A process - oriented verification and validation for real time embedded systems

Un processus - vérification et validation orientées pour les systèmes embarqués en temps réel

Nadia Chabbat & Salim Ghanemi

Department of Computer Sciences, Badji Mokhtar Annaba University, Po Box 12, Annaba, 23000, Algeria.

Article Info

Article history:

Received 21/09/2017
Revised 13/01/2019
Accepted 20/01/2019

Key words

Real Time Embedded Systems (RTES), Timed CSP (Communicating Sequential Processes), Clock Constraint Specification Language (CCSL), MARTE (Modeling and Analysis of Real Time Embedded Systems), V&V (Verification & Validation)

ABSTRACT

Based on formal and robust concepts, the real-time embedded systems allow meeting the requirements for the quality of the systems. Considering the UML MARTE profile as the standard by excellence, it has been the most used in the modelling and analysis of systems. In addition, for the support of time constraints, the CCSL language (Clock Constraint Specification Language) has been proposed. However, the MARTE-CCSL profile allows only formal verification, which represents only the static validation. As a complement to the analysis, it is essential to consider the dynamic validation step as well. In this paper, we suggest a hybrid process-oriented verification approach (HV&V) for MARTE-CCSL models. The HV&V approach is based on a transformation of MARTE-CCSL models to Timed CSP (Communicating Sequential Processes) models. Thus, the Timed CSP model and the generated counter-examples will be automatically used by the validation step. This last step helps to quickly generate a prototype that is functional and verifiable at low cost. The approach is tested on the elevator control system.

Mots-clés

Systèmes Embarqués Temps Réel (SETR), Timed CSP (Processus Séquentiels Communicants Temporisés), Langage de Spécification de Contraintes d'horloges (CCSL), MARTE (Modélisation et Analyse des Systèmes Embarqués Temps Réel), V&V (Vérification & Validation)

RESUME

Basés sur des notions formelles et robustes, les systèmes temps réel embarqués permettent de répondre aux exigences de qualité des systèmes. Vu que le profile UML MARTE est le standard par excellence, il a été le plus utilisé dans la modélisation et l'analyse des systèmes. De plus, pour la prise en charge des contraintes temporelles, le langage CCSL a été proposé. Cependant, le profile MARTE-CCSL ne permet que la vérification formelle qui représente uniquement une validation statique. Comme complément à l'analyse l'étape de validation dynamique a été considérée. Dans cet article une approche de vérification hybride (HV&V) orientée processus pour les modèles MARTE-CCSL est proposée. L'approche HV&V repose sur une transformation de modèles MARTE-CCSL vers des modèles Timed CSP. Ainsi, le modèle Timed CSP et les contre-exemples générés seront exploités automatiquement par l'étape de validation. Cette dernière étape a permis de générer rapidement un prototype fonctionnel et vérifiable avec un faible coût. L'approche est testée sur un système de contrôles d'un ascenseur.

Corresponding Author
Nadia Chabbat
Department of Computer Sciences, Badji Mokhtar Annaba University,
Po Box 12, Annaba, 23000, Algeria.
Email: Lydia.chab@hotmail.fr

1. INTRODUCTION

Real-time Embedded systems (RTES) are becoming increasingly complex and require a high level of safety and reliability, especially in high-risk applications involving human lives. The functional verification phase is one of the most time consuming phases among the various design phases (more than 50% of the cost). A poorly designed product entails a very high maintenance cost and a considerable time reduction of the product-to-market [1]. The complexity of many RTES requires the application of a battery of technics. The use of formal verification and dynamic validation methods are two interesting and promising ways of reducing the overall system development cost, ensuring development reliability as well as achieving a complete end product without any faults. Formal verification and dynamic validation are two joined technics. Verification is the means of establishing the mapping between a software product and its specification, and validation is the means to ensure that the software achieves the function for which it was designed. It is often said, in a less formal way, that validation is meant to ensure that the *right software* has been realized, and that the verification decides if it has been *realized rightfully*.

