The COMPASTA Approach for MBSE*

Alberto Bombardelli Marco Bozzano Roberto Cavada Alessandro Cimatti
Alberto Griggio Massimo Nazaria Edoardo Nicolodi
Stefano Tonetta
Fondazione Bruno Kessler
{abombardelli,bozzano,cavada,cimatti,griggio,mnazaria,enicolodi,tonettas}@fbk.eu

Objectives:

O-2 Limitations of current MBSE approaches and ways to circumvent or resolve these (e.g. through customization of processes and tools)

O-4 Leveraging MBSE results to improve the definition and development of other downstream applications and use cases (e.g. simulation, validation and verification, operations)

1 Introduction

The COMPASTA project is an ESA study that started in April 2021. The goal of COMPASTA is to integrate the formal verification functionalities of the COMPASS toolset [2, 3, 4], which are based on model checking, into TASTE [7, 12]. COMPASS [2, 3, 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. COMPASS uses the ocra tool for contract-based design [10], the nuXmv model checker [9] and the xSAP safety analysis platform [13] as back-ends. TASTE [7, 12] is a design and development environment dedicated to embedded, real-time systems, which has been actively developed by 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, automatic code generation, deployment and simulation. TASTE is based on different languages, such as AADL and SDL. TASTE has been adopted as a glue technology and for system deployment in several projects, both in aerospace and in other domains, e.g. [1, 8, 11]. COMPASTA will deliver a full end-to-end coherent tool chain, covering system design, HW/SW implementation, deployment and testing, extending TASTE with a comprehensive support for performing early verification and assessment of the design models.

2 The COMPASTA Approach Exemplified

The integration of COMPASS into TASTE aims to harmonize, both syntactically and semantically, 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. The COMPASS functionality is complementary with respect to the one available in TASTE. In particular, COMPASS adds the possibility to specify and validate a set of requirements,

*This work is funded by ESA/ESTEC under Contract No. 4000133700/21/NL/GLC/kk.

Submitted to:
MBSE 2022

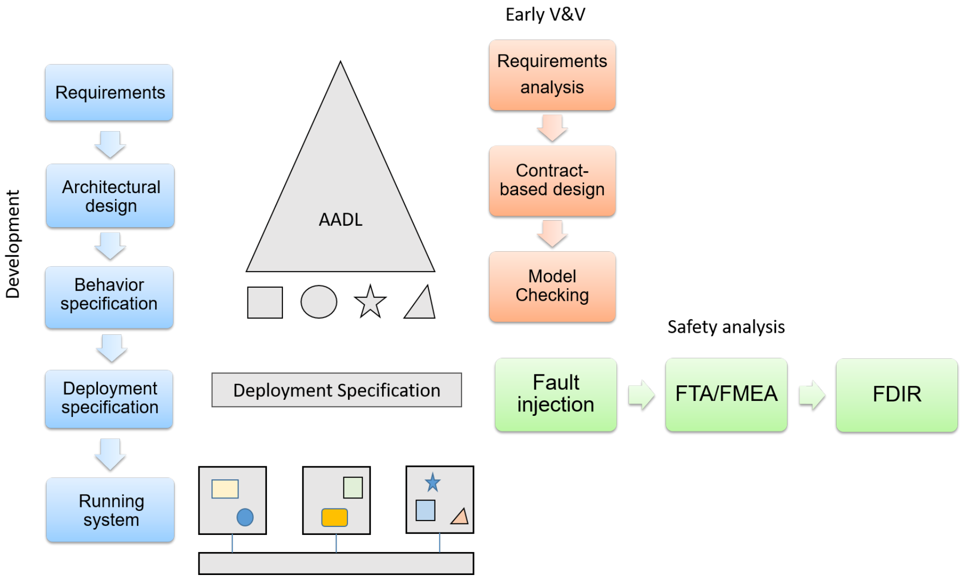


Figure 1: Functionality of the integrated TASTE+COMPASS toolset.

use contract-based design to design the system architecture, model the HW components of the system and their functional behavior, model HW faults, perform formal verification, safety and dependability assessment. TASTE is used to specify the behavior of SW components, their deployment on the target HW, their implementation, and for testing and simulation. We exemplify the COMPASTA workflow using the example in Fig. 2, modeled in TASTE. 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 is to guarantee powering of the sensors. The FDIR components can be modeled 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 for the Switch_1 component, where transitions correspond to possible reconfigurations. The SDL/SLIM models are then translated into the language supported by the back-ends. COMPASTA defines the semantics of the communication between HW and SW, the scheduling constraints and their encoding.

