3rd International Conference on Automation, Control, Engineering and Computer Science (ACECS'16)

Proceedings of Engineering & Technology (PET)
pp. 721-725

Using NCES for Modeling and Validating Dynamic Adaptation

Mohamed Naija, Samir Ben Ahmed

Laboratory of Computer for Industrial Systems
Carthage university, INSAT, Tunisia
naija.mohamed@gmail.com
samir.benahmed@fst.rnu.tn

***Abstract*— It is becoming increasingly important to be able to adapt a system's behavior at run time in response to changing requirements and environmental conditions. Crucially, the adaptation model includes invariant properties and constraints that allow the validation of the adaptation rules before execution in order to produce a correct system configuration that should be executed. The formal methods have proven to be useful for making the development process reliable at a high abstraction level. Based on this approach, this current research proposes a generic process to specify, design, and verify Adaptive Real-Time Embedded Systems. This contribution uses the formalism Net Condition Event System (NCES) for modeling and validating adaptive systems with the model Checker SESA. We illustrate the advantages and effectiveness of our proposal by modeling and verifying an automotive system.**

Keywords- Real-Time Embedded Systems; adaptability; Design; NCES; Verification.

I. INTRODUCTION

The design of safe and efficient Real-Time Embedded Systems (RTES) is one of the biggest challenges facing designers of such systems. These systems are considered high-assurance since errors during execution could result in injury, loss of life, environmental impact, and/or financial loss [1]. The addition of adaptability to RTES further hardens and delays their modeling and validating especially with the current lack of design models and tools for adaptive RTES.

In our previous work [2], adaptation is defined as any modification in the structure, behavior or architecture of the system to accommodate external or internal change of their operating environment or context and according to predefined adaptation plan and rules. In order to specify, verify, and validate an adaptive system before its realization, it is important to have mechanisms to ensure that the system functions correctly during and after adaptations. Model checking offers an attractive approach to automatically analyzing models for adherence to formal properties. Thus, it allows the designer to face the development risks for both functional properties and non-functional properties (NFP) (such as efficiency, reliability, robustness, stability, and vivacity). Indeed, among the existing formalisms, we bet our choice on the Net Condition Event System (NCES) [3] formalism due to its special sufficiency to support the embedded control systems. NCES is modular with extra condition/event signals and can be verified using the model checking [4]. The possibility of reuse [4] and firing several transitions simultaneously, make it more powerful than Petri nets [11]. Furthermore, its hierarchical composition allows the considerable reduction of the size and complexity of the nets.

In this paper, we propose a new design approach that relies on modeling and verifying techniques in order to validate complex, adaptive, and critical systems. This approach consists in representing the dynamic behavior of the system by (i) enumerating the system's operational modes, by (ii) representing mode switches into communicating mode automata, and by (iii) specifying which of the architecture characteristics are valid or not in a given mode. This information is then interpreted in order to validate this adaptation specification. Indeed, we consider a mode as the abstract definition of a set of functionalities provided by a system or a subsystem. When adapting to new operational conditions, a system may have (i) to switch from a source mode to a target mode, and (ii) to modify the software application configuration (e.g. by disabling or enabling communication links between components). The model validated at design time is used at runtime. In this work, as for the formal verification engine, we apply the model checker SESA [5] to check NCES-based models of components and to verify requirements and functional properties. Moreover, SESA allows performing analysis of typical properties such as the liveness of transitions and boundless of places of the net. It computes also sets of reachable states exactly and effectively [5].

This work facilitates complex systems modeling, reduces the development time and cost and improves software process quality. The above benefits have been illustrated through the application of the proposed process to an automotive case study.

The outline of this paper is as follows. Section 2, provides an overview of the formalism Net Condition Event Systems. In section 3 related work is discussed. In section 4 a presentation of the case study is given. Section 5 gives a description of our proposal. Finally, section 6 concludes the paper and sketches some future work.

ISSN: 2356-5608

II. OVERVIEW OF NCES

In this section we provide enough information about Net Condition Event Systems to understand how we use them in our approach.

Net Condition/Event Systems (NCES) are an extension of Petri nets. They were introduced by Rausch and Hanisch in [3] according to which a completely composed NCES is a Place-Transition Net formally represented as follows:

$$NCES = (PTN, CN, WCN, I, WI, EN, em) \quad (1)$$

where:

- PTN : is a classic Place/Transition Net;
- $CN \subseteq (P \times T)$ is a set of condition signals;
- $W_{CN} : CN \mapsto \mathbb{N}$ defines a weight for each condition arc ;
- $I : \subseteq (P \times T)$ is a set of inhibitor arcs ;
- $WI : I \mapsto \mathbb{N}$ defines a weight for each inhibitor arc ;
- $EN : \subseteq (T \times T)$ is a set of event signals ;
- em : maps an event-processing mode (AND or OR) to each transition.

