COMPASS 3.0

Marco Bozzano[1(✉)], Harold Bruintjes[2], Alessandro Cimatti[1], Joost-Pieter Katoen[2], Thomas Noll[2], and Stefano Tonetta[1]

[1] Embedded Systems Unit, Fondazione Bruno Kessler, Trento, Italy
bozzano@fbk.eu

[2] Software Modeling and Verification Group, RWTH Aachen University, Aachen, Germany

Abstract. COMPASS (COrrectness, Modeling and Performance of AeroSpace Systems) is an international research effort aiming to ensure system-level correctness, safety, dependability and performability of on-board computer-based aerospace systems. In this paper we present COMPASS 3.0, which brings together the results of various development projects since the original inception of COMPASS. Improvements have been made both to the frontend, supporting an updated modeling language and user interface, as well as to the backend, by adding new functionalities and improving the existing ones. New features include Timed Failure Propagation Graphs, contract-based analysis, hierarchical fault tree generation, probabilistic analysis of non-deterministic models and statistical model checking.

1 Introduction

The COMPASS toolset provides an integrated model-based approach for System-Software Co-Engineering in the aerospace domain. It uses formal verification techniques, notably model checking, and originates from an ESA initiative dating back to 2008 [6]. Over the past eight years, various projects followed which extended the toolset. In a recent effort funded by ESA, the results of this work have been thoroughly consolidated into a single release, which is now available.

COMPASS 3.0 includes features originally included in distinct tool releases that diverged from the original development trunk[1]. The AUTOGEF and FAME projects focused on Fault Detection, Identification, and Recovery (FDIR) requirements modeling and development, and on fault propagation analysis; HASDEL extended formal analysis techniques to deal with the specific needs of launcher systems, with a strong focus on timed aspects of the model; and finally CATSY had the goal of improving the requirements specification process.

This paper presents an overview of the toolset, as well as a description of the enhancements made since its last official release in 2013 (v2.3). For a more detailed description of the (pre-)existing features and capabilities, we refer to [5,6].

[1] See www.compass-toolset.org.

This work has been funded by the European Space Agency (ESA-ESTEC) under contract 4000115870/15/NL/FE/as.

T. Vojnar and L. Zhang (Eds.): TACAS 2019, Part I, LNCS 11427, pp. 379–385, 2019.
https://doi.org/10.1007/978-3-030-17462-0_25

2 Toolset Overview

The COMPASS toolset can be divided into the user facing side (the frontend), and the verification engines used (the backend). The frontend provides a GUI that offers access to all the analysis functions of the toolset, as well as command-line scripts. The backend tools are chosen and invoked by the toolset automatically.

