Towards Trustworthy Aerospace Systems: An Experience Report

Joost-Pieter Katoen

RWTH Aachen University, Software Modeling and Verification Group, Germany

Introduction

Building modern aerospace systems is highly demanding. They should be extremely dependable. They must offer service without interruption (i.e., without failure) for a very long time — typically years or decades. Whereas "five nines" dependability, i.e., a 99.999 % availability, is satisfactory for most safety-critical systems, for on-board systems it is not. Faults are costly and may severly damage reputations. Dramatic examples are known. Fatal defects in the control software of the Ariane-5 rocket and the Mars Pathfinder have led to headlines in newspapers all over the world. Rigorous design support and analysis techniques are called for. Bugs must be found as early as possible in the design process while performance and reliability guarantees need to be checked whenever possible. The effect of fault diagnosis, isolation and recovery must be quantifiable.

Tailored effective techniques exist for specific system-level aspects. Peer reviewing and extensive testing find most of the software bugs, performance is checked using queueing networks or simulation, and hardware safety levels are analysed using an profiled Failure Modes and Effects Analysis (FMEA) approach. Fine. But how is the consistency between the analysis results ensured? What is the relevance of a zero- bug confirmation if its analysis is based on a system view that ignores critical performance bottlenecks? There is a clear need for an integrated, coherent approach! This is easier said than done: the inherent heterogeneous character of on-board systems involving software, sensors, actuators, hydraulics, electrical components, etc., each with its own specific development approach, severely complicates this.

Modeling using an AADL dialect

About three years ago we took up this grand challenge. Within the ESA-funded COMPASS (COrrectness, Modeling and Performance of Aerospace SyStems) project, an overarching model-based approach has been developed. The key is to model on-board systems at an adequate level of abstraction using a general-purpose modeling and specification formalism based on AADL (Architecture Analysis & Design Language) as standardised by SAE International. This enables engineers to use an industry-standard, textual and graphical notation with precise semantics to model system designs, including both hardware as well as software components. Ambiguities about the meaning of designs are abandoned. System aspects that can be modeled are, amongst others,

- (timed) hardware operations, specified on the level of processors, buses, etc.,
- software operations, supporting concepts such as processes and threads,
- hybrid aspects, i.e., continuous, real-valued variables with (linear) time-dependent dynamics, and
- faults with probabilistic failure rates and their propagation between components.

A complete system specification describes three parts: (1) nominal behavior, (2) error behavior, and (3) a fault injection—how does the error behavior influence the system's nominal behavior? Systems are described in a component-based manner such that the structure of system models strongly resembles the real system's structure. A detailed description of the language and its formal semantics can be found in [2].

Formal verification

This coherent and multi-disciplinary modeling approach is complemented by a rich palette of analysis techniques. The richness of the AADL dialect gives the power to specify and generate a single system model that can be analysed for multiple qualities: reliability, availability, safety, performance, and their mixture. All analysis outcomes are related to the same system's perspective, thus ensuring compatibility. First and foremost, mathematical techniques are used to enable an early integration of bug hunting in the design process. This reduces the time that is typically spent on a posteriori testing —in on-board systems, more time and effort is spent on verification than on construction!— and allows for early adaptations of the design. The true power of the applied techniques is their almost full automation: once a model and a property (e.g., can a system ever reach a state in which the system cannot progress?) are given, running the analysis is push-button technology. In case the property is violated, diagnostic feedback is provided in terms of a counterexample which is helpful to find the cause of the property refutation. These model-checking techniques [1] are based on a full state space exploration, and detect all kinds of bugs, in particular also those that are due to the intricacies of concurrency: multiple threads acting on shared data structures. This type of bugs are becoming increasingly frequent, as multi-threading grows at a staggering rate.

Requirements

Whereas academic tools rely on properties defined in mathematical logic, a language that is major obstacle for usage by design engineers, COMPASS uses specification patterns [5]. These patterns act as parametrised "templates" to the engineers and thus offer a comprehensible and easy-to-use framework for requirement specification. In order to ensure the quality of requirements, they can be validated independently of the system model. This includes property consistency (i.e., checking that requirements do not exclude each other), and property assertion (i.e., checking whether an assertion is a logical consequence of the requirements).

Safety

Analysing system safety and dependability is supported by key techniques such as (dynamic) fault tree analysis (FTA), (dynamic) Failure Modes and Effects Analysis (FMEA), fault tolerance evaluation, and criticality analysis [4]. System models can include a formal description of both the fault detection and isolation subsystems, and the recovery actions to be taken. Based on these models, tool facilities are provided to analyze the operational effectiveness of the FDIR (Fualt Detection, Isolation and Recovery) measures, and to assess whether the observability of system parameters is sufficient to make failure situations diagnosable.

Toolset

All techniques and the full modeling approach are supported by the COMPASS toolset [3], developed in close cooperation with the Italian research institute Fondazione Bruno Kessler in Trento, and is freely downloadable for all ESA countries from the website compass.informatik.rwth-aachen.de. The tool is graphical, runs under Linux, and has an easy-to-use GUI.

Industrial evaluation

The COMPASS approach and toolset was intensively tested on serious industrial cases by Thales Alenia Space in Cannes (France). These cases include thermal regulation in satellites and satellite mode management with its associated FDIR strategy. It was concluded that the modeling approach based on AADL provides sufficient expressiveness to model all hardware and software subsystems in satellite avionics. The hierarchical structure of specifications and the component-based paradigm enables the reuse of models. Also incremental modeling is very well supported. The RAMS analyses as provided by the toolset were found to be mature enough to be adopted by industry, and the corresponding results allowed the evaluation of design alternatives [6]. Current investigations indicate that the integrated COMPASS approach significantly reduces the time and cost for safety analysis compared to traditional on-board design processes.

Acknowledgement. We thank all co-workers in the COMPASS project for their contributions, in particular Thomas Noll and Viet Yen Nguyen (RWTH Aachen University), Marco Bozzano, Alessandro Cimatti and Marco Roveri (FBK, Trento), Xavier Olivé (Thales) and Yuir Yushstein (ESA). This research is funded by the European Space Agency via several grants.

References

1. C. Baier and J.-P. Katoen. *Principles of Model Checking.* MIT Press, 2008.
2. M. Bozzano, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll, and M. Roveri. Safety, dependability, and performance analysis of extended AADL models. *The Computer Journal*, doi: 10.1093/com, March 2010.

3. M. Bozzano, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll, M. Roveri, and R. Wimmer. A model checker for AADL (tool presentation). In *Proc. of 22nd Int. Conf. on Computer Aided Verification (CAV 2010)*, LNCS. Springer, 2010. To be published.
4. M. Bozzano and A. Villafiorita. *Design and Safety Assessment of Critical Systems*. CRC Press, 2010.
5. L. Grunske. Specification patterns for probabilistic quality properties. In *Int. Conf. on Software Engineering (ICSE)*, pages 31–40. ACM, 2008.
6. Y. Yushstein, M. Bozzano, A. Cimatti, J.-P. Katoen, V. Y. Nguyen, T. Noll, X. Olivé, and M. Roveri. System-software co-engineering: Dependability and safety perspective. In *4th IEEE International Conference on Space Mission Challenges in Information Technology (SMC-IT)*. IEEE CS Press, 2011.