Timing analysis of TASTE models for reconfigurable software

Jean-Charles Roger, Pierre Dissaux, Jérôme Legrand, Mourad Dridi, Stéphane Rubini, Frank Singhoff

Timing analysis of TASTE models for reconfigurable software

Model-Based Systems and Software Engineering
(MBSE 2021)
29 - 30 September 2021

Virtual Workshop

Jean-Charles Roger[(1)], Pierre Dissaux[(1)], Jérôme Legrand[(1)], Mourad Dridi[(2)], Stéphane Rubini[(2),] Frank Singhoff[(2)]

[(1)]Ellidiss Technologies
24 quai de la douane, 29200 Brest, France
Email:@ellidiss.com

[(2)] University of Brest,
Lab-STICC, UMR CNRS 6285
6 avenue Le Gorgeu, 29285 Brest Cedex, France
Email: @univ-brest.fr

ABSTRACT

INTRODUCTION

Model Based Software Engineering aims at mastering the complexity of software development processes while ensuring a high level of performance, safety and security that can be estimated early in the life cycle by appropriate verification techniques. Dedicated to the space domain, the TASTE [2,3] approach brings an integrated solution to build embedded real-time software running on a list of predefined execution platforms. The TASTE package comes with a graphical editor, various libraries, tools and Ada/C code generators. The underlying modelling paradigms are based on international standards such as ASN.1 and AADL [6].

MOSAR (MOdular Spacecraft Assembly and Reconfiguration), is a project developed by a consortium of European organizations [1]. This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No. 821966. The goal of the MOSAR project is to demonstrate several scenarios of modular satellite reconfiguration during operation. This includes adding or removing standardized hardware modules of a client spacecraft by a servicer spacecraft. The impact of spacecraft reconfiguration during operation on the on-board software has been studied using the TASTE toolchain that has been enhanced with proper support of operational modes and an early verification approach based on timing simulation.

TASTE FOR RECONFIGURABLE SOFTWARE

TASTE Life Cycle

The TASTE development process is briefly summarized in Figure 1. During step 1, the software architect can build the model through three main views:

- The Data View formalizes all data types that may be used for communication messages.

- The Interface View shows the logical architecture of the software composed of interacting modules.

- The Deployment View describes the physical architecture in terms of computational nodes and networks.

A fourth view, the Concurrency View, is automatically derived from the three above mentioned modelling views. It contains a consistent AADL model including all the required real-time artifacts to capture the software behavior at architectural level. This model is the baseline for source code generation, identified by step 3 in Figure 1. The Concurrency View model can also be efficiently used for early timing analysis to assess timing aspects of the software architecture before code generation. Time aspect verification can be made with the Marzhin AADL simulator [4] and with the Cheddar tool [5]. This is represented by step 2 in Figure 1.

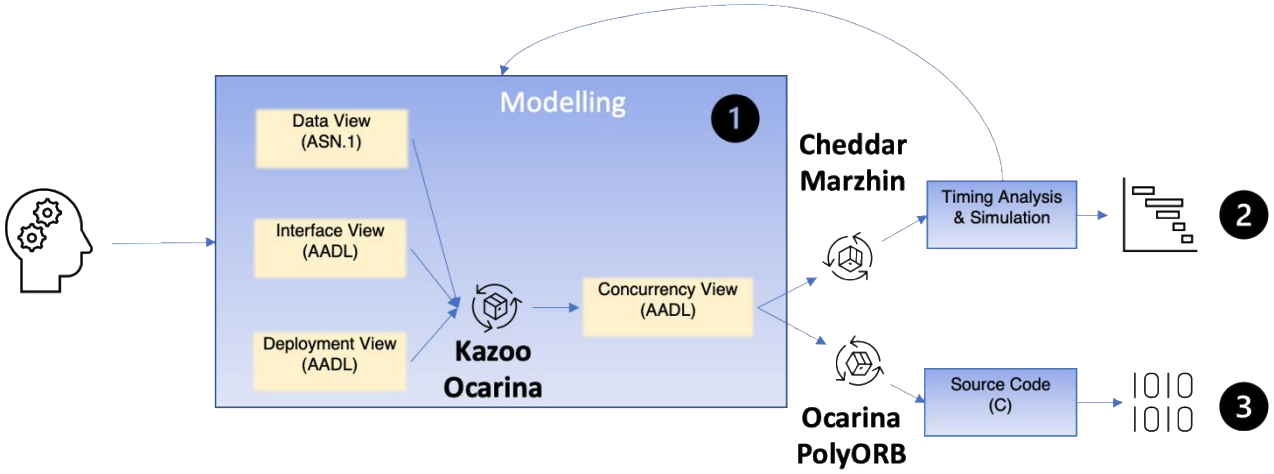

Figure 1: TASTE development life cycle

TASTE Enhancements for Dynamic Software Reconfiguration

A requirement of the MOSAR project is to support software reconfiguration during operation. This can be typically supported by the management of operational modes, which may be expressed within the TASTE model at an applicative level. In order to ease the use of this feature by the software architect and make it more robust and consistent all along the development life cycle, operational modes management has been added to the list of artifacts natively supported by TASTE models and tools.