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 into the system model, e.g., a fault injection can model a permanent "stuck-at-zero" fault of the `voltage_out` signal of a battery. xSAP can automatically generate safety artifacts, e.g. Fig. 4 show an example fault tree.

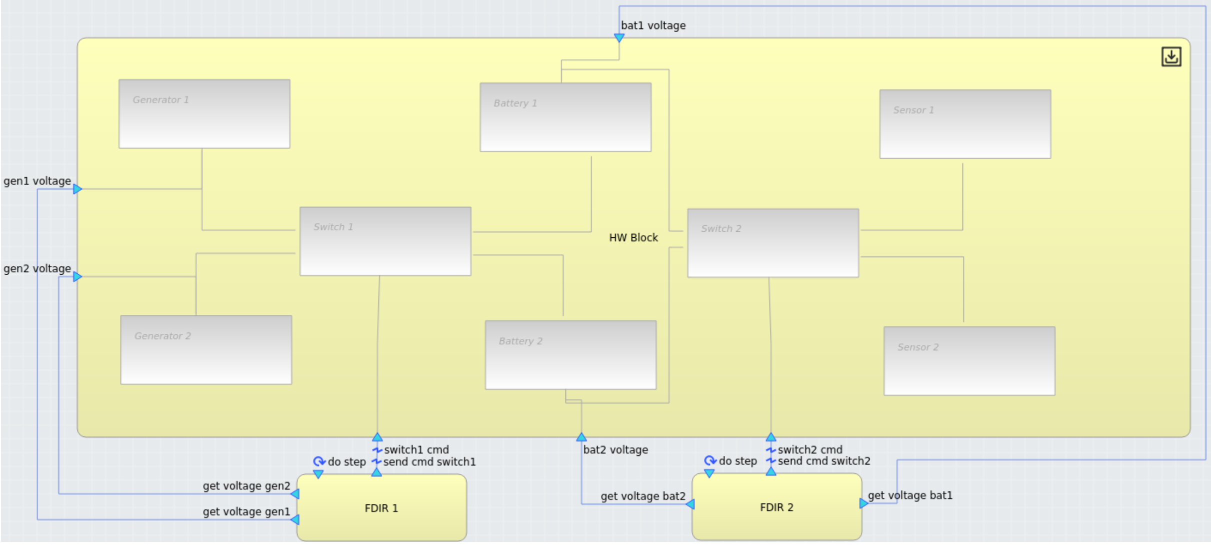
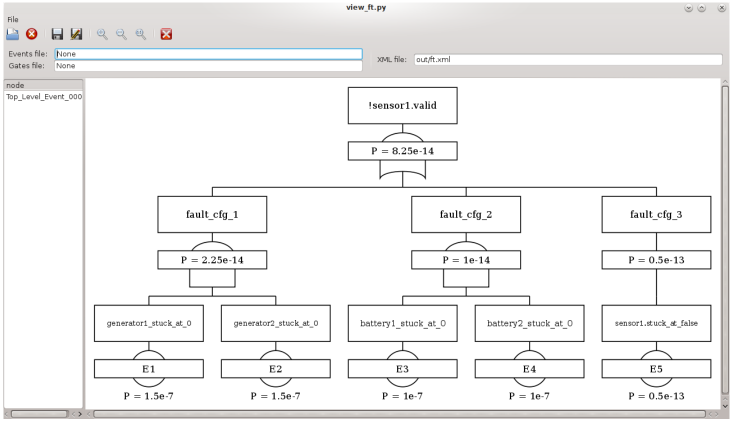


Figure 2: A power system example.



