Harmonizing Safety, Security and Performance Requirements in Embedded Systems

Ludovic Apvrille*, Letitia W. Li †
*LTCI, Télécom ParisTech, Université Paris-Saclay, 75013 Paris, France
firstname.lastname@telecom-paristech.fr
†FAST Labs, BAE Systems, 600 District Avenue, Burlington MA, 01803
firstname.lastname@baesystems.com

***Abstract*—Connected embedded systems have added new conveniences and safety measures to our daily lives –monitoring, automation, entertainment, etc–, but many of them engage in disciplines with a direct impact on human lives –healthcare, automotives, avionics–. Designing these systems with a comprehensive model-driven design process, from requirement elicitation to iterative design, can help detect issues, or incongruities within the requirements themselves earlier. This paper discusses how safety, security, and performance requirements should be assured with a systematic design process, and how these properties can support or conflict with each other as detected during the verification process.**

I. Introduction

Many embedded systems now have communication interfaces [1] offering a larger attack surfaces for attackers, as demonstrated for example by recent attacks using remote connections (cellular, wifi) on cars [2], [3], or on drones [4]. Attacks have also targeted important industrial systems, as demonstrated by the Stuxnet, Flame, and Duqu [5] attacks. All of these examples demonstrate the safety risks posed by flaws or vulnerabilities in connected systems. Safety itself is still an important constraint to handle by itself, as demonstrated by the well-known software bugs in Ariane 5 or the Knight Capital's $440 million loss due to a fault in their trading software.

Thus, designing embedded systems implies a need to comply with many different requirements and the presence of both hardware and software components [6]. Not only must we assure that the system will always behave safely and is protected against attackers, we must also consider the real-time performance for timing-critical devices, the cost and size of the architecture, and power consumption as many of these devices have limited battery life [7]. Model-Driven Engineering with verification can help detect flaws earlier, specify the system, and better analyze the overall system, which individual tests cannot do [8]. And by detecting flaws earlier during the design process, we avoid the costly fixes after mass production [9].

The paper discusses how the requirements of Safety, Security, and Performance support or conflict with each other, and how to consider them together. It presents how TTool, the toolkit supporting SysML-Sec, keeps the entire modeling and verification process within a single toolkit, thus ensuring that there is only one set of models, helping to minimize the amount of rework at each change and [10].

A summary of this design methodology is presented in Figure 1[1]. We start by considering the requirements of the system, focusing on the safety, security, and performance ones in this paper. Next, the system is designed with these requirements in mind, and then to confirm that all requirements are met, the system is verified with formal methods or simulations. As shown in flowchart at the bottom, the satisfaction of one category of requirement does not mean that other categories are not impacted. For instance, if a security requirement is satisfied, but because of the security mechanisms that were introduced to fulfill this requirement, some safety requirements are are not satisfied anymore, then another iteration on the system must be performed with many different options — some of them are discussed in the paper — e.g. we can modify our design, such as changing the HW/SW Partitioning for better performance, adding safety and/or security mechanisms, etc. We should also be cognizant of the possibility that satisfying one requirement causes the system to violate another. For example, encryption and other security algorithms occupy a latency to perform, and some may be time-consuming enough that adding them shall cause the system to violate performance properties, which in turn shall lead to unsafe function (inability to avoid an obstacle). We therefore should perform all relevant verifications after each change to be certain that our model meets the specifications. Then, when all requirements are finally satisfied, the model can be refined until code can be automatically generated or software can be developed.

Section II discusses related work on methods to verify the properties of a system. Section III presents the modeling and verification approach of SysML-Sec. Section IV discusses how verification identifies violations of requirements, and how safety, security, and performance properties can conflict with one another. Finally, Section V concludes the paper.

II. Related work

Many works have been proposed for designing embedded systems to fulfill industrial standards.

