severe threats and swift runtime adjustments to ensure the system maintains the desired state when threats emerge. Both strategies require runtime adaptations within the control architecture to respond to unforeseen situations, utilizing available resources swiftly and planned configurations from design-space exploration. Such adaptations may include (a) activating the simplex subsystem if the complex fails to provide timely and accurate information, (b) modifying the simplex replication policy, changing the detection quorum that enhances resilience over subsequent control epochs, (c) relocating consistently failing controllers to spare resources and (d) adapting the frequency of rejuvenation. These runtime adaptations ensure efficient system response, leveraging redundancy and planned measures for effective performance in unforeseen circumstances.

However, control tasks are susceptible to failure or compromise over time as adversaries seek to gain control of the system and cause harm to the operating environment. Ensuring the recovery of these control tasks and the resources they utilize becomes crucial to maintain healthy majorities and withstanding failures. In ADMORPH, we have enhanced both the state-capturing capability of replicated controllers and their ability to restart replicas in a stateless manner, while also incorporating the capability to introduce additional replicas. These new replicas are initialized using pre-compiled and pre-analyzed binary images sourced from a diverse pool. This diversity thwarts adversarial knowledge regarding replica attacks. By starting the replicas without their previous states, the replication controller injects the captured state and maintains synchronization with the control tasks, ultimately transitioning the responsibility to control the device or cyber-physical system. This approach allows for the creation of additional replicas, which can later join the active consensus group once they are fully operational.

Another essential aspect that ADMORPH addresses is related to the time of adaptation and reconfiguration. In ADMORPH, we are considering finer-grained reconfiguration, and we do so by integrating fault models into the scheduling analysis of redundant dataflow tasks. This way, we determine the Worst-Case Execution Time (WCET) of such tasks in the presence of errors down to a specific probability.

Nevertheless, certain factors can still result in unbounded reconfiguration times, such as repeated resource reboots following a crash or the reactivation of the same software vulnerability. These situations often indicate a persistent fault within the hardware resource or the re-instantiated software component. In the ADMORPH approach, we are considering software diversification and relocation to address this issue. We have considered the implications of these actions on the WCET of the software component.

To address these concerns, the ADMORPH control architecture employs a three-step reconfiguration process: (1) the new configuration is initiated by creating new replicas, starting components that implement the new functionality, or utilizing spare resources, (2) the new configuration establishes connections with the existing setup, ensuring it receives state updates, sensor inputs, and is monitored effectively and (3) once the preparation phase is completed and all components confirm their readiness, control is transitioned atomically by updating the control system and associated components to consider the new subsystems instead of the previous configuration.

During this transition, the control system exclusively considers and applies proposals from the old configuration, leveraging its inherent fault tolerance for a specific duration. Once the new configuration is fully established, regardless of the time taken, the control system solely considers proposals from the new configuration at the beginning of the next control epoch. This decoupling of configurations allows timely control to be maintained by either the old configuration (until the transition point) or the new configuration (from the transition point onwards).

4 Runtime and OS support for resilient control

In ADMORPH, to ensure security and safety of the application workloads, we have used a separation kernel to provide

strict separation. Historically, a governing principle of a separation kernel is to enable static configurations for relatively simple embedded systems and applications [17]. With safety-critical embedded systems and CPS(oS) growing to more powerful hardware, more complex embedded systems become feasible, and, in part the ADMORPH use cases were helpful to gather demands for runtime and OS support.

We have shown that it is possible to extend the separation kernel on the sensing side by host intrusion detection by control flow integrity [18, 19], network intrusion detection with Suricata, and safety monitoring infrastructure for heterogeneous systems. For the acting side, we have extended the separation kernel experimental by more flexible scheduling (run-time adaptation of time windows and cross-CPU thread/task migration). We have also implemented mechanisms for FPGA reconfiguration. For secure update, we have demonstrated the use of Mender.

5 Use Cases

ADMORPH's adaptivity technology is being evaluated through three use cases. These use cases have been selected to cover a significant safety- and mission-critical CPS(oS) spectrum. They span different domains with varying system requirements and needs regarding robustness and quality of service.