The UML MARTE profile is imposed as a standard specified by the OMG [2] for the modelling and analysis of real-time embedded systems [3]. It specifies concepts for characterizing UML elements to model temporal entities, software and hardware platforms, resources, and quantitative characteristics on a system, such as execution time. In UML MARTE, time constraints are defined and used in conjunction with the Clock Constraint Specification Language (CCSL) [4]. CCSL is a declarative language, annexed to the specification of the UML MARTE profile to specify, solve and simulate time constraints by the Time Square tool [5]. However, once the software is modelled, the difficulty lies in the expression of appropriate properties and formal checks. In order to meet these requirements, some formal analytical approaches have been developed. All are based on the transformation of a high-level model written in MARTE towards a specific model that can be verified formally either by dedicated tools such as UPPAAL [6] and ROMEO [7] or by robust semantic languages such as PROMELA [8] or FIACRE [9] where they can then be used with formal verification methods, for example based on automata or Petri nets. However, current methods for verifying UML MARTE models and CCSL specifications focused only on static verification system. To complete the analysis, we considered it essential to integrate the dynamic validation step into the same verification process.

In this article, we suggest a Hybrid Verification and Validation approach (HV&V). The advantages of the approach are more flexibility in the design, specification is more expressive and full and powerful automated verification. Dynamic verification relies essentially on the automatic generation of a prototype described in an appropriate language and its execution on a set of tests. Since we have selected a process-oriented approach, we suggest using two languages: Timed CSP and OCCAM [10, 11]. The latter is used to describe the prototype while the former allows formal verification. The introduction of temporal logic in the Timed CSP specification makes it possible to express easily safety and liveness properties of systems [12].

Our HV&V approach is based on a transformation of a model described in MARTE-CCSL towards a model described in Timed CSP (Fig. 1). The properties to be checked are expressed in the time logic LTL. From the Timed CSP model, after its formal verification, a prototype in OCCAM is generated automatically. The latter is executed on a set of tests derived from the counter-examples generated during the formal verification step. The rest of the article is structured in the following way: in the second section, we discuss some related work. The architecture of the proposed HV&V approach is explained in the third section. In the fourth section, we apply our suggested HV&V on a case study. Finally, we give a conclusion and some perspectives in the fifth section.