These enhancements have impacted the following components of the TASTE package:

- Extension of the TASTE meta-model to support modes and modal elements;

- Add-ons in the TASTE editor to capture modal information;

- Update of the Concurrency View generator to reflect the intended mode change behavior;

- Update of the timing analysis tools included in the TASTE toolchain.

CASE STUDY

To illustrate the improvement made in the TASTE toolset, we introduce a case study shown in Figure 2. The case study is based on a simplified model of the MOSAR satellite including reconfigurable payloads. The OBC (On Board Computer) is a fixed subsystem that handles the communication with the ground and all general satellite management tasks. The R-ICU (Reduced Instrument Control Unit) is also fixed and controls the internal communication network. Communications use the R-MAP protocol through SpaceWire connections. The SMs (Spacecraft Modules) are payloads of the satellite than can be removed and replaced during flight operation thanks to a standardized mechanical, electrical and thermal interface. Three of them have been selected for the purpose of our case study: a battery payload (BAT), an optical sensor (OSP1) and a second optical sensor (OSP2).

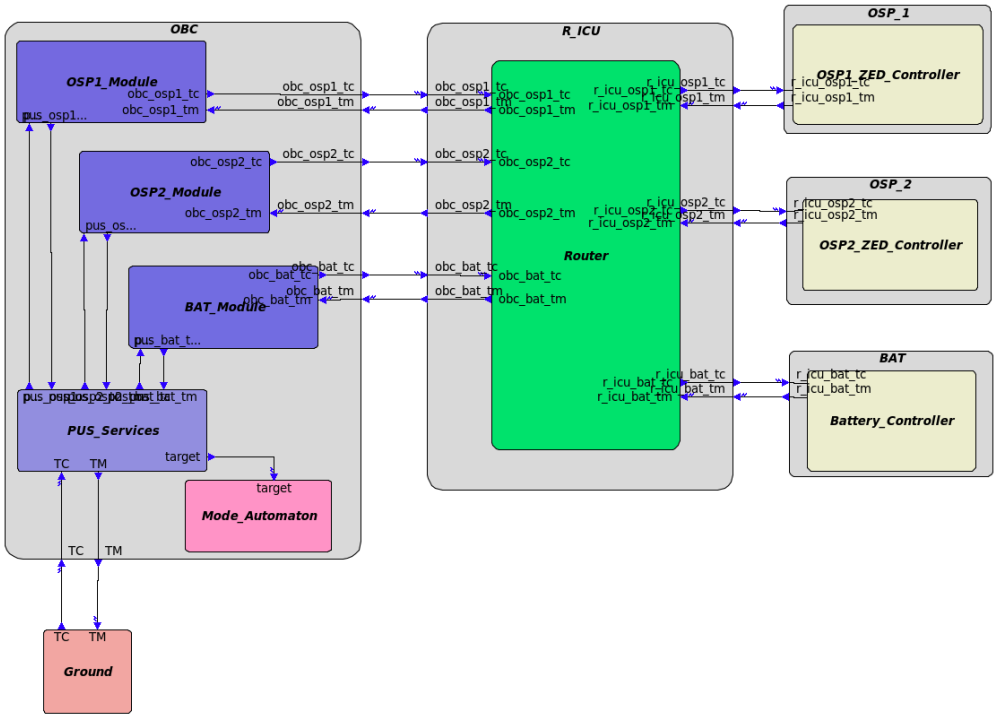

Figure 2: MOSAR Case Study

Marzhin Simulation

This MOSAR model mainly focuses on specific scenarios and transitional regimes between modes to analyze the timing dynamics of software reconfiguration using the Marzhin AADL run-time simulator. The demonstration scenario aims at showing the behavior of the various software tasks that are distributed over the internal satellite network during a reconfiguration operation. Each task in the model has its own behavior defined using the AADL Behavioral Annex leveraged by Marzhin for the simulation. Data exchanged between the tasks through the busses are encoded as integer messages.

Figure 2 presents the TASTE Interface View for the case study. It is a preliminary architecture of a client satellite subject to receipt or disposal of hardware modules during operation. We use such model to run timing simulation with Marzhin. The model contains 3 payloads: BAT (battery), OSP1 (optical sensor 1) and OSP2 (optical sensor 2). The simulation is running with a scenario that reconfigures the system several times and exchange telemetry in the meantime. Figure 3 presents the timelines produced by the Marzhin simulation.