Each system state is represented by a marking M of the net. The notation *M(p)* denotes the number of tokens in place *p* in marking *M*. M_0 is the initial marking of the net. A place *p* is called a source place of a transition *t* if there is a condition signal from *p* to *t*. A transition $t' \in T$ is called a forcing (resp, forced) transition of transition *t* if there is an event signal from *t'* to *t* (resp, from *t* to *t'*)

The semantics of NCES are defined by the firing rules of transitions [6]. There are several conditions to be fulfilled to enable a transition to fire. First, as it is in ordinary Petri nets, an enabled transition has to have a token concession. That means that all pre-places have to be marked with at least one token. In addition to the flow arcs from places, a transition in NCES may have incoming condition arcs from places and event arcs from other transitions. A transition is enabled by condition signals if all source places of the condition signals are marked by at least one token. The other type of influence on the firing can be described by event signals which come to the transition from some other transitions. Transitions having no incoming event arcs are called spontaneous, otherwise forced. A forced transition is enabled if it has token concession and it is enabled by condition and event signals.

III. RELATED WORK

There have been a number of approaches proposed for the modeling and verification of RTES from high level models. We will discuss in the following the methodologies that particularly deal with high level modeling and verification of adaptive RTES, which are still not well tackled.

In [7], authors have benefited from MARTE to model reconfigurable architectures such as FPGAs based Systems-on-Chip (SOC). They extended the MARTE profile with some semantics and Xilinx specific concepts, which limits their applicability for diverse systems, to support Dynamic and Partial Reconfiguration (DPR) of FPGA. The functional model is mapped into a hardware accelerator which is considered as a reconfigurable region having different implementations. Unlike this contribution, we aim to propose a process to verifying adaptive systems which is independent from any specific platform.

In the context of verification approaches, Boukhanoufa et al. proposed in [8] an MDE approach for modeling and offline validation of application timing constraints. In fact, this article uses state machine to represent the application configurations and transitions between them to represent adaptation rules. This work is based on the generation of all possible configurations of a system before running, in order to validate timing constraints. The number of configurations varies from one system to another and it can be very large, this combinatorial explosion makes the timing analysis inapplicable.

In the same vein, [9] have presented a model driven framework for the modeling, verification and application reconfiguration of multitask networked but only for non-functional requirements.

Other efforts have been specifically based on Model checking to automatically analyze adaptive software models by separating the steady-state program verification from the adaptive logic verification. Ramirez et al. present in [10] an iterative approach of modeling and analyzing the behavior of the adaptation logic through UML state diagrams. The adaptive models are analyzed for adherence to both system invariants and properties that should hold during adaptation through the Spin model checker. Zhang et al. propose in [1] a modular model checking approach to verify that a formal model of an adaptive program satisfies its requirements specified in Linear Temporal Logic (LTL) and A-LTL (an adapt-operator extension to LTL), respectively. By separating concerns at the model level, they assume that each adaptation can be only partial. Unfortunately, the full adaptation is difficult to analyze.

IV. CASE STUDY PRESENTATION

In this section we present a specific case of study in the automotive domain, i.e. the *piloting system*. This system is considered as a complex and high-assurance RTES, which can operate in two operational modes: an automatic mode (A) and a manual mode (M).

The *piloting system* consists of two sub-systems. A sub-localization system, that provides the current position of the system every ten milliseconds, and a navigation subsystem whose behavior is different according to the current mode of the system: in automatic mode, the navigation subsystem computes the guidance commands when receiving the current position of the system from the localization subsystem; in manual mode, the navigation computes the guidance commands from orders of an end-user of the vehicle. Fig. 1 illustrates the functional model of this system.