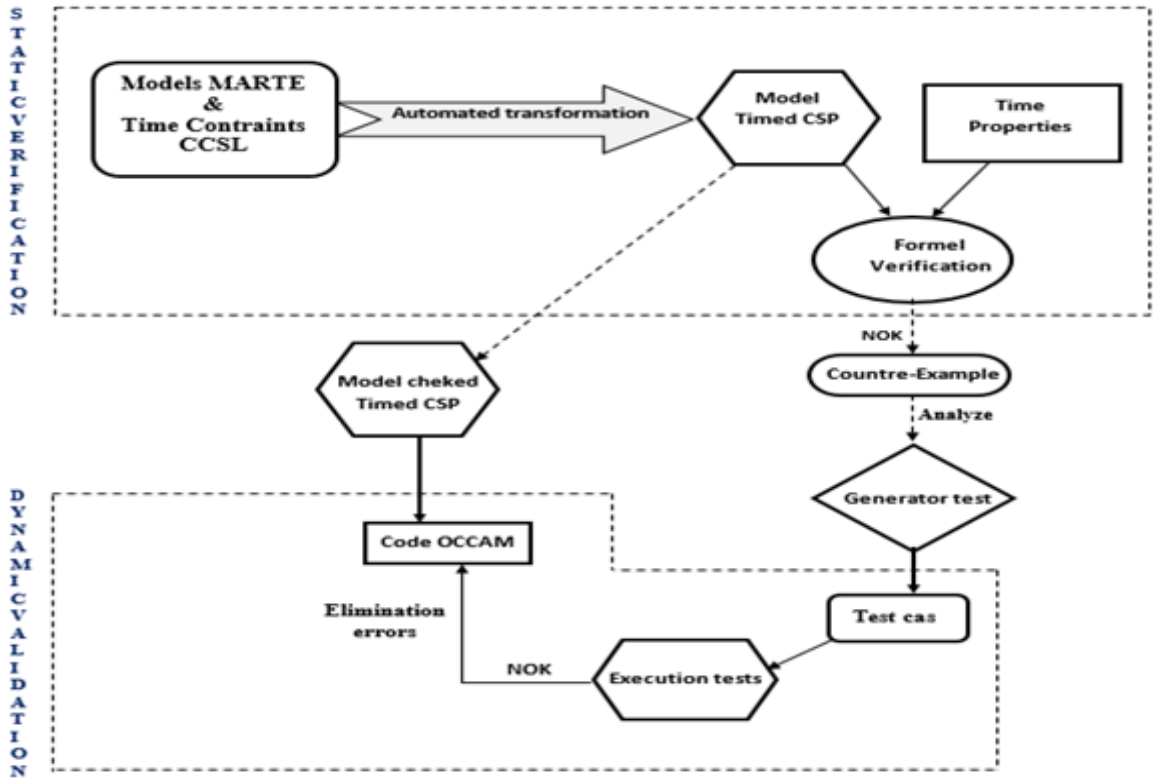

Figure 1: Architecture of HV&V approach

2. RELATED WORK

Several model transformation-based approaches are published that allow formal verification of MARTE-CCSL specifications. C. André [13] suggested a systematic approach transforming the CCSL specifications into Esterel observers and checking them using the Esterel Studio tool [14]. The tool was then improved by the development of a library of observers. A similar approach was also applied in [15]. It aimed to generate VHDL observers with the definition of a state-based semantics for some CCSL operators and relied on the labelled transitions system (LTS). The technique for generating CCSL specification observers was presented in reference [16]. H. Yu et al. [17] presented a framework for the exhaustive verification of CCSL specifications using the Sigali model-checker. However, this approach was limited because it focused only on the implementation of CCSL constraints in signal language. Thus, it allowed us to check only the temporal constraints without taking into consideration the verification of functional properties. L. Yin and F. Mallet[18] suggested a correct transformation of the CCSL specifications to a Promela model and verified its accuracy by the SPIN model-checker, for example the verification of the properties of deadlocks, assertions and expected properties. However, the disadvantage of this transformation is the implementation of the clocks in the CCSL specification by global variables, which negatively affects the verification time, and consequently does not avoid the combinatorial explosion problem during the exploration of models. J. Suryadevara et al. [19] presented a strategy to transform a MARTE-CCSL behavioral model into a timed automaton, and to check the logical and chronometric properties using the UPPAAL model-checker. This strategy only takes into account time specification and verification. N. Menad and P. Dhaussy [20] suggested a more general verification approach, which allows automatic transformation of MARTE-CCSL models into FIACRE models. It checks not only the CCSL constraint implementations, but also the properties of the model including all functional components. The properties are expressed in observer automata in the Context Description Language (CDL), and are checked by the OBP (Observer Based Prover) model-checking tool. Thus, the separation of the functional properties of the application through CDL language facilitates the separation of concerns.

Other approaches, based on the transformation of a specification language to another language supporting formal verification methods and model-checking tools have been suggested [21, 22]. Both works use the Time Petri Net Analyzer (TINA) model-checking tool for formal verification of UML MARTE structural and behavioral diagrams. M.P. Ning Ge and X. Cregut [21] suggested a verification approach to ensure the accuracy of temporal properties by transforming the activity and state machines diagrams into a Time Transition System (TTS). This approach does not take into account the time aspect of the system, and is limited by verification of two types of properties, synchronization and schedulability. M.P. Ning Ge and X. Cregut [22] presented a framework for transforming UML structural and behavioral models to TPN models in order to perform and verify temporal properties by TINA model-checker.

