COMPASTA: Extending TASTE with formal design and verification functionality*

Alberto Bombardelli, Marco Bozzano, Roberto Cavada, Alessandro Cimatti, Alberto Griggio, Massimo Nazaria, Edoardo Nicolodi, and Stefano Tonetta

Fondazione Bruno Kessler, Via Sommarive 18, 38123 Trento, Italy
{abombardelli,bozzano,cavada,cimatti,griggio,mnazaria,enicolodi,tonettas}@fbk.eu

Abstract. TASTE is a development environment dedicated to embedded, real-time systems, developed under the initiative of the European Space Agency. It consists of various tools, such as graphical editors, code generators and visualizers, which support model-based design of embedded systems, automatic code generation, deployment and simulation. TASTE currently lacks a comprehensive support for performing early verification and assessment of the design models.
The goal of the COMPASTA study is to integrate the formal verification capabilities of COMPASS into TASTE. COMPASS is a tool for model-based System-SW Co-Engineering developed in a series of ESA studies, offering formal design and verification capabilities, such as requirements analysis, contract-based design, functional verification and safety assessment, fault detection and identification analysis. COMPASTA will deliver a full end-to-end coherent tool chain, based on TASTE, covering system design, HW/SW implementation, deployment and testing.

1 Introduction

TASTE [5, 11, 12] is a design and development environment dedicated to embedded, real-time systems, which has been actively developed by the European Space Agency (ESA) since 2008. It consists of various tools such as graphical editors, visualizers and code generators that support the development of embedded systems within a MBSE (Model Based Systems Engineering) approach. TASTE is based on the following technologies and languages: AADL (Architectural Analysis and Design Language) for architectural modeling, ASN.1 for data modelling and SDL (Specification and Description Language) for behavioral specification. TASTE has been adopted as a glue technology and for system deployment in several projects, both in aerospace and in other domains, e.g. [9, 1, 6, 10].

The standard modeling workflow in TASTE includes: the definition of data models using ASN.1; the definition of the functional logical architecture using an Interface View description in AADL; the definition of the behavior of each functional block, e.g. in SDL; the definition of the physical architecture using a Deployment View description in AADL. The physical architecture binds the

* Work funded by ESA/ESTEC under Contract No. 4000133700/21/NL/GLC/kk.

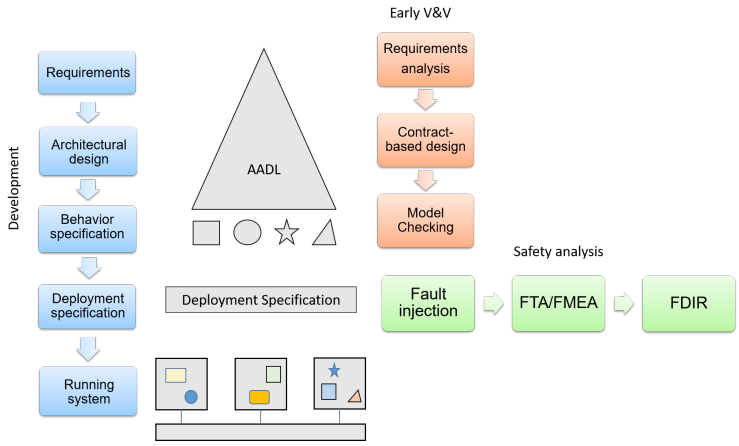

Fig. 1: Functionality of the integrated TASTE+COMPASS toolset.

functional blocks and logical connections to the HW nodes and devices, to enable code generation and building for the target platforms.

COMPASS [2–4] is a tool for System-SW Co-Engineering developed in a series of ESA studies from 2008 to 2016. It is based on a dialect of AADL and provides a full set of formal techniques, based on model checking, such as requirements analysis, fault injection, property verification, safety assessment, fault detection and identification analysis. It is based on the concept of model extension, i.e., the automatic injection of user-specified faults into a nominal model. The extended model is internally converted into a symbolic model amenable to formal verification, whereas properties are translated into temporal logic. COMPASS uses the ocra tool for contract-based design [8], the nuXmv model checker [7] and the xSAP safety analysis platform [13] as back-ends.