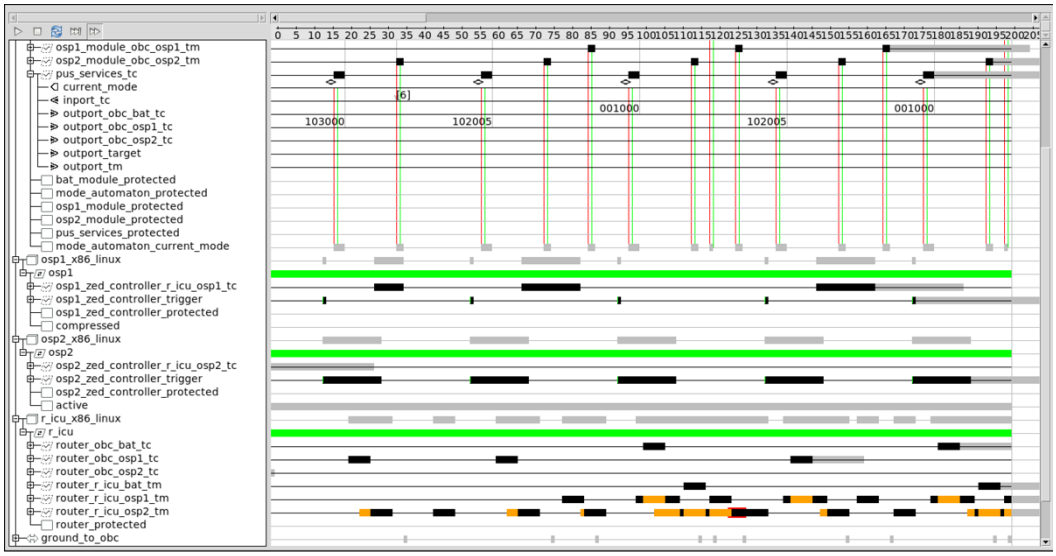

Figure 3 Marzhin simulation chronogram

Worst case timing analysis with the Cheddar tool

The TASTE model shown in Figure 2 can also be used to drive worst case analysis of the timing properties. For such a purpose, we use the Cheddar tool. It focuses on steady regimes, one mode at a time.

Cheddar allows to assess schedulability of a real-time system either by simulation or by mathematical feasibility tests. Contrary to Marzhin which runs online and interactive simulations, Cheddar computes the worst case timing behavior to verify that the timing requirements of the MOSAR case study are never jeopardized. Computing the worst case time behavior requires to specify the worst case behavior of each entity of the TASTE model, which is not required with Marzhin. Furthermore, Cheddar analysis is not run interactively but is made off-line.

For each modal configuration of the MOSAR case study, we generate a Cheddar model on which the schedulability is assessed. This allows us to verify that timing requirements are met on any modal state the MOSAR system can reach. This analysis covers both tasks run on the MOSAR nodes and the SpaceWire communications between those nodes. To both analyze tasks and communications, we apply SCM (SpaceWire Communication Model) which combines task and communications interference specification on the same DAG model for schedulability computation.

CONCLUSION

The full article will start by presenting the TASTE toolset developed at ESA and the extensions added to support dynamic system reconfiguration. It will then describe the MOSAR project and its context. A description of the MOSAR client satellite use case will follow together with the early timing analysis objectives. The first analysis, using the Marzhin simulator will focus on specific scenarios and transitional regimes between modes. The second analysis, using Cheddar will use formal computation and focus on steady states (one mode at a time). We will conclude the article with a comparison between these two approaches and their respective strengths and weaknesses.

References

[1] MOSAR project: http://www.h2020-mosar.eu

[2] M. Perrotin, E. Conquet, J. Delange, A. Schiele, and T. Tsiodras. TASTE: a real-time software engineering tool-chain overview, status, and future. In International SDL Forum (pp. 26-37). Springer, Berlin, Heidelberg, 2011, July.

[3] TASTE project*: http://taste.org*

[4] P. Dissaux, O. Marc, S. Rubini, C. Fotsing, V. Gaudel, F. Singhoff, A. Plantec, V. Nguyen-Hong and N. Hai Tran. The SMART Project: Multi-Agent Scheduling Simulation of Real-time Architectures. 7th European Congress ERTSS Embedded Real Time Software and System, Toulouse, France, February 2014.

[5] F. Singhoff, J. Legrand, L. Nana and L. Marcé. Cheddar: a Flexible Real-time Scheduling Framework. ACM SIGAda Ada Letters, volume 24, number 4, pages 1-8. Edited by ACM Press, New York, USA. December 2004, ISSN:1094-3641.

[6] Feiler, P. H., and Gluch, D. P. (2012). Model-based engineering with AADL: an introduction to the SAE architecture analysis & design language. Addison-Wesley.