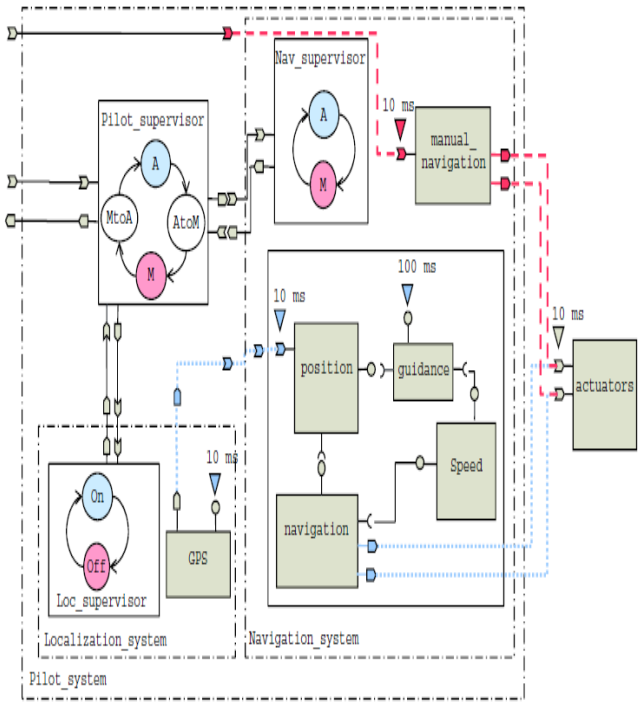

Fig 1. System Functional Model.

The mode automaton of the *piloting system* is represented on the top left corner. It actually defines four modes: *A* represents the automatic mode, *M* the manual mode, while *MtoA* and *AtoM* are transitional modes. When the *piloting system* is running in manual mode, there are two possibilities to switch to automatic mode (*M*→*A*). In the first one, the switching can be requested by the user. If the localization subsystem is off, then the user request is rejected and the system returns in mode *M*. In addition, when the communication sub system is disconnected, then the transition *M*→*MtoA* is fired and an adaptation occurs for setting the navigation subsystem to mode *A*. Mode switch from *A* to *M* is similar, besides the fact that it may occur upon a failure of the localization subsystem.

V. OUR PROPOSAL

Our approach addresses the problem of safety in the design of adaptive RTES. We introduce our process divided in two-step optimization to model and analyze an adaptive system. Since, the first consist mainly in modeling the behavior of the step and adaptation rules, the second step provide a result of analyzing as an output. More precisely, we rely on NCES formalism for the modeling and validating of NFPs properties during and after adaptation. The model validated at design time is used at runtime. More details for each step are mentioned below.

A. Modeling Adaptive Systems

At design time, additional information compared to non-adaptive system, has to be modelled (adaptation rules, variability, transitional modes). In adaptive systems, we model all alternatives and possible variations of the system elements. Moreover, our approach is based on the concept of mode which is a subset of system features: when the system is in a given mode, it provides this subset of features. Thus, we propose to represent the system operating modes and the conditions that trigger modes and limit changes. For instance, we specify using NCES the both operating modes of the *piloting system* (i.e., Manual mode and Automatic mode) by specifying the end-to-end scenarios. This complements the structural functional of the system cited in Fig.1. We need to build a model for the source mode and a model for the target mode. The source and target models should not include information about each other, or about the adaptation.

After identifying models, it is necessary to specify adaptation rules. These are conditions that should be respected during and after adaptation step. In this work, each condition *C* is modelled in his normal form *C* and negative form ¬*C* and should be linked to source and/or target models to fulfil adaptation rules and requirements. In addition, event signals are used between models of source and target to define transitional modes. Fig. 2 below illustrates the modeling of the *piloting system* through NCES formalism.

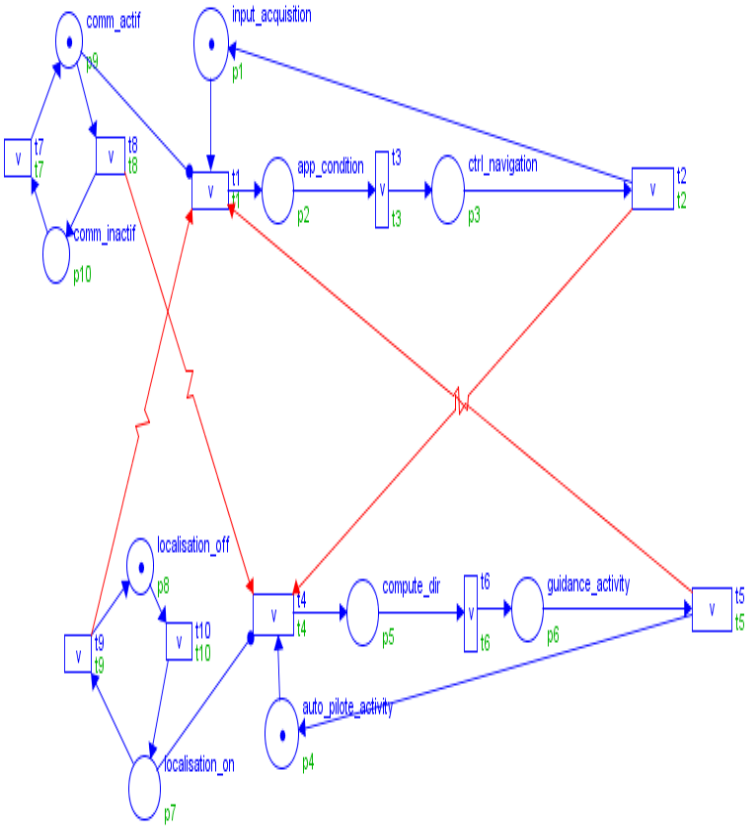

Fig 2. NCES Model.