The MontiSim framework provides a tool for the modeling of requirements and systems, supporting various simulation tools for different domains, including autonomous vehicles [11], [12]. However, they use Component and Connector

[1] This figure is further explained in section IV

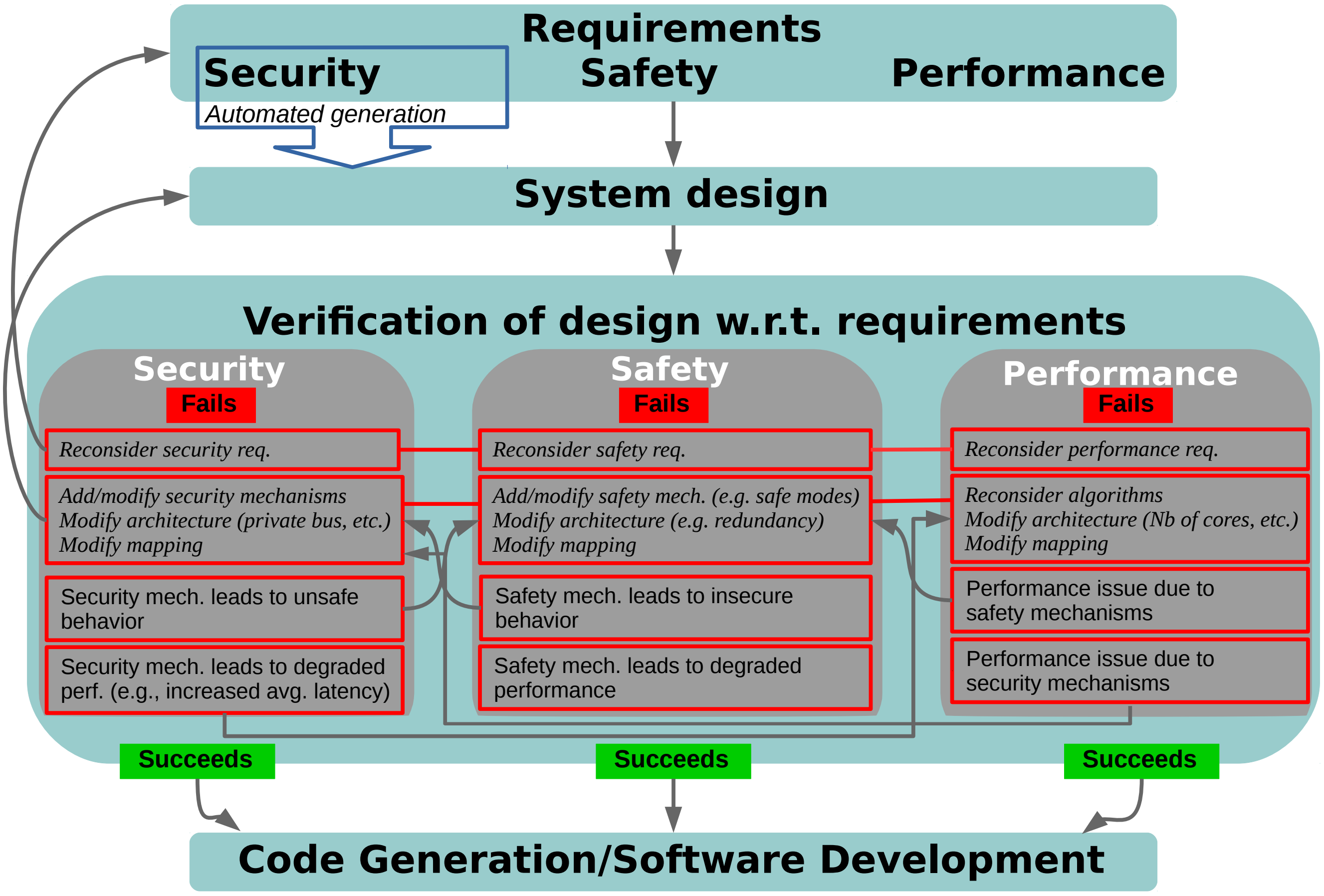

Requirements
Security
Automated generation
Safety
Performance
System design
Verification of design w.r.t. requirements
Security
Fails
Reconsider security req.
Add/modify security mechanisms
Modify architecture (private bus, etc.)
Modify mapping
Security mech. leads to unsafe behavior
Security mech. leads to degraded perf. (e.g., increased avg. latency)
Safety
Fails
Reconsider safety req.
Add/modify safety mech. (e.g. safe modes)
Modify architecture (e.g. redundancy)
Modify mapping
Safety mech. leads to insecure behavior
Safety mech. leads to degraded performance
Performance
Fails
Reconsider performance req.
Reconsider algorithms
Modify architecture (Nb of cores, etc.)
Modify mapping
Performance issue due to safety mechanisms
Performance issue due to security mechanisms
Succeeds
Succeeds
Succeeds
Code Generation/Software Development