In this paper, we suggest a verification approach (HV&V) hybridizing the various technical types of static and dynamic analysis in order to improve current processes of verification and confidence carried to software products. Our HV&V approach not only verifies CCSL constraints and functional properties, but also the dynamic requirements of the system (the requirements are dynamic). The dynamic validation technique used is based on the observation of a selected set of controlled executions, or test cases provided from the counter-examples that have been generated from the formal Timed CSP model.The aim researched of the dynamic validation is not to correct the program but to allow the detection of program failures with low implementation costs. Thus, one of the advantages of using the OCCAM language, which is directly related to the CSP model, is the fast generation of an executable and verifiable prototype through software testing. This possibility is very effective for the programming of critical real-time and embedded applications whose safety is essential. It is thus possible to guarantee the accuracy of the algorithms and their implementation.

3. ARCHITECTURE OF THE PROPOSED APPROACH

HV&V approach combines two complementary techniques: static verification and dynamic validation for real-time embedded systems. The first technique consists of formally verifying the specifications of the Timed CSP model, obtained by transforming the MARTE-CCSL model, using the PAT model-checker. While the second technique exploits the model, verified Timed CSP and the results obtained from the first technique. The aim is therefore to validate the dynamic behaviors of the RTES by simulation. In the following sections, we will go through these two techniques, the used tools and the interactions between them.

3.1. Technique 1- Static verification

The aim of this static verification technique is to transform both MARTE-CCSL architecture models and temporal behavior into Timed CSP executable and verifiable models. The transformation of MARTE-CCSL specifications to Timed CSP descriptions formally describes the behavior of an architecture using sequential communicating processes. Timed CSP provides accurate semantics (trace, failures / divergence,) and robust verification tools such as FDR, PAT, ProBe, ARC [23, 24, 25, 26]. The behavior of the processes is described by the operations of composition (parallelism, sequence, and temporal occupation) and syntactic rules, according to the events caused by changes of state in the system, and to check the functional properties by Equivalence and bi-simulation techniques, and as a function of the temporal characteristics of the modeled processes.

After the transformation of the MARTE-CCSL model into a Timed CSP model, we proceed to the verification of some properties of the system such as safety, liveliness and free deadlock using the PAT tool. The formal verification process is illustrated in figure 2. Formal verification of Timed CSP specifications in the PAT model-checker is performed in three steps:

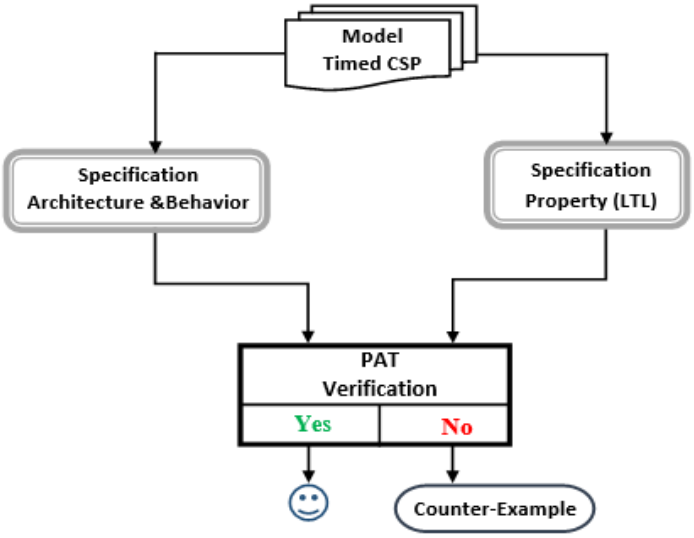

Figure 2 : Formal verification of a timed CSP model

- Step 1 – Modelling

In this step, the Timed CSP system specifications are automatically translated into PAT specifications in the form of labelled transitions system (LTS). Indeed, it is necessary to interpret the Timed CSP specifications in order to extract a formal model specific to the PAT model-checker. Thus, it is possible to formally check the Timed CSP system. To minimize the size of the state space, to avoid the combinatorial explosion problem of the state space of labelled transitions systems, and to perform efficiently, advanced optimization technics are implemented in the PAT, for example, partial order reduction, symmetry reduction, abstraction-counting process.

- **Step 2 – Specification**

The specification is the formal expression of the properties of the model to be verified in a specialized language. For the PAT tool, the temporal properties to be checked by PAT such as safety, liveness and fairness are expressed in linear time logic (LTL) [27]. The set of properties and the labelled transitions system (LTS) are entered as input to the verification step.