Events file: None
Gates file: None
XML file: out/ft.xml

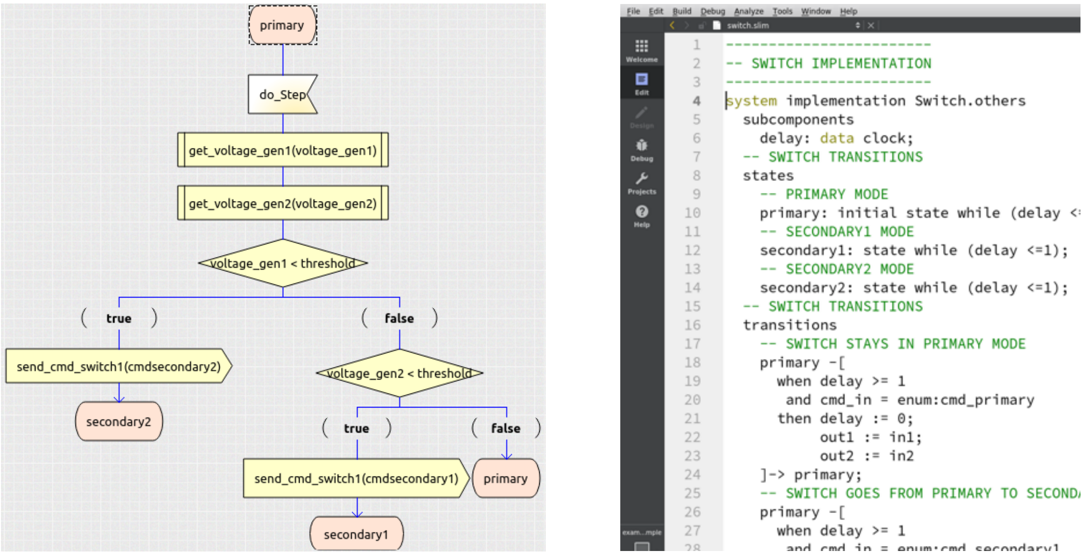
Figure 4: An example Fault Tree.

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 between SW and HW. Then, the deployment of the SW components (binding of the SW to the target HW platform(s)) is specified. TASTE can then be used to generate the executable code for the target platform(s) and to test and simulate the final implementation.

3 Conclusions and Future Perspectives

The COMPASTA project started in April 2021 and will be completed by the end of 2022. 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. 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 integration is based on the COMPASS* idea [5], namely a view where the COMPASS backends are split from the COMPASS front-end and integrated in other model-based design environments such as TASTE. On the same lines, OCRA, nuXmv, and xSAP have been integrated into CHESS for a SysML-based design [6], while FBK is working on the integration of such back-ends into CAMEO and


primary
do_Step
get_voltage_gen1(voltage_gen1)
get_voltage_gen2(voltage_gen2)
voltage_gen1 < threshold
true
false
send_cmd_switch1(cmdsecondary2)
secondary2
voltage_gen2 < threshold
true
false
send_cmd_switch1(cmdsecondary1)
primary
secondary1
File Edit Build Debug Analyze Tools Window Help
switch.slim
Welcome
Edit
Design
Debug
Projects
Help
exam...mple

```
------------------------
-- SWITCH IMPLEMENTATION
------------------------
system implementation Switch.others
  subcomponents
    delay: data clock;
  -- SWITCH TRANSITIONS
  states
    -- PRIMARY MODE
    primary: initial state while (delay <
    -- SECONDARY1 MODE
    secondary1: state while (delay <=1);
    -- SECONDARY2 MODE
    secondary2: state while (delay <=1);
  -- SWITCH TRANSITIONS
  transitions
    -- SWITCH STAYS IN PRIMARY MODE
    primary -[
      when delay >= 1
       and cmd_in = enum:cmd_primary
      then delay := 0;
           out1 := in1;
           out2 := in2
    ]-> primary;
    -- SWITCH GOES FROM PRIMARY TO SECOND
    primary -[
      when delay >= 1
       and cmd_in = enum:cmd_secondary1
```

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

the prototype support for SysML-V2.

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] COMPASS Consortium: COMPASS roadmap. Tech. rep. (2016), available from `http://www.compass-toolset.org/docs/compass-roadmap.pdf`

[6] Debiasi, A., Ihirwe, F., Pierini, P., Mazzini, S., Tonetta, S.: Model-based Analysis Support for Dependable Complex Systems in CHESS. In: MODELSWARD. pp. 262–269. SCITEPRESS (2021)

[7] 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)

[8] MOSAR: Modular Spacecraft Assembly and Reconfiguration, `https://www.h2020-mosar.eu`

[9] nuXmv web page (2021), `https://nuxmv.fbk.eu`

[10] ocra web page (2021), `https://ocra.fbk.eu`

[11] 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)

[12] TASTE web page, `https://taste.tools/`

[13] xSAP web page (2021), `https://xsap.fbk.eu`