The COMPASTA project is an ongoing study funded by the European Space Agency (ESA) that started in April 2021. By integrating the COMPASS functionality into TASTE, COMPASTA will deliver a comprehensive, end-to-end tool chain dedicated to the design and development of embedded systems, covering system design, HW/SW implementation, deployment and testing. This integration will bridge the gap between the architectural level design and the system implementation and deployment, harmonizing the functionality of both tools into a coherent tool chain.

2 The COMPASTA Approach

The integration of COMPASS into TASTE aims to harmonize the models and input languages provided by COMPASS (SLIM, an extension of AADL) with the ones available in TASTE (in particular, AADL and SDL) for the specification of system architecture, component behavior and interaction, system implementation and deployment. Fig. 1 illustrates the functionality of the integrated toolset,

Development phase	**COMPASS functionality**	**TASTE functionality**
Requirements specification	Specification of properties and requirements Requirements analysis	
Architectural design	Contract-based design and refinement Specification of system architecture (AADL)	 Specification of system architecture (AADL)
Behavioral specification	Specification of the behavior of HW components (AADL/SLIM) Formal verification of functional behavior Specification of HW faults Fault injection/ Model extension Formal verification of functional behavior (in presence of faults) Safety and dependability assessment (FTA, FMEA) Fault Detection, Identification and Recovery (FDIR)	Specification of the behavior of SW components (SDL or other languages)
Deployment specification	 Trace validation for testing	Specification of the deployment on the target HW Code generation Testing of the implementation

Table 1: A comparative view of COMPASS and TASTE functionality.

allocated to the different development phases. The COMPASS functionality is complementary with respect to the one available in TASTE. A comparative view is given in Table 1. In particular, COMPASS adds the following capabilities:

- Specify and validate a set of requirements specified in a formal language.
- Use contract-based design to design the system architecture.
- Model the HW components of the system and their functional behavior, using SLIM (an extension of AADL) to specify state machines.
- Model HW faults and automatically inject them into the system model.
- Perform formal verification, safety and dependability assessment.
- Re-execute and validate a trace generate by TASTE testing in the formal model, generating a compatible execution of the HW.

The existing functionality of TASTE, on the other hand, is used to specify the behavior of SW components, their deployment on the target HW, their implementation (via code generation) and to test the final implementation.

The technical objectives of the COMPASTA project include the definition of an extension of the AADL language that is compliant, both syntactically and semantically, with the subset available in TASTE, and enables the specifications and analyses made available by COMPASS (in particular, specification of contracts, properties, faults, and of the behavior of HW components). Semantically, the semantics of the SLIM language from COMPASS needs to be adapted to match the different possibilities available in TASTE (synchronous and asynchronous communication, buffered communication). Finally, COMPASTA requires the design of a translator from AADL/SLIM and SDL input languages into the languages supported by the back-ends, and the integration of the back-ends themselves (ocra, nuXmv and xSAP [8, 7, 13]) into TASTE.

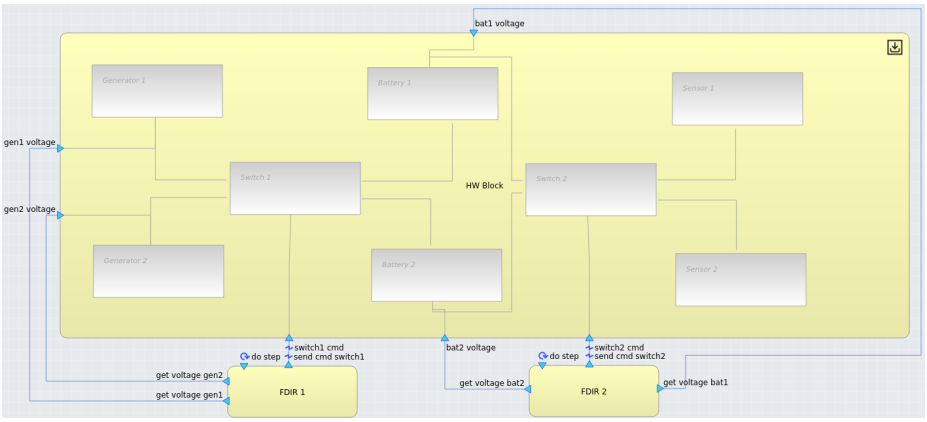

Fig. 2: A power system example.