- **Step 3 – Verification**

This step represents an execution of the model checking by the PAT tool through a formal analysis and an exhaustive path of the labelled transitions system model. At the end of each verification, the PAT model-checker provides two distinct outputs:

1. Positive response: Property satisfied for all the behaviors of the model.
2. Negative response: Unsatisfied property as well as illustrative counterexamples are generated.

Counter-examples are used to generate test cases for dynamic validation (simulator execution). In addition, to the verification technic, we can simulate or debug models by creating an executable system. We can use an interactive simulation and debugger to check each step-by step interaction, and thus we know the state or interactions that are executable. These analyses assess the reliability of the system, refine and correct the behavior of the systems.

3.2. Technique 2- Dynamic validation

Once the formal verification of the Timed CSP model is established, the dynamic validation technic is initiated through the tests. This technic aims to validate the dynamic behaviors taking into account the evolutionary aspects (functional, temporal) of the software during its execution. It consists mainly in executing an OCCAM code generated automatically from the Timed CSP model on a set of tests derived from the counter-examples of the model-checker PAT.

The use of the OCCAM language as well as the appropriate tools such as the KRoC [28] compiler, the virtual machine (VM) or even more a network of transputers, leads to an efficient simulation. The hierarchy included in the language allows a decomposition into increasingly fine

levels of abstraction, in order to adjust the precision of the desired simulation type. Implementation of this step involves three main activities: test scenarios generation, execution, and output analysis (Fig. 3).

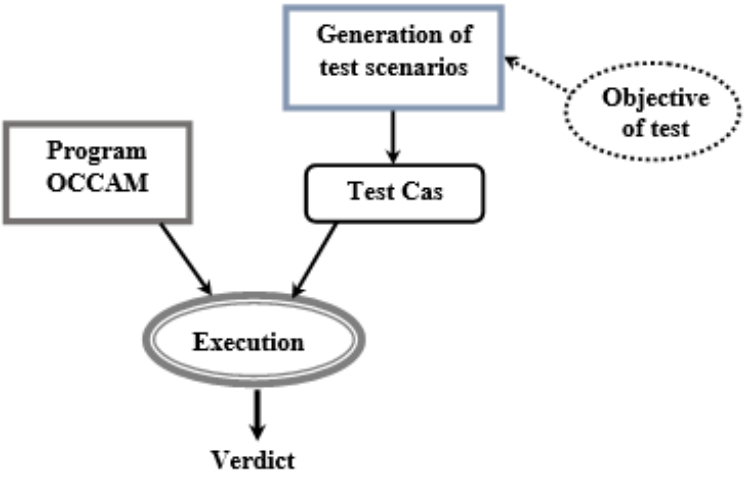

Figure 3: Overview of software test

1) Test scenarios generation

This step involves generating scenarios to validate a property that should be checked by the model. It is based on an objective to define a test case. The scenario is done by creating test cases that reproduce end-user usage. A scenario can be an independent test case or sequence of test cases that follow. The selection of test scenarios has received a great attention in the scientific literature in this field, in other words, how to identify a sequence of test cases that is effective. The construction of the test scenarios is expensive and source of error. An effective test set is one that demonstrates that the software behaves as expected, or otherwise, by observing existing dysfunctions. Obviously, an effective sequence of test cases is the starting point for successful test scenario.

We believe that counter-examples are a reliable source from which to generate efficient test cases as these counter-examples have already been verified automatically by the model-checker PAT. They are obtained by identifying the properties that the system must satisfy. In fact, in order to obtain test cases, emphasis must be placed on negative properties, where the model-checker PAT generates counter-examples, which are transformed manually and /or automatically into executable tests. The purpose is to generate efficient test cases for critical parts of the system.

2) Test scenarios execution

It represents the transition from the test scenario described during the design of the test plan to the achievement of concrete results, which gives us a good visual idea of the quality of the product. In addition to running the prototype itself, the KRoC compiler in the OCCAM language is able to ensure the temporal validity of the values taken by the variables and of the data flows traversing the prototype. In this way, it was possible to avoid the generation of unrealizable architectures in real-time leading to development cost reduction, fast development time and reduced product-to-market time. Finally, it is possible to check some behavioural properties such as deadlock or dead branches.