5.1 Autonomous Aerospace Systems

Flight delays due to airport congestion create a costly ripple effect for airlines and airports. Enhancing aircraft autonomy in specific flight phases can mitigate these issues while ensuring safety remains crucial in unforeseen circumstances like system faults and security attacks. In a highly regulated environment, it is not yet clear who the decision maker for specific actions will be. In some cases, it will be Air Traffic Control (ATC), whereas in other cases, it will be the aircraft. This provides a perfect environment for System of Systems (SoS) demonstration where ADMORPH solution can be applied to provide adaptivity and safety.

Nowadays, in commercial aircrafts, the level of autonomy during the cruise is very high. However, takeoff, landing, and taxiing are still pending subjects and very critical. In this use case, we are implementing a hybrid simulation environment that combines model-in-the-loop and hardware-in-the-loop approaches. The co-simulation involves integrating Simulink and FlightGear, while real-time execution takes place on the Ultrascale+ platform. This configuration can be equipped with a safety-critical operating system or hypervisor, such as PikeOS, to ensure real-time characteristics. This setup mimics the functionality of a single computer unit within an Integrated Modular Avionics architecture, allowing for fault injection testing. These faults will simulate operational failures and attacks. Leveraging the expressive coordination language utilized in the project, detecting these faults will be used to trigger adaptivity measures, thereby ensuring the overall system's safety.

5.2 Radar Surveillance Systems

Radar surveillance systems are essential for ships as they provide crucial situational awareness, enabling vessels to detect other ships, obstacles, and navigational hazards, enhancing maritime safety and preventing collisions. This used case involves evaluating specific methods within the context of an industrial embedded software system (or subsystem) used for radar surveillance processing in a laboratory setup using a realistic processing platform. As command and control decisions require reliable and robust real-time data processing, the ability of the ADMORPH approach to achieve fault tolerance is substantially assessed and validated.

In this particular use case within the ADMORPH framework, the coordination language is crucial in specifying adaptive systems alongside adaptive runtimes. This approach ensures formal guarantees and facilitates comprehensive testing of the runtime system itself.

5.3 Transport system

This use case focuses on evaluating ADMORPH approaches within the context of a Train Supervision Surveillance System, which falls under the category of a System of Systems (SoS). The railway system operates multiple information flows between ground systems and moving trains, each serving specific purposes with varying levels of criticality. In the event of incidents, it becomes crucial to maintain communication channels associated with the most critical information flows. Hence, a system capable of adapting to diverse operating conditions such as signal level, channel interference, hardware faults, and line overload is essential. In this use case, the TeamPlay coordination language describes and generates adaptation targets. The PikeOS hypervisor is employed to partition the system into isolated and independent partitions that communicate through proprietary tools. It facilitates resource allocation and sharing while enabling the execution of multiple Linux instances within isolated partitions. This configuration allows the creation of replicas and seamless switching between them, facilitating necessary adaptations in the presence of faults or attacks.

6 Conclusion

This paper comprehensively overviews the ADMORPH architecture and its key components. We delve into the mechanisms employed by ADMORPH to facilitate real-time adaptation while upholding safety considerations. While adaptation strategies are generated offline, potentially through design-space exploration, the actual runtime adaptation necessitates careful attention to ensure swift response times within individual CPSoS building blocks. Tasks with strict timing requirements may require internal resilience mechanisms to effectively handle accidental and malicious faults. We identified and described various strategies for adapting to different scenarios, bounding reconfiguration times, and particularly decoupling reconfiguration from the operational behavior of components. These strategies enable efficient and reliable runtime adaptation within the ADMORPH solution. Lastly, we referred to the Use Cases that support the project by briefly describing the challenges and how the ADMORPH solutions can address them.

Acknowledgments

This work was supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 871259 (ADMORPH project).

References

[1] D. Sapra and A. D. Pimentel, "Exploring multi-core systems with lifetime reliability and power consumption trade-offs," in *Embedded Computer Systems: Architectures, Modeling, and Simulation: 23rd International Conference, SAMOS 2023*, Springer, 2023.

[2] A. D. Pimentel, "Exploring exploration: A tutorial introduction to embedded systems design space exploration," *IEEE Design & Test*, vol. 34, no. 1, 2017.