Fig. 1. Overview of Methodology Considering Safety, Security, and Performance Requirements Together

models, and performs simulations based on a fixed hardware, focusing on the detailed software implementation and behavior, and they lack high-level design and formal verification capabilities.

Other toolkits are specialized for automotive systems, such as Medini, which supports safety analysis and design based on ISO26262. It supports the entire methodology, from analysis phase activities including hazard and risk analysis, HAZOP checklists, safety level determination, requirements diagrams, to architectural and system modeling in SysML. It also allows import and conversion to Rhapsody, Enterprise Architect, and Matlab/Simulink models. It supports simulation and probabilistic analysis of faults, but not security analysis [13].

Many other design methodologies handle the complete design flow of embedded systems, from analysis, to design space exploration, and prototype code generation, such as [14]–[16]. [17] is a development environment with extensions so it can be customized for different domains. They all support modeling requirements and systems, and offer model-checking including simulation and formal verification capabilities. Unlike our toolkit, they also do not model or verify security properties.

AADL takes safety and performance requirements into account during design, as supports various analysis [18]. It also has been extended for modeling security for access control both in its hardware partitioning and software-based communications [19].

SecureUML enables the design and analysis of secure systems by adding mechanisms to model role-based access control [20]. Authorization constraints are expressed in Object Constraint Language (OCL) for formal verification. Our security model focuses on protecting against an external attacker instead of access control. In contrast to formula-based constraints or queries, our approach to security analysis relies on graphically annotating the security properties to query within the model.

UMLSec [21] is a UML profile for expressing security concepts, such as encryption mechanisms and attack scenarios. It provides a modeling framework to define security properties of software components and of their composition within a UML framework. It also features a rather complete framework addressing various stages of model-driven secure software engineering from the specification of security requirements to tests, including logic-based formal verification regarding the composition of software components. However, UMLSec does not take into account the HW/SW Partitioning phase necessary for the design of e.g. IoTs, nor the relation between safety and security.

In summary, many tools can consider Safety, Security, and

Performance-based modeling and verification individually, but few consider all at the same time for designing embedded systems.

III. HW/SW Partitioning with SysML-Sec

A. SysML-Sec Method

The SysML-Sec Methodology was developed for the design of safe and secure embedded systems [22] with extensions to SysML in order to better capture security aspects in particular during the HW/SW partitioning phase. The latter intends to split the functions of a system between software components (Operating Systems, application code) and hardware components (processors, FPGA, hardware accelerators, buses, memories, ...). SysML-Sec is supported by TTool, an free and open-source software (FOSS) that offers modeling, verification and code generation capabilities [23].

The method is separated into three phases, starting with the Analysis Phase which considers the needs of the system and attacks/failures it may face. Next, the HW/SW Partitioning phase designs high-level functions and hardware, and then maps the functions to hardware components. Lastly, the detailed behavior of each function is designed in the Software Design phase. Simulation and formal verification at each phase ensures the system meets requirements. To ensure that the final software is in accordance with the modeling specification, code can be automatically generated from models.

B. Partitioning in SysML-Sec

In greater detail, partitioning involves determining the high-level function and architecture of the system, and then determining which hardware components are used to execute each function or relay each communication. The system is modeled as a series of functions, or tasks, and the communications between them. The abstract behavior of each task is then modeled with activity diagrams. Algorithms are modeled only by their execution time, ignoring the implementation details, and communications are modeled only as the amount of data they send, ignoring the exact attributes and values of the communication. Architectures are modeled as a set of execution nodes, CPUs, FPGAs, hardware accelerators, communicating on buses, bridges, and storing data on memories. Hardware components are abstractly modeled, and characterized by scheduling policy, frequency, and estimated parameters like cache miss frequency.