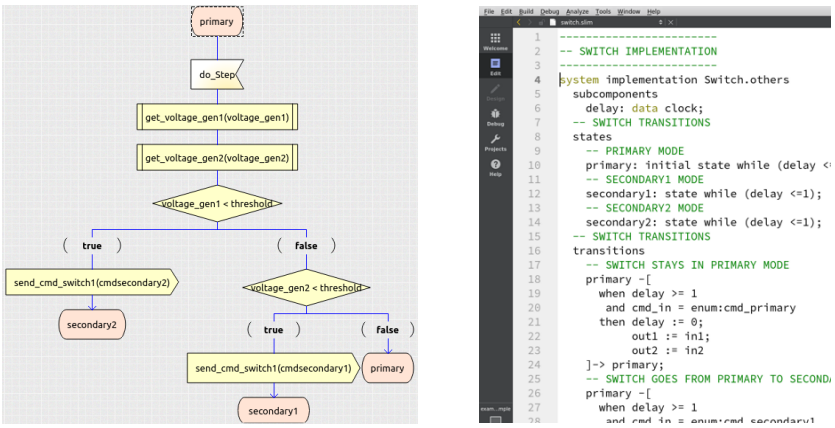

Fig. 3: Sample code in SDL (left) and SLIM (right) for FDIR_1 and Switch_1.

3 An Illustrative Example

We exemplify the COMPASTA workflow using the example in Fig. 2, modeled in the TASTE interface view. It contains both SW (the FDIR components) and HW (batteries, generators, sensors). Two (redundant) generators feed two (redundant) batteries, feeding two sensors. In case of a fault of a generator or battery, two switches can reconfigure the power lines, to exclude the broken item. The desired behavior of the circuit is to guarantee powering of the sensors.

The FDIR components can be modeled in TASTE using SDL. Fig 3 (left) shows an excerpt of the code for FDIR_1. It periodically reads the input voltages of the two generators and, in case one of them is under a given threshold, it sends a command to the Switch component to change from primary mode to a secondary mode. Modeling of the HW requires the COMPASTA extension,

which uses the SLIM language. Fig 3 (right) shows some sample code specifying the behavior of the Switch_1 component. In particular, it models a state machine where transitions correspond to possible reconfigurations (from primary mode to a secondary mode). The input models (in AADL/SLIM and SDL) are then translated into the language supported by the back-ends (SMV). Data ports and connections with different semantics are used to connect HW and SW, i.e. ports with periodic (cyclic) activation for SW, ports with synchronous communication (to model HW data read by SW) and ports with asynchronous (buffered) communication (to model commands sent from SW to HW). COMPASTA defines the semantics of the different forms of communication, the scheduling constraints of SW and HW components, and their encoding into SMV.

Contract-based design can be used to design and validate the system architecture. Contracts (as pairs assumption/guarantee) may be associated to components, e.g., a contract for a battery can have an assumption `always(voltage_in >= 10)` and a guarantee `always(voltage_out >= 10)`. An example of system-level contract is one with an assumption `true` and a guarantee `always(one_valid)` (at least one sensor has a valid output). The system-level contract can be validated against the component-level ones. Moreover, component-level contracts can be checked against an implementation of the respective component. Model checking can be used to verify functional properties, e.g., "Globally, it is always the case that `sensor1.valid` and `sensor2.valid` holds", i.e. the outputs of both sensors are always valid. Fault definitions can be picked from a library, and automatically injected, e.g., a fault injection for the battery is specified as the following record: `[Description => "stuck-at-zero"; Fault_Model => StuckAt; Fault_Dynamics => Permanent; Probability => 1.e-7; Input => voltage_out; Varout => voltage_out; ParameterList => ([Name => Term; Value => "0"])]`, modeling a permanent "stuck-at-zero" fault of the `voltage_out` signal of the battery. The extended model is generated automatically as part of the translation process. xSAP can generate safety artifacts such as Fault Trees and FMEA tables; Fig. 4 show an example fault tree. Once the formal validation of the model has been completed, the TASTE workflow can be used to specify the implementation of the SW components. We briefly sketch this workflow. First, the HW block is replaced by a "HW block I/O" component, which represents the SW layer realizing the communication