In the Manuel mode: The initial place *input_acquisition* marked with an initial token corresponding to the launching of the run-time scenario. Places without any token at the beginning *app_cond* and *ctrl_navigation* depict the execution of the component *manuel navigation*. The condition of the availability of the communication system is modelled in two normal form (*comm_actif*) and negative form (*comm_inactif*). The Automatic mode is similar, triggered by an event signals (i.e., arc flow from t2 to t4). Event signals, that fire the initial transition of each mode, denote transitional mode with resecting to adaptation rules (condition).

In our approach, a task is considered as being possibly impacted by a mode switch if its execution flow depends on the value of the current mode. Thus, a mode switch must really occur when all the tasks being possibly impacted by this mode switch have completed their last execution, and none of those threads can restart until this adaptation is finished.

B. Verifying properties

Once the modeling step is realized, the second stage consists of analyzing the net in order to verify and validate the non-functional properties. As already mentioned, the advantage with NCES-based models is that offers an effective and optimal solution to make the verification process easier with a low complexity.

The verification process is to check the vivacity, boundedness, detect dead states and transitions using SESA tool in order to prove stability, consistency and the correctness of an adaptive system. The case study described before is verified and discussed below in order to prove the consistency and the correctness of the system behavior subject to different fault problem cases that can occur. The most important checked properties are (1) verify that all modes are achievable by verifying liveness of the net, (2) verify that during adaptation no deadlock will occur by verifying boundedness and (3) the generation of the reachability graph prove that the system state are finite, so stability can be proved.

VI. CONCLUSION

In this paper, we introduced a two-step process to model and checking adaptive RTES against their global properties. Our process focuses on modeling the behavioral aspect of functional and adaptive logic through NCES formalism. As source and target models are created, they are automatically checked using SESA tool. A key contribution of this approach is the ability to verify transitional properties which, previously, had not been well tackled. This approach enables developers to detect errors in the adaptive logic before implementation and deployment. We note the potential for improving model checking performance by combining our approach with existing techniques.

As future work, we will investigating strategies to combine our approach with others to further reduce the complexity of adaptive system model checking and we plan to automate as much as possible the implementation model.

REFERENCES

[1] J. Zhang, H. Goldsby, and Betty H. C. Cheng, “Modular erification of dynamically adaptive systems”. In AOSD, 2009, pp 161–172, 2009.

[2] M. Naija, J-M Bruel, and S. Ben Ahmed, “Towards a MARTE extension to address adaptation mechanisms”. In HASE, 2016, in press.

[3] M. Rausch and H.-M. Hanisch. Net condition/event systems with multiple condition outputs. In Emerging Technologies and Factory Automation, 1995. (ETFA), Proceedings INRIA/IEEE Symposium on, 1995, volume 1, pp 592–600 vol.1.

[4] Z.W. Li J.F. Zhang, M.Khalgui and O.Mosbahi. R-TNCES: a Novel Formalism for Reconfigurable Discrete Event Control Systems. IEEE, 2013.

[5] Valeriy Vyatkin. Modelling and verification of discrete control systems.

[6] M. Khalgui. “Nces-based modelling and ctl-based verification of reconfigurable embedded control systems”. Computers in Industry, 2010, 61(3): pp198 – 212.

[7] I-R. Quadri, S. Meftali, and J-L. Dekeyser. "A Model based design flow for Dynamic Reconfigurable FPGAs". International Journal of Reconfigurable Computing, 2009.

[8] M-L. Boukhanoufa, A. Radermacher, and F. Terrier. "Offline validation of real-time application constraints considering adaptation rules". In Proceedings of the 2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TRUSTCOM), 2011, pp 974–980.

[9] M-B. Said, N-B. Amor, Y-H. Kacem, M. Kerbo,euf and M. Abid. "A Model driven approach for the development of fine-grain self-adaptive multitask and networked RTE systems". In IEEE 23rd International WETICE Conference, 2014.

[10] Andres J. Ramirez and Betty H. C. Cheng. “Verifying and analyzing adaptive logic through uml state models”. In ICST, 2008, pp 529–532.

[11] T. Murata, “Petri nets: Properties, analysis and applications”. In Proceedings of the IEEE, vol. 77, no. 4, pp. 541-580, 1989.