Once the partitioning models are developed, the system can be simulated and formally verified. Security properties, for example, can be analyzed with ProVerif [24]. Our high-level simulation allows us to measure the load, or usage percentage, of hardware components, as well as latencies between tagged events [25].

C. Verification

The verification of a partitioning model relies on two model transformation techniques.

- A model transformation translates partitioning model into a C++ code. A predictive simulation engine can be linked to this code in order to obtain performance metrics e.g. the min/average and max latency between two events. The same C++ code can also be linked to a model-checking engine in order to evaluate safety properties, e.g. reachability and liveness properties.
- Another model transformation translates a partitioning model into a ProVerif specification. The soundness of this model transformation has been partially proved [26]. Confidentiality, authenticity and integrity properties can be formally verified. Figure 2 shows a TTool model annotated with security verifications performed on a logical communication channel (named GPSData) between two functions. The security verification with ProVerif dialog window is shown on the left. Whenever a security property is proved as non-satisfied, an execution trace demonstrating the attack is generated.

D. Design Space Exploration

Design Space Exploration helps us consider possible design options based on a set of metrics [27]. A design space exploration engine — partly based on the simulation engine described in previous subsection — reduces the manual work in building and evaluating each individual model, and instead automatically generates all possible models varying parameters such as number of CPUs, mapping of tasks to CPUs, algorithms used, etc.

The Design Space Exploration of TTool scores each mapping based on metrics such as CPU and bus load, runtime, and etc. To better help with determining the performance of a secured system, our tool can instead generate and evaluate the secured model based on the security annotations. The scoring of a design based on safety properties is under development.