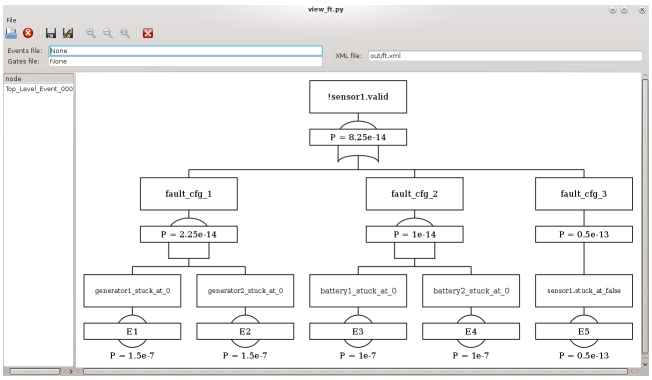

Fig. 4: An example Fault Tree.

between SW and HW in the final implementation. Then, the deployment of the SW components (binding of the SW to the target HW platform(s)) is specified. TASTE can be used to generate the executable code for the target platform(s) and to test the implementation. Finally, COMPASTA offers the possibility to execute a trace, generated by TASTE, on the formal model (including the HW).

4 Conclusions

COMPASTA aims to extend the TASTE toolset with formal verification and assessment functionality, creating a digital continuity from the architectural functional design and system-level safety analysis to the deployment of the embedded software, using the MBSE paradigm. We think that this integration will significantly foster the adoption of the TASTE and COMPASS toolsets. In the intended workflow, system, safety, and software engineers work on the same models in an iterative process supported by various analyses that increase the confidence in the internal and external consistency of the system.

The COMPASTA project started in April 2021 and will be completed by the end of 2022. A first prototype of the integrated tool chain will be released in the first half of 2022, in time for a demonstration at the IMBSA conference.

References

1. ADE: Autonomous Decision Making in Very Long Traverses, `https://www.h2020-ade.eu`
2. Bozzano, M., Bruintjes, H., Cimatti, A., Katoen, J.P., Noll, T., Tonetta, S.: COMPASS 3.0. In: Proc. TACAS 2019 (2019)
3. Bozzano, M., Cimatti, A., Katoen, J.P., Katsaros, P., Mokos, K., Nguyen, V., Noll, T., Postma, B., Roveri, M.: Spacecraft Early Design Validation using Formal Methods. Reliability Engineering & System Safety **132**, 20–35 (2014)
4. Bozzano, M., Cimatti, A., Katoen, J.P., Nguyen, V., Noll, T., Roveri, M.: Safety, Dependability and Performance Analysis of Extended AADL Models. Computer Journal **54**(5), 754–775 (2011)
5. Hugues, J., Pautet, L., Zalila, B., Dissaux, P., Perrotin, M.: Using AADL to build critical real-time systems: Experiments in the IST-ASSERT project. In: Proc. ERTS (2008)
6. MOSAR: Modular Spacecraft Assembly and Reconfiguration, `https://www.h2020-mosar.eu`
7. nuXmv web page (2021), `https://nuxmv.fbk.eu`
8. ocra web page (2021), `https://ocra.fbk.eu`
9. PERASPERA, a PSA activity under the Horizon 2020 Space "COMPET-4-2014: Space Robotics Technologies" Work Programme (Grant Agreement 640026)
10. R. Cavada and A. Cimatti and L. Crema, and M. Roccabruna and S. Tonetta: Model-Based Design of an Energy-System Embedded Controller Using Taste. In: Proc. FM 2016. LNCS, vol. 9995, pp. 741–747 (2016)
11. TASTE web page, `https://taste.tools/`
12. Qualifiable code generation backend for TASTE, ESA Contract No. 4000118510/16/NL/CBi
13. xSAP web page (2021), `https://xsap.fbk.eu`