3) Output Analysis

The main purpose of this step is to detect possible errors in the model. Based on the results provided by this step, the system observer analyses the test results provided by the previous step, and therefore make a decision:

- *Pass:* the test has been successfully executed and the property expressed by the test objective is checked.
- *Fail:* an error is observed during the test.
- *Inconclusive:* the observed behaviour is correct but does not correspond to the objective of the test.

4. VALIDATION OF TIMED CSP MODELS

Formal verification and validation of critical real-time and embedded critical systems are two fundamental and important steps in the systems development cycle. In order to take benefit of the advantages of these methods and for the sake of efficiency, a hybridization of these two technics has been proposed. In this section, we will describe the validation step on a case study: an Elevator Control System (ECS).

We consider the case of a single elevator in a building with N levels used to carry up and down users as well as goods. The elevator control system (ECS) is an embedded real-time system implementing interaction between sensors, actuators and users. Figure 4 gives an overview of these interactions. Various sensors measure the state of the ECS system. The sensor information is sampled and processed by the controller. Then the result of the processing is converted into analogue signals generating direct actions on the environment (Displayed information on to the screen for the users, doors, motors, etc.).

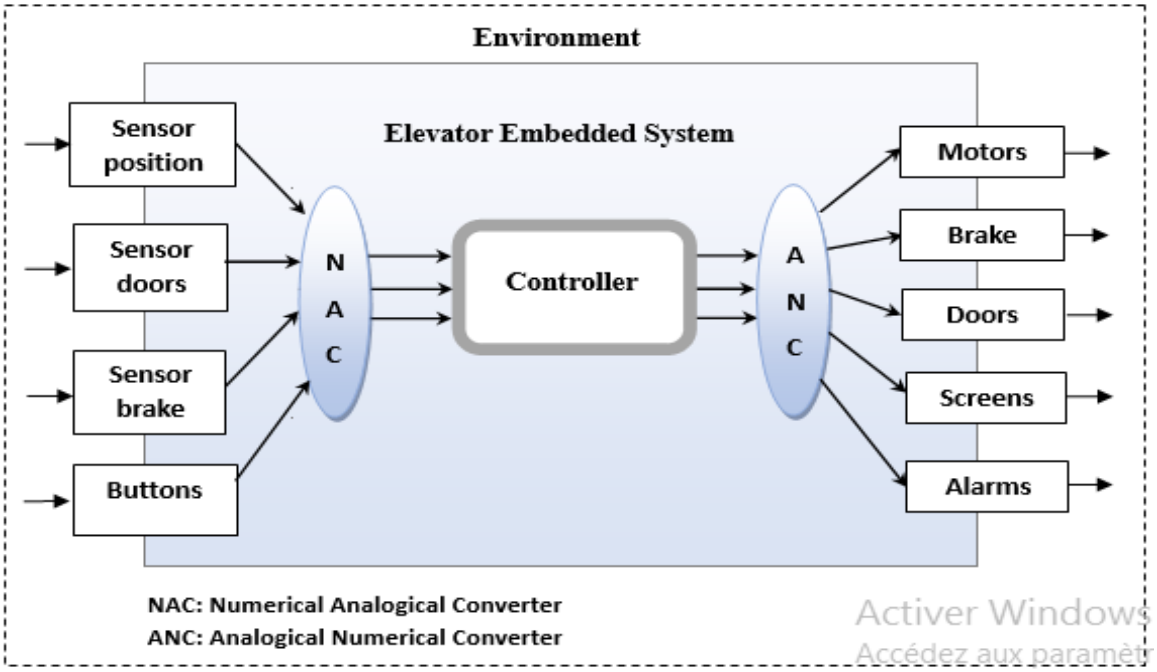

Figure 4 : Elevator system structure

From the formal verification step, a prototype was generated in OCCAM modelling the ECS system. The driver OCCAM code is given below:

```
INT n:    -- numbers stages
  SEQ
    While running
      PAR
        Sensors.position ()
        Sensors.floor ()
        Sensor.brake ()
        Sensor.doors ()
        Controller ()
        Motor ()
        Buttons ()
        Brake ()
        Doors ()
        Alarms ()
        Display ()
```