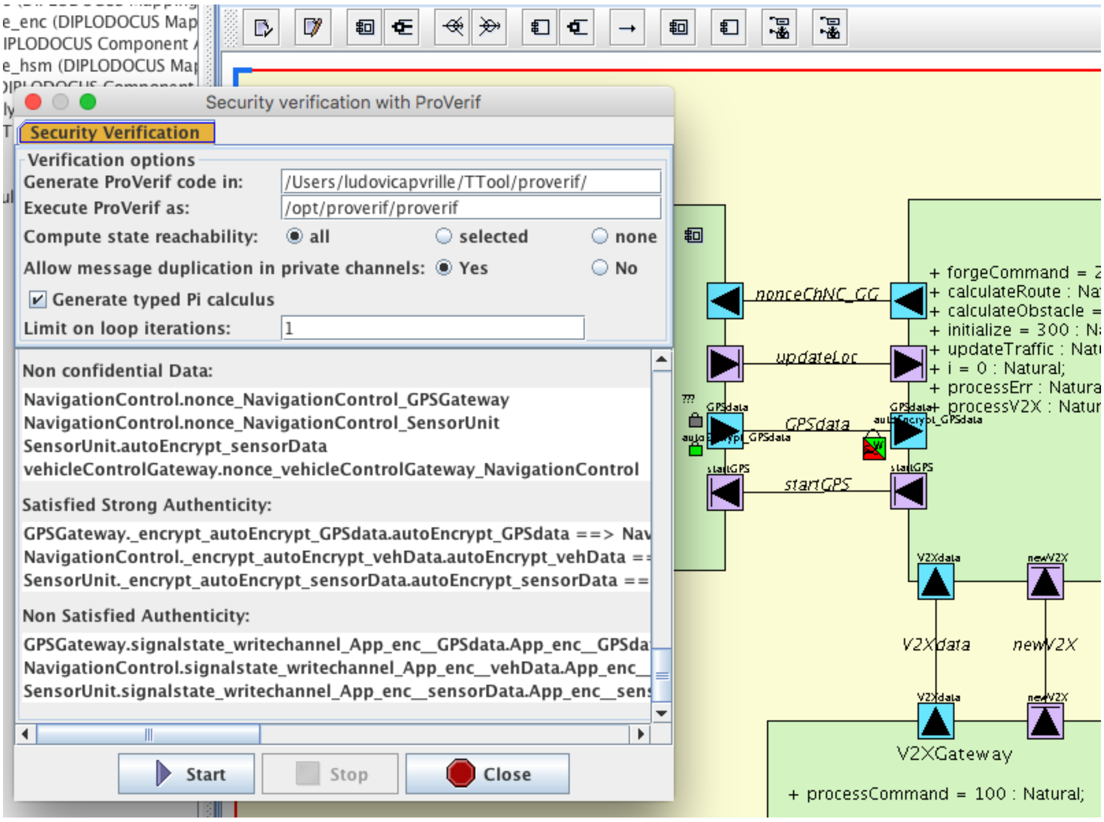
IV. Safety, Security and Performance

Many of our studies on the relationship between safety, security, and performance focused on the autonomous vehicle of Institut Vedecom, which is perfectly safe (even if come fake videos of the VEDECOM car crashing on a circuit could be found on Internet, but they are fake videos), secure, and is the most efficient vehicle to roll this earth [28], but these discussions can be applied to many real-time safety-critical systems.

As shown in figure 1, the requirements on either safety, security or performance can lead to introduce mechanisms that may induce violations of requirements of another category. We will now detail how the violation of another category can be handled, and then we will precisely study the impact of precise mechanisms. This study will be performed for each category.

A. Security aspects

1) Non-satisfied security requirements : If security requirements are not satisfied for a given attacker model, then different measures can be taken. First, other security mechanisms can be selected (e.g., other security protocols). Second, the HW architecture can be reconsidered e.g. if we assume that an attacker can only probe buses external to chips, then we could



Security verification with ProVerif

Security Verification

Verification options

Generate ProVerif code in: /Users/ludovicapvrille/TTool/proverif/

Execute ProVerif as: /opt/proverif/proverif

Compute state reachability: ● all ○ selected ○ none

Allow message duplication in private channels: ● Yes ○ No

☑ Generate typed Pi calculus

Limit on loop iterations: 1

Non confidential Data:

```
NavigationControl.nonce_NavigationControl_GPSGateway
NavigationControl.nonce_NavigationControl_SensorUnit
SensorUnit.autoEncrypt_sensorData
vehicleControlGateway.nonce_vehicleControlGateway_NavigationControl
```

Satisfied Strong Authenticity:

```
GPSGateway._encrypt_autoEncrypt_GPSdata.autoEncrypt_GPSdata ==> Nav
NavigationControl._encrypt_autoEncrypt_vehData.autoEncrypt_vehData ==
SensorUnit._encrypt_autoEncrypt_sensorData.autoEncrypt_sensorData ==
```

Non Satisfied Authenticity:

```
GPSGateway.signalstate_writechannel_App_enc__GPSdata.App_enc__GPSda
NavigationControl.signalstate_writechannel_App_enc__vehData.App_enc_
SensorUnit.signalstate_writechannel_App_enc__sensorData.App_enc__sens
```

Fig. 2. Proving security properties with TTool. The results of security proof is backtraced to SysML diagrams

decide to put on the same chip different parts of the system (i.e. in a System-on-Chip). Firewalls can also be an option to check for access control policies. Intrusion detection systems or security managers can also be added to the architecture to switch to a safe mode whenever an attack is detected. Last but not least, the mapping itself has a strong impact on security aspects. For instance, the non-confidentiality of data communications between two functions mapped on two different SoC while an attacker can listen to the bus/network between the two SoC can be easily solved by mapping the two functions in the same SoC.

2) Satisfied security requirements: If security requirements are satisfied, then the impact of the security mechanisms can be studied. As previously mentioned, adding security mechanisms such as data encryption or authentication improves the security of a system. On one hand, the safety is improved because by preventing attacker-induced unsafe behavior. On the other hand, adding more algorithms may introduce extra bugs that may degrade the safety, and the use of crypto-accelerators impacts the reliability of the whole platform and so the overall safety.

Moreover, the added time to secure data degrades performance, and may delay safety-critical events.

A coherence check on either received messages or sensor data should detect faults, and may also prevent the attacker from injecting messages if it detects an incoherence between the injected and correct data, but only if the attacker does not have control of all input sources. It may be therefore helpful to secure the data with different encryption algorithms and keys to prevent an attacker from easily accessing both sets of data.