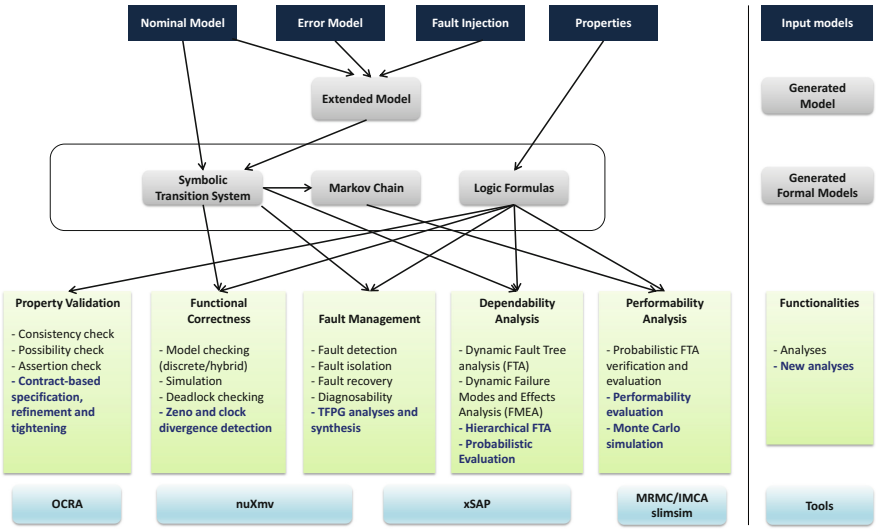

Fig. 1. Overview of the COMPASS toolset

The functionalities of COMPASS are summarized in Fig. 1, where arrows represent I/O relationships for model transformations, and link models with the corresponding analyses. User inputs are models written in a dialect of AADL [18] and properties. COMPASS is based on the concept of *model extension*, i.e., the possibility to automatically inject faults into a nominal model, by specifying error models and a set of fault injections. The extended model is internally converted into a symbolic model amenable to formal verification and to a Markov chain (for probabilistic analyses). Properties are automatically translated into temporal logic formulas. Given the models and the properties, COMPASS provides a full set of functionalities, including property validation, functional correctness, fault management, dependability and performability analyses.

The analyses are supported by the following verification engines: nuXmv [9] for correctness checking; OCRA [10] for contract-based analysis; IMCA [19] and MRMC [20] for performance analysis by probabilistic model checking; slimsim [8] for statistical model checking and xSAP [1] for safety analysis.

COMPASS is licensed under a variant of the GPL license, restricted to ESA member states. It is distributed on a pre-configured and ready-to-use virtual machine, or as two different code bundles.

In a typical workflow, one would start with both a nominal and an error specification of the system and then use simulation to interactively explore its dynamic behavior. In a next step, verification of functional correctness based on user-defined properties can be performed, followed by more specialized analyses as indicated in Fig. 1. For a complete overview of the COMPASS toolset, we refer to the COMPASS tutorial [15] and user manual [16].

3 Input Language

Input models for COMPASS use a variant of the AADL language [18], named SLIM. AADL is standardized and used in e.g., the aerospace and automotive industries. SLIM provides extensions for behavior and error specification. A model is described in terms of components, which may specify subcomponents, forming a component hierarchy. Components interact with each other by means of ports, which send either discrete events or data values. Components may furthermore specify modes, which may render subcomponents active or inactive, thus enabling system reconfiguration. Error behavior is specified in terms of error components, which define error states, (probabilistic) events and propagations which may trigger a synchronization between error components. The impact of faults occurring in the error model onto the nominal model is described by means of *fault injections*, which specify the fault effect by update operations on data components. This has been extended by supporting specifications that can inhibit certain events from occurring, or specifying a set of modes that is available in a certain failure state. The language's semantics and syntax are described in [14].

Language Updates. COMPASS 3.0 supports the property system used by AADL. This makes it possible to annotate various elements in the model by using SLIM specific attributes, and makes the language more compatible with the core AADL specification, improving interoperability. New features also include timed error models (that is, error components may contain clocks), non-blocking ports and separation of configuration and behavior. The latter entails that composite components can only specify modes and cannot change data values or generate events by means of transitions, whereas atomic components may specify states.

Property Specification. Properties can be specified in three different ways. The first two options are simpler to use, since they hide the details of the underlying temporal logic, but less expressive. *Design attributes* [3] represent a specific property of a model's element, such as the delay of an event or the invariant of a mode. They are directly associated to model elements. Formal properties are automatically derived based on the Catalogue of System and Software Properties (CSSP) [3]. The *pattern-based* system uses pre-defined patterns with placeholders to define formal properties. Time bounds and probabilities can optionally be specified. As a last option, the user can encode properties directly using *logical expressions*. This enables the modeler to exploit the full power of the underlying temporal logics, and offers the highest expressivity.

4 New Functionalities in COMPASS 3.0

In this section, we discuss the main functionalities of the COMPASS 3.0 toolset.

Correctness Checking. COMPASS supports checking for correctness of the model by providing properties. The toolset indicates for each property whether it holds or not, and gives a counter example in the latter case. Verification relies on edge technologies based on BDD- and SAT-based model checking, including *K-liveness verification* [12]. In order to assist the user in the specification of timed models, COMPASS 3.0 offers functionality to check the timed correctness of the model w.r.t. *Zenoness* and *clock divergence*. The former is caused by cycles in the system's state space that do not require progressing of time. The latter is caused by clocks that can attain an arbitrarily large value. The toolset can automatically check Zenoness for all modes in the system, and divergence for all clocks.