The code shows a parallel activation of a set of four (04) sensors and seven (07) actuators needed in the operation of any elevator. The program is compiled in the KRoC environment under the Linux operating system Ubuntu 16.04 LTS by the command (*kroc Elevator.occ -lcourse*). Running the prototype through the command (*. / Elevator*) allowed us to check some properties such as, deadlock and livelock. Figure 5 indicates a deadlock detection in the prototype. Using the KRoC debugger, we were able to identify the source of the error and after correcting the error, we were able to validate the system.

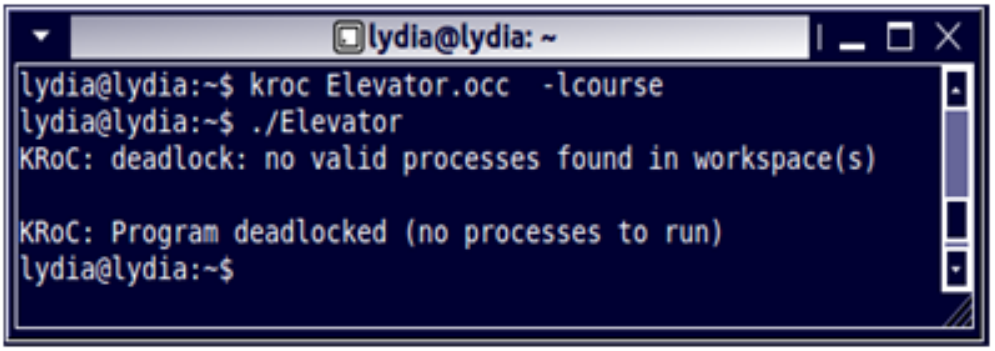

Figure 5 : Case of deadlock on the program of ECS

It was found that the rapid and automatic generation of an executable prototype in OCCAM as well as the examples generated during the formal verification step contributed greatly to the implementation of the validation. Thus, the HV&V approach allows formal verification and validation in a methodical and complementary way.

5. CONCLUSION

In this paper, we showed the benefits of our suggested Hybrid Verification and Validation approach for MART-CCSL models. The overall goal of this work was to create and enrich a conceptual and operational toolkit for modelling, verifying and validating RTES systems.

This toolkit was meant to be complementary and non-substitutable to existing tools. Thus, we used the Timed CSP specification language to detect and correct errors in MARTE-CCSL models in the early stages of the development life cycle. The use of the OCCAM language, which is closely related to the Timed CSP allowed us, first of all, to automatically develop the prototypes of the systems and, secondly, to check their temporal behavior.

The case study of an elevator control system enabled us to implement just the validation step of the HV&V approach and confirm its benefits.

As further work, we suggest firstly, to apply the complete analysis process V&V on the case study Elevator Control system. Secondly, to extend our HV&V model to applications that are more complex.

REFERENCES

[1] S. Miller, *et al.,*"Early Validation of Requirements through Formal Methods", *Software Tools forTechnology Transfer*, volume 8, number 4, 2006.

[2] OMG. Unified Modeling Language (UML), version 2.4.1, http: //www.omg.org/spec/UML/2.4.1/, 2011.

[3] S. Gérard*, et al.,* "Modeling languages for realtime and embedded systems", *New Technologies of Distributed Systems (NOTERE)*, vol 6100 of Lecture Notes in Computer Science, p129–154. Springer Berlin, 2011.

[4] F. Mallet., "Clock constraint specification language: specifying clock constraints with UML/MARTE", *Innovations in Systems and Software Engineering*, vol. 4, no.3, 309–314, 2008.

[5] J. Deantoniet and F. Mallet., "Time Square: treat your models with logical time ", in C.A Furia, S Nanz, (eds.) TOOLS'12. LNCS, vol. 7304, 34–41, Springer, Heidelberg, 2012.