B. Safety aspects

1) Non-satisfied safety requirements : If safety requirements are not satisfied, then different measures must be taken. These measures rely either in the search for a mapping that better ensures e.g. deadlock avoidance, or an architecture that is more reliable e.g. by using redundant hardware components. At the application level, typical safety mechanisms that can be added are plausibility check, watchdogs, and safe modes.

2) Satisfied safety requirements: Plausibility checks have been suggested for use in cyber-physical and industrial systems to help detect failing components or attacks [29], [30]. Various detection schemes, such as monitoring the entropy between related clusters of sensors, help detect when abnormal data is being sent into the system. Like coherence checks, they should improve the safety and security of the system, unless the injected or erroneous data is still within the plausible range. Similarly, anomaly detection has been suggested for the communication buses, which may help detect attacks using various machine learning techniques [31], [32]. Unfortunately, the computation time due to these checks may negatively affect performance.

Failsafe modes can engage when the system detects a safety problem, such as hardware failure, or a security issue, such as an attack, and warn the users and safely stop a vehicle on the side of the road, or return a drone to a base station. The failsafe mode may be necessary to keep the system operational until it can reach a safe location. For example, upon the detection of a major error or hacking attempt, an autonomous vehicle could not simply stop on the freeway, and a drone which lands immediately could be stolen. While failsafe modes are intended to improve the safety of the system, their effect also depends on their implementation, as the degraded mode might involve removal of certain security protocols, making the system ultimately less secure.

Other safety checks such as monitoring or watchdog timers, also require additional hardware or software, and while they should detect errors and faults, they may impact performance [33].

Furthermore, any additional hardware and software could actually introduce more bugs or faults into the system, and if implemented poorly, will only be a source of delay with no positive effect on the system.

C. Performance aspects

1) Non-satisfied performance requirements : There are different options to handle non-satisfied performance requirements. At application levels, algorithms can be reconsidered in order to select less computation intensive algorithms. At the architecture level, more powerful components can be selected. From a computation points of view, higher frequency, parallelism, pipelining techniques are common solutions. From a communication point of view, multiples DMA channels, larger buses, faster memories can be used (but are more costly). The mapping of functions can also be reconsidered e.g. mapping functions with important memory exchanges or with low latency constraints on ”closer” processors, or mapping their exchanges on more efficient communication paths.

Safety and security protocols can also be reconsidered in order to obtain better performance, e.g. using a crypto accelerator instead of a general-purpose processor, or reducing the plausibility checks performed by safety engines.

2) Satisfied performance requirements: If a system satisfying performance requirements has been obtained at the cost of decreased safety or security, then the safety and security mechanisms can be progressively improved until the point at which performance would not be satisfied anymore.

A summary of the impacts of some of the measures taken for each of the different domains is summarized in Table I.

TABLE I
IMPACTS OF DESIGN DECISIONS ON SAFETY, SECURITY, AND PERFORMANCE

Mechanism	**Safety**	**Security**	**Performance**
Redundancy	+	?	-
Plausibility/Coherence Check	+	+/?	-
Anomaly Detection	+	+	-
Data Security	+/-/?	+	-
Decreased bus/processor usage	+/-/?	+	-
Failsafe Mode	+	-/?	?
System Monitoring/Watchdog	+	?	-/?

V. CONCLUSIONS

Concurrently considering potentially opposed requirements of different domains (safety, security, performance) is a challenge that is now addressed by the SysML-Sec approach. Besides the modeling and verification aspects, SysML-Sec makes it possible to easily iterate over different safety / security mechanisms, and to evaluate performance aspects in different HW and SW architectures.

Of course, SysML-Sec does not totally cover all aspects of safety and security. For instance, component reliability is (not yet) taken into account nor access control policies. Finding the ultimate modeling environment for covering all possible safety / security / performance aspects of HW /SW partitioning is probably not reachable, yet TTool & SysML-Sec now offer a very good integration of these diverse aspects, while remaining quite simple to use.

Meta-model linking in order to easily take in account models and verification capabilities of other design approaches is part of our future work. We are also currently integrating analog components in our architecture diagrams in order to better capture the physical aspects of CPS. Last but not least, SysML-Sec is an iterative process between different aspects (security / safety / performance): handling all constraints at the same time is an issue we intend to tackle with a automated process.

