Petri net and UML for the V&V of ITASAT Satellite Prototype

Eduardo Correia, Paulo Claudino Véras, Osamu Saotome, Emilia Villani

Instituto Tecnológico de Aeronáutica (ITA)
Pça. Marechal Eduardo Gomes, 50 – Vila das Acácias – 12228-900 – São José dos Campos – SP – Brazil
{ecorreia,pcv,osaotome,evillani}@ita.br

Abstract. This paper approaches the process of validating and verifying the embedded control system of a satellite prototype. The focus of this approach is on the integration of UML and Petri nets, aiming at the formal verification of the system requirements. The approach is composed of: (1) simulation of the UML model in a CASE tool; (2) simulation of the Petri net model in a Petri net simulator; and (3) formal verification of the Petri net model. The Petri net model is obtained from the UML statecharts by applying a set of rules. The formal verification is done by associating the system requirements to Petri net properties that are proved using Petri net analysis methods.

1. Introduction

This paper presents a systematic approach to validate and verify (V&V) real-time embedded systems. Its main contribution is on the merging UML and formal modeling techniques. By this way, the gap between the ‘state of art’ (the academic world) and the ‘state of practice’ (the industrial reality) in V&V is shortened.

This work is part of ITASAT Program, which aims at the development of satellites in a university environment in partnership with INPE, the Brazilian institute for spatial research.

This paper is organized as follows. Section 2 describes the ITASAT satellite prototype and presents the proposed approach. Section 3 presents some conclusions and discusses future steps.

2. The Proposed Approach

2.1. The ITASAT Satellite Prototype

One of the key research points of ITASAT Program is the design of on-board computers. In this context, one of the purposes of the first ITASAT Satellite Prototype is to elaborate and test V&V techniques. The system is composed of a compressed air bearing that holds a platform containing the on-board computer, a momentum wheel and a gyroscope. The momentum wheel is moved by a DC-servomotor and uses the principle of the conservation of the angular momentum to vary the platform attitude. The sensor used is a gyro that measures the variation of the platform angle. The embedded system of the platform must perform the control law for attitude control, which is a PID (Proportional Integrate Derivative) controller. It must also perform a
number of functions related to data handling and housekeeping, such as interacting with the ground station, sending and receiving data, etc.

2.2. The Proposed Approach

The proposed approach for the system V&V is composed of three different activities. The first one is the simulation of the UML embedded system model in a CASE tool. The second is the simulation of the Petri net model in a Petri net simulator. The third one is the formal verification of requirements in the Petri net model. The main steps of the approach used for V&V are presented in Figure 1.

![Figure 1. Main steps of the V&V process.](image)

The CASE tool used in Steps 2-4 is Rhapsody [Telelogic AB 2007], while the Petri net simulator used in Steps 5-8 is HPSim [Anschuetz 2007].

Step 1 – Specification of System Requirements

Basically, the on board computer must receive the attitude reference from the ground station, determine the satellite attitude using the signal provided by gyro, calculate the control signal and provide it to the momentum wheel. It also receive information through the sensor about the constantly disturb of the system, providing the constant attitude control calculation.

Steps 2-4 – UML Modeling and Simulation in Rhapsody

The embedded system architecture of the on-board computer is composed by the classes OperatorReference, which has the function of station ground, the LawControl, which provides the formula to control the system, UpdatePosition, which gets the sensor values to the control function, and VoltageControl, which provides the actuator command of the system. The components that interact with these classes are: Sensor, MomentumWheel and GroundStation.

The onboard computer (OBC) commands acquisition of the actual position, which is transmitted from the Sensor to the UpdatePosition. Then, the ControlLaw calculates the new value of the actuation, according to the current and reference positions, and sends the new value to the VoltageControl, whose function is to calculate the necessary voltage of the motor that controls the momentum.

Once the UML model was elaborated, it is analyzed using simulation techniques available at the CASE tool.
Step 5-8 – From Statecharts to Petri nets

The proposed approach to obtain the Petri net model from the UML model is based on the conversion of the statecharts. The statecharts of every class are designed separately from each other. The communication among them is accomplished with the exchange of messages and/or data. This communication, however, is not explicit in the statecharts, as well as the resulting behavior of the composition of the various statecharts. In this sense, the conversion of all of the statecharts to a single Petri net model allows the observation of the behavior of the whole system in a unique model. The parallelism and hierarchy of statecharts were not used in the UML model, therefore, these features were not considered in the rules described below.

Briefly, the following rules are defined for converting the UML statecharts to a Petri net model:

1) A Petri net place is associated to each state of the UML statecharts.
2) A transition between two states in the statechart is converted in a sequence of arc-transition-arc in the Petri net model.
3) The choice point of the statechart represents a conflict situation, hence it is modeled in the Petri nets by a place with two conflict output transitions. If there is N choice points in sequence, it is enough to insert (N + 1) conflict transitions in the Petri net.
4) The initial state in the UML is the first state that the diagram reaches. It is represented in the Petri net by an initial token in the corresponding place.
5) Finally, the exchange of messages between two capsules in the UML model is modeled in the Petri net by an interface place, which is simultaneously an output place of a transition of the capsule model that sends the message and an input place of a transition of the capsule model that receives the message.

Due to the possibility of inserting source code in the statechart transitions, it is possible to execute different pieces of code in each firing of some transition. Therefore, each piece must be associated to a different transition, where these transitions are in conflict. Figure 2 presents an example of the conversion of the ControlLaw statechart to Petri net. This conversion from statechart to Petri net is made in a manual manner.

The integrated Petri net of the system is composed of the on board computer and 3 external components, translating in a very large model, consequently, it is not presented in this paper.

![Figure 2. From statechart to Petri net model.](image)

Once the Petri net of the system is built, the requirements must be converted into Petri net properties. Examples of requirements and the equivalent Petri net property are:
1) Determine the satellite attitude using the signal provided by sensor at each cycle - the sequence of transition firings T10-T11 must be performed at each cycle (these transitions represent the event of determining the satellite attitude);
2) Calculate the control signal and provide it to the momentum wheel at each cycle - the sequence T13-T15-T16-T7-T6-T23-T8-T24 must be performed at each cycle (these transitions represent the performance of the cycle of calculating the control signal);
3) Change the reference of the system when requested by the user – perform the sequence of transitions firings T2-T1-T5-T4-T12-T14-T5 when T2 is fired (these transitions represent the action of changing the reference of the system).

**Step 9-10 – Formal Verification**

These final steps consist of the verification of good properties, such as boundness, safeness and liveness, as well as by the verification of the system requirements, which can be performed by building the reachability graph of the Petri net. This graph contains any marking that can be reached from the Petri net initial marking. It also illustrates all the possible firings at a certain marking, in other words, it can be seen all the possible scenarios of execution of the model.

**3. Conclusions**

The approach of this paper is composed of three main activities, whereas the first one consists of simulating the UML control system model in a CASE tool. The second activity consists of translating the UML statechart to Petri net and simulating the obtained model. The last activity is a formal verification of requirements using the Petri net model.

Next steps are the inclusion of time requirements in the approach, using Timed Petri net, and development of analysis strategies for the formal verification of requirements.

**4. Acknowledgements**

The authors would like to thank the financial support of the following entities: AEB, FAPESP, CNPq, CAPES and FINEP.

**5. References**