[6] K.G. Larsen, *et al.,* "UPPAAL in a Nutshell", *Int. Journal on SoftwareToolsfor Technology Transfer*, 1(1–2):134–152, 1997.

[7] Romeo tools, https://romeo.rts-software.org/, 2018.

[8] J. Gerard, "The model checker spin", *IEEE Transactionson Software Engineering*, 23:279–295, 1997.

[9] P. Farail, P. Gaufillet, F. Peres, J. Bodeveix, M. Filali, B. Berthomieu, S. Rodrigo, F. Vernadat, H. Garavel and F. Lang, 2008. FIACRE: an intermediate language for model verification in the TOPCASED environment,
In European Congress on Embedded Real-Time Software (ERTS)'08, Toulouse, 2008.

[10] G.M. Reed and A.W. Roscoe, "A Timed Model for Communicating Sequential Processes ", In Proceedings of
ICALP'86, Springer-Verlag, LNCS 226, p 314–323. (Also, Theoretical Computer Science'88, 58, p249–261),
1988.

[11] Inmos Limited,"Occam Programming Manual, Prentice Hall", 1984.

[12] A.F. Ates *et al.,*"Using Timed CSP for Specification Verification and Simulation of Multimedia Synchronization", *IEEE journal on selected areas in communications*, vol. 14, no.1, 1996.

[13] C. André,"Verification of clock constraints: CCSL observers in Esterel ", *Tech. Rep*, 7211, Inria, p 59, 2010.

[14] F. Boussinot and R. D. Simone, "The esterel language", *Proceeding of the IEEE*, 79 (9), 1293-1304, 1991.

[15] F. Mallet *et al.,*"Vhdl observers for clock constraint checking", In SIES 2010, Proceedings of the 2010

Symposium on Industrial Embedded Systems, IEEE Computer Society, p 98–107, 2010.

[16] H. Yu *et al.,*"polychronous controller synthesis from MARTE CCSL timing specifications", in Memocode, 2011.
[17] Dutertre .B, 1992."Spécification et preuves de systèmes dynamiques". PhD thesis, Université de Rennes I, IFSIC.

[18] L. Yin and F. Mallet,"Correct transformation from ccsl to promela for verification", *Technical Report* 7491, Inria, 2011.

[19] J. Suryadevara *et al.,*"Verifying MARTE/CCSL mode behaviour using UPPAAL", in SEFM, p 1-15, 2013.

[20] N. Menad and P. Dhaussy, "A transformation approach for multiform time requirements", in 11th International
Conference on Software Engineering and Formal Methods (SEFM'13), Madrid, Spain, vol. 8137, p 16{30, Lecture Notes in Computer Science, 2013}, 2013.

[21] M.P. Ning Ge and X. Cregut, "Time properties dedicated transformation from UML-MARTE activity to time
transition system", pp 37(4):1-8, 2012.

[22] M.P. Ning Ge and X. Cregut, "A framework dedicated to time properties verification for UML-MARTE specifications", 2012.

[23] A.W. Roscoe, "Model checking CSP", in a classical mind: essays in honour of C.A.R. Hoare, Prentice Hall, 1994.

[24] S. Jun *et al.,*"PAT: Towards Flexible Verification under Fairness", Proceedings of the 20th International Conference on Computer-Aided Verification (CAV 2009). Lecture Notes in Computer Science. 5643. Springer.
Retrieved, 2009.

[25] Process Behaviour Explorer ProBE User Manual, http://www.fsel.com/software.html, 2003.

[26] K. Parash *et al.,*"ARC - a tool for efficient refinement and equivalence checking for CSP ", IEEE Int. Conf. on
Algorithms and Architectures for Parallel Processing ICA3PP '96. P 68–75, 1996

[27] A.P. Sistla and E. Clarke,"The Complexity of Propositional Temporal Logics", *the Journal of ACM*, 32:733–
749, 1986.

[28] KROC: http://wotug.org/kroc/, 2005.