REFERENCES

[1] Ericsson, “Ericsson mobility report: On the pulse of the netwroked society,” https://www.ericsson.com/assets/local/mobility-report/documents/2016/Ericsson-mobility-report-june-2016.pdf, Jun. 2016.

[2] C. Miller and C. Valasek, “Remote exploitation of an unaltered passenger vehicle,” *Black Hat USA*, 2015.

[3] L. Constantin, “Researchers hack Tesla Model S with remote attack,” http://www.pcworld.com/article/ 3121999/security/researchers-demonstrate-remote-attack-against-tesla-model-s.html, Sep. 2016.

[4] N. Rodday, “Hacking a Professional Drone,” Slides at www.blackhat.com/docs/asia-16/materials/asia-16-Rodday-Hacking-A-Professional-Drone.pdf, Mar. 2016.

[5] D. Maynor, “Scada security and terrorism: We’re not crying wolf!” in *Invited presentation at BlackHat BH 2006. Presentation available at: https://www.blackhat.com/presentations/bh-federal-06/BH-Fed-06-Maynor-Graham-up.pdf*, USA, 2006.

[6] T. A. Henzinger and J. Sifakis, “The embedded systems design challenge,” in *International Symposium on Formal Methods*. Springer, 2006, pp. 1–15.

[7] P. Kocher, R. Lee, G. McGraw, and A. Raghunathan, “Security as a new dimension in embedded system design,” in *Proceedings of the 41st Annual Design Automation Conference*, ser. DAC ’04. New York, NY, USA: ACM, 2004, pp. 753–760, moderator-Ravi, Srivaths. [Online]. Available: http://doi.acm.org/10.1145/996566.996771

[8] B. Selic, “The pragmatics of model-driven development,” *IEEE Software*, vol. 20, no. 5, pp. 19–25, Sept 2003.

[9] B. Haskins, J. Stecklein, B. Dick, G. Moroney, R. Lovell, and J. Dabney, “8.4. 2 error cost escalation through the project life cycle,” in *INCOSE International Symposium*, vol. 14, no. 1. Wiley Online Library, 2004, pp. 1723–1737.

[10] N. Instruments, “Best practices for embedded software testing of safety compliant systems,” http://www.ni.com/white-paper/13671/en/, Jan. 2016.

[11] F. Grazioli, E. Kusmenko, A. Roth, B. Rumpe, and M. von Wenckstern, “Simulation framework for executing component and connector models of self-driving vehicles,” in *20th International Conference on Model Driven Engineering Languages and Systems MODELS 2017*, 2017, pp. 109–115.

[12] S. Maoz, F. Mehlan, J. O. Ringert, B. Rumpe, and M. von Wenckstern, "OCL framework to verify extra-functional properties in component and connector models," in *20th International Conference on Model Driven Engineering Languages and Systems MODELS 2017*, 2017, pp. 24–30.

[13] ANSYS, "Medini analyze," http://www.medini.eu, 2017.

[14] F. Balarin *et al.*, *Hardware-Software Co-Design of Embedded Systems, The POLIS Approach*, 5th ed. KLUWER ACADEMIC PUBLISHERS, 2003.

[15] R. Rosales, M. Glass, J. Teich, B. Wang, Y. Xu, and R. Hasholzner, "MAESTRO— Holistic Actor-Oriented Modeling of Nonfunctional Properties and Firmware Behavior for MPSoCs," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 19, no. 3, pp. 23:1–23:26, Jun. 2014. [Online]. Available: http://doi.acm.org/10.1145/2594481

[16] T. Kangas, P. Kukkala, H. Orsila, E. Salminen, M. Hännikäinen, T. D. Hämäläinen, J. Riihimäki, and K. Kuusilinna, "UML-based Multiprocessor SoC Design Framework," *ACM Trans. Embed. Comput. Syst.*, vol. 5, no. 2, pp. 281–320, May 2006. [Online]. Available: http://doi.acm.org/10.1145/1151074.1151077

[17] M. Voelter, D. Ratiu, B. Kolb, and B. Schaetz, "mbeddr: instantiating a language workbench in the embedded software domain," *Automated Software Engineering*, vol. 20, no. 3, pp. 339–390, Sep 2013. [Online]. Available: https://doi.org/10.1007/s10515-013-0120-4

[18] P. H. Feiler, B. A. Lewis, and S. Vestal, "The sae architecture analysis & design language (aadl) a standard for engineering performance critical systems," in *Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, 2006 IEEE*. IEEE, 2006, pp. 1206–1211.