[3] J. Roeder, B. Rouxel, S. Altmeyer, and C. Grelck, "Towards energy-, time- and security-aware multi-core coordination," in *22nd International Conference on Coordination Models and Languages (COORDINATION 2020), Malta* (S. Bliudze and L. Bocchi, eds.), vol. 12134 of *Lecture Notes in Computer Science*, pp. 57–74, Springer, 2020.

[4] B. Rouxel, U. Pagh Schultz, B. Akesson, J. Holst, O. Jorgensen, and C. Grelck, "PReGO: a generative methodology for satisfying real-time requirements on cots-based systems: Definition and experience report," in *19th ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences (GPCE 2020), Chicago, USA*, pp. 70–83, ACM, 2020.

[5] J. Roeder, B. Rouxel, S. Altmeyer, and C. Grelck, "Energy-aware scheduling of multi-version tasks on heterogeneous real-time systems," in *36th ACM/SIGAPP Symposium on Applied Computing (SAC 2021)*, pp. 500–510, ACM, 2020.

[6] J. Roeder, B. Rouxel, and C. Grelck, "Scheduling dags of multi-version multi-phase tasks on heterogeneous real-time systems," in *14th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC 2021), Singapore*, IEEE, 2021.

[7] B. Rouxel, S. Altmeyer, and C. Grelck, "YASMIN: a Real-time Middleware for COTS Heterogeneous Platforms," in *22nd ACM/IFIP International Middleware Conference (MIDDLEWARE 2021)*, ACM, 2021.

[8] L. Miedema and C. Grelck, "Strategy Switching: Smart Fault-tolerance for Weakly-hard Resource-constrained Real-time Applications," in *Software Engineering and Formal Methods, 20th International Conference, SEFM 2022*, pp. 129–145, Springer, LNCS 13550, 2022.

[9] M. Maggio, A. Hamann, E. Mayer-John, and D. Ziegenbein, "Control system stability under consecutive deadline misses constraints," in *ECRTS, Euromicro Conference on Real-Time Systems*, 2020.

[10] N. Vreman, A. Cervin, and M. Maggio, "Stability and performance analysis of control systems subject to bursts of deadline misses," in *ECRTS, Euromicro Conference on Real-Time Systems*, 2021.

[11] G. Bernat, A. Burns, and A. Liamosi, "Weakly hard real-time systems," *IEEE Transactions on Computers*, vol. 50, no. 4, pp. 308–321, 2001.

[12] N. Vreman, R. Pates, and M. Maggio, "WeaklyHard.jl: Scalable analysis of weakly-hard constraints," in *RTAS, IEEE Real-Time and Embedded Technology and Applications Symposium*, 2022.

[13] N. Vreman, P. Pazzaglia, V. Magron, J. Wang, and M. Maggio, "Stability of linear systems under extended weakly-hard constraints," *IEEE Control Systems Letters*, vol. 6, pp. 2900–2905, 2022.

[14] B. J. Josephrexon and M. Maggio, "Experimenting with networked control software subject to faults," in *2022 IEEE 61st Conference on Decision and Control (CDC)*, pp. 1547–1552, 2022.

[15] P. Pazzaglia and M. Maggio, "Characterizing the effect of deadline misses on time-triggered task chains," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 11, pp. 3957–3968, 2022.

[16] P. Pazzaglia, A. Hamann, D. Ziegenbein, and M. Maggio, "Adaptive design of real-time control systems subject to sporadic overruns," in *DATE, Design, Automation and Test in Europe Conference*, 2021.

[17] S. Tverdyshev, H. Blasum, B. Langenstein, J. Maebe, B. De Sutter, B. Leconte, B. Triquet, K. Müller, M. Paulitsch, A. Söding-Freiherr von Blomberg, and A. Tillequin, "MILS Architecture," Sept. 2013.

[18] M. Kadar, *Integration Methods for Host Intrusion Detection into Embedded Mixed-Criticality Systems*. PhD thesis, TU Kaiserslautern, 2022.

[19] D. Kuzhiyelil, P. Zieris, M. Kadar, S. Tverdyshev, and G. Fohler, "Towards Transparent Control-Flow Integrity in Safety-Critical Systems," in *Information Security*, Lecture Notes in Computer Science, (Cham), pp. 290–311, Springer International Publishing, 2020.