A classification of real-time specifications complexity

Vieri del Bianco, Luigi Lavazza, Marco Mauri

CEFRIEL and Politecnico di Milano

1. Introduction

The authors of this paper are currently involved in the ITEA DESS project (Software Development Process for Real-Time Embedded Software Systems), whose goal is to develop a methodology for developing real-time software systems.

In the DESS project we are mainly involved in defining methods for the representation of requirements of real-time software, and in the definition of methods for analyzing real-time specifications. Figure 1 illustrates the DESS approach to requirements specification and analysis. The DESS environment allows analysts to model real-time systems using an extension of UML, which can support the translation of models into formal notations. In this way we retain the expressiveness and ease of use of UML [6], while gaining the power to use formal methods, which can be employed for verifying properties of specifications, for deriving test cases, etc. More on DESS an be found in [2].

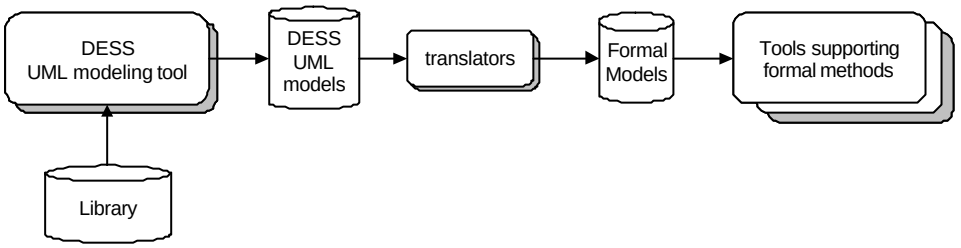

Figure 1: The DESS environment for requirements specification and verification.

Up to now our method has been applied only to sample "toy" examples, as the validation phase of DESS – which involves the execution of realistic case studies– is just starting. Nevertheless, the application of our method to sample systems has revealed that the difficulty of writing satisfactory specifications of real-time systems may vary widely. More precisely, in a formal context the effort devoted to the specification phase is distributed in several activities:

- Modeling (by means of UML, or a suitable extension).
- Specification of the properties that the specified system should own.
- Verification of the properties.

The third point is generally not very difficult, as long as we are able to translate UML models into formal notations, and tools are available to analyze the resulting formal specifications.

The modeling phase may be more difficult and time-consuming, but –according to our experience– once you have selected a suitable modeling language (e.g., UML equipped with proper stereotypes) and you have well understood the system, there is no major difficulty in building the models. The complexity of the real-time systems does not affect much the modeling effort. In other words, real-time systems have all more or less the same complexity (the size of the systems will mo re likely affect the modeling effort).

The activity with the most relevant variations in difficulty (and henceforth in effort) is by far the specification of properties involving the behavior in time of the system.

In order to illustrate this point, we briefly discuss three examples, one of which is the case study proposed for the workshop [1].

2. Examples

2.1 The Generalized Railroad Crossing (GRC)

The GRC problem [3] is a well-known benchmark for real-time analysis methods and notations. It is a system containing a set of tracks, a railroad crossing and a gate. The system must take into consideration the "physical" properties of its elements, i.e., the maximum and minimum train speed, the position of sensors with respect to the crossing, and the time required to close or open the gate.

The complete model of the system written with the DESS extensions of UML is reported in [4]. It is not reported here for space reasons. However, the model is relatively simple, and it only requires extensions of UML concerning the explicit representation of time.

The desired properties of the model are:

- Safety: the gate must be closed whenever a train is crossing.
- Utility: the gate must be open whenever this is compatible with safety and the physical constraints (e.g. the gate takes some time to open).

These properties have a few relevant characteristics:

- They involve several items (the gate, the crossing, the trains).
- They concern a set of object's states at the same time (e.g., the state of the gate at time *t* must be closed if the state of the crossing is busy at the same time *t*).

The conditions above made the specification of the desired properties relatively simple. In fact it was possible to define the state diagrams in a way that allows the representation of properties by means of OCL statements involving just states. Note that the states involved in the OCL conditions were not needed to express the behavior of the system: they were added for the specific purpose of property specification.

In conclusion