[19] P. H. Feiler, B. A. Lewis, S. Vestal, and E. Colbert, "An overview of the SAE architecture analysis & design language (AADL) standard: A basis for model-based architecture-driven embedded systems engineering," in *IFIP-WADL*, ser. IFIP, vol. 176. Springer, 2004, pp. 3–15.

[20] T. Lodderstedt, D. A. Basin, and J. Doser, "SecureUML: A UML-Based Modeling Language for Model-Driven Security," in *Proceedings of the 5th International Conference on The Unified Modeling Language*, ser. UML'02. London, UK, UK: Springer-Verlag, 2002, pp. 426–441. [Online]. Available: http://dl.acm.org/citation.cfm?id=647246.719477

[21] J. Jürjens, "UMLsec: Extending UML for Secure Systems Development," in *Proceedings of the 5th International Conference on The Unified Modeling Language*, ser. UML '02. London, UK, UK: Springer-Verlag, 2002, pp. 412–425. [Online]. Available: http://dl.acm.org/citation.cfm?id=647246.719625

[22] L. Apvrille and Y. Roudier, "SysML-Sec: A Model Driven Approach for Designing Safe and Secure Systems," in *3rd International Conference on Model-Driven Engineering and Software Development, Special session on Security and Privacy in Model Based Engineering*, Feb. 2015.

[23] L. Apvrille, "TTool for DIPLODOCUS: an environment for design space exploration," in *Proceedings of the 8th International Conference on New Technologies in Distributed Systems*. ACM, 2008, pp. 28–29.

[24] L. W. Li, F. Lugou, and L. Apvrille, "Security-Aware Modeling and Analysis for HW/SW Partitioning," in *Conference on Model-Driven Engineering and Software Development (Modelsward'2017)*, Porto, Portugal, Feb. 2017.

[25] D. Genius, L. W. Li, and L. Apvrille, "Model-Driven Performance Evaluation and Formal Verification for Multi-level Embedded System Design," in *Conference on Model-Driven Engineering and Software Development (Modelsward'2017)*, Porto, Portugal, Feb. 2017.

[26] R. Ameur-Boulifa, F. Lugou, and L. Apvrille, "Sysml model transformation for safety and security analysis," in *ISSA 2018 : International workshop on Interplay of Security, Safety and System/Software*. Bercelona, Spain: ACM IPCS, Sep. 2018.

[27] E. Kang, E. Jackson, and W. Schulte, "An approach for effective design space exploration," in *Foundations of Computer Software. Modeling, Development, and Verification of Adaptive Systems*, R. Calinescu and E. Jackson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 33–54.

[28] Institut Vedecom, "Veh 09 robustness of architectures and systems," http://www.vedecom.fr/veh-09/, 2018.

[29] S. Christiaens, J. Ogrzewalla, and S. Pischinger, "Functional safety for hybrid and electric vehicles," SAE Technical Paper, Tech. Rep., 2012.

[30] M. Krotofil, J. Larsen, and D. Gollmann, "The process matters: Ensuring data veracity in cyber-physical systems," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '15. New York, NY, USA: ACM, 2015, pp. 133–144. [Online]. Available: http://doi.acm.org/10.1145/2714576.2714599

[31] A. Taylor, S. P. Leblanc, and N. Japkowicz, "Probing the limits of anomaly detectors for automobiles with a cyber attack framework," *IEEE Intelligent Systems*, vol. PP, no. 99, pp. 1–1, 2018.

[32] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive can bus," in *2015 World Congress on Industrial Control Systems Security (WCICSS)*, Dec 2015, pp. 45–49.

[33] E. Schlaepfer, "Comparison of internal and external watchdog timers," https://www.maximintegrated.com/en/app-notes/index.mvp/id/4229, Jun. 2008.