Contract-Based Analysis. COMPASS 3.0 offers the possibility to perform *contract-based analysis* [11]. Contracts must be specified in the model and attached to components. Each contract consists of an *assumption* (a property of the environment of the component) and a *guarantee* (a property of the implementation of the component, which must hold as long as the assumption holds). In order to perform compositional analysis, a contract refinement must be further specified, which links a contract to a set of contracts of the subcomponents. COMPASS 3.0 supports the following analyses. *Validation* is performed on assumptions and guarantees. The user can choose a subset of these properties and check consistency or entailment. *Refinement checking* verifies whether the contract refinements are correct. Namely, that whenever the implementations of the subcomponents satisfy their contracts and the environment satisfies its assumption, then the guarantee of the supercomponent and the assumptions of its subcomponents are satisfied. Finally, *tightening* looks for a weakening and/or strengthening of the assumptions/guarantees, respectively, such that the refinement still holds.

Fault Trees. COMPASS 3.0 can generate fault trees associated with particular error states in the model. Standard fault trees are flat in nature (being two- or three-leveled), hiding some of the nested dependencies. Contract-based analysis can be used to generate a *hierarchical fault tree*, which captures the hierarchy of the model. This approach makes use of the specified contracts, and checks which events may cause them to be invalidated. COMPASS 3.0 offers further alternatives to analyze fault trees. Static probabilities can be calculated for the entire tree by specifying the probabilities of basic events. Fault Tree Evaluation calculates the probability of failure for a given time span. Finally, Fault Tree Verification checks a probabilistic property specified for the fault tree.

Performability. COMPASS 3.0 offers two model checking approaches to probabilistic analysis (which, using a probabilistic property, determine the

probability of failure within a time period): using numerical analysis or using Monte-Carlo simulation. The former originally only supported Continuous Time Markov Chains (CTMCs) using the MRMC [20] tool. This has now been extended to Interactive Markov Chains (IMCs) using IMCA [19], which makes it possible to analyze continuous-time stochastic models which exhibit *non-determinism*. However, neither approach supports hybrid models containing clocks. For the analysis of these models, statistical model checking techniques [7,8] are employed, which use Monte-Carlo simulation to determine, within a certain margin of likelihood and error, the probability of quantitative properties.

Timed Failure Propagation Graphs. Timed Failure Propagation Graphs (TFPGs) [2] support various aspects of diagnosis and prognosis, such as modeling the temporal dependency between the occurrence of events and their dependence on system modes. A TFPG is a labeled directed graph where nodes represent either fault modes or discrepancies, which are off-nominal conditions that are effects of fault modes. COMPASS 3.0 supports three kinds of analyses based on TFPGs: *synthesis*, where a TFPG is automatically derived from the model, *behavioral validation*, which checks whether a given TFPG is complete (i.e., a faithful abstraction) w.r.t. the model; and *effectiveness validation*, which checks whether the TFPG is sufficiently accurate for allowing diagnosability of failures.

5 Related Work and Conclusion

Closely related to COMPASS is the TASTE toolset, dedicated to the development of embedded, real-time systems. It focuses on the integration of heterogeneous technologies for modeling and verification (including AADL), code generation and integration (e.g., written in C and ADA) and deployment. Another ESA initiative is COrDeT-2, part of which defines OSRA (On-board Software Reference Architecture), which aims to improve software reuse by defining a standardized architecture. Security extensions in COMPASS 3.0 have been added as part of the D-MILS project [21], enabling reasoning on data security.

Various case studies have been performed using COMPASS 3.0. The first one was targeting at the Preliminary Design Review stage of a satellite's design [17]. The study lasted for six months and encompassed a model of about 90 components. A second case study followed during the Critical Design Review stage, focusing on modeling practices and diagnosability [4], with a scale twice the size of [17]. A smaller scale case study was later performed as part of the HASDEL project [8]. Recently, the CubETH nano-satellite was represented as a model with 82 components and analyzed using COMPASS 3.0 [7].

The case studies demonstrate that the key benefit of the COMPASS approach is the culmination of a single comprehensive system model that covers all aspects (discrete, real-time, hybrid, probabilistic). This ensures consistency of the analyses, which is a major benefit upon current practices where various (tailored) models are constructed each covering different aspects. For further directions,

we refer to the COMPASS roadmap [13], which thoroughly discusses goals for the toolset as well as the development process, research directions, community outreach and further integration with other ESA initiatives.

References

1. Bittner, B., et al.: The xSAP safety analysis platform. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 533–539. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_31
2. Bittner, B., Bozzano, M., Cimatti, A.: Automated synthesis of timed failure propagation graphs. In: Proceedings of IJCAI, pp. 972–978 (2016)
3. Bos, V., Bruintjes, H., Tonetta, S.: Catalogue of system and software properties. In: Skavhaug, A., Guiochet, J., Bitsch, F. (eds.) SAFECOMP 2016. LNCS, vol. 9922, pp. 88–101. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45477-1_8
4. Bozzano, M., et al.: Spacecraft early design validation using formal methods. Reliab. Eng. Syst. Saf. **132**, 20–35 (2014)
5. Bozzano, M., Cimatti, A., Katoen, J.P., Nguyen, V.Y., Noll, T., Roveri, M.: Safety, dependability and performance analysis of extended AADL models. Comput. J. **54**(5), 754–775 (2011)
6. Bozzano, M., et al.: A model checker for AADL. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 562–565. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_48
7. Bruintjes, H.: Model-based reliability analysis of aerospace systems. Ph.D. thesis, RWTH Aachen University (2018)
8. Bruintjes, H., Katoen, J.P., Lesens, D.: A statistical approach for timed reachability in AADL models. In: Proceedings of DSN, pp. 81–88. IEEE (2015)
9. Cavada, R., et al.: The nuXmv symbolic model checker. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 334–342. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_22
10. Cimatti, A., Dorigatti, M., Tonetta, S.: OCRA: a tool for checking the refinement of temporal contracts. In: Proceedings of ASE, pp. 702–705 (2013)
11. Cimatti, A., Tonetta, S.: Contracts-refinement proof system for component-based embedded systems. Sci. Comput. Program. **97**, 333–348 (2015)
12. Claessen, K., Sörensson, N.: A liveness checking algorithm that counts. In: Proceedings of FMCAD, pp. 52–59 (2012)
13. COMPASS Consortium: COMPASS roadmap. Technical report (2016). http://www.compass-toolset.org/docs/compass-roadmap.pdf
14. COMPASS Consortium: SLIM 3.0 - syntax and semantics. Technical report (2016). http://www.compass-toolset.org/docs/slim-specification.pdf
15. COMPASS Consortium: COMPASS tutorial - version 3.0.1. Technical report (2018). http://www.compass-toolset.org/docs/compass-tutorial.pdf
16. COMPASS Consortium: COMPASS user manual - version 3.0.1. Technical report (2018). http://www.compass-toolset.org/docs/compass-manual.pdf
17. Esteve, M.A., Katoen, J.P., Nguyen, V.Y., Postma, B., Yushtein, Y.: Formal correctness, safety, dependability and performance analysis of a satellite. In: Proceeding of ICSE, pp. 1022–1031. ACM and IEEE (2012)
18. Feiler, P.H., Gluch, D.P.: Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis & Design Language. Addison-Wesley, Upper Saddle River (2012)

19. Guck, D., Han, T., Katoen, J.-P., Neuhäußer, M.R.: Quantitative timed analysis of interactive Markov Chains. In: Goodloe, A.E., Person, S. (eds.) NFM 2012. LNCS, vol. 7226, pp. 8–23. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28891-3_4
20. Katoen, J.P., Zapreev, I.S., Hahn, E.M., Hermanns, H., Jansen, D.N.: The ins and outs of the probabilistic model checker MRMC. Perform. Eval. **68**(2), 90–104 (2011)
21. van der Pol, K., Noll, T.: Security type checking for MILS-AADL specifications. In: International MILS Workshop. Zenodo (2015). http://mils-workshop-2015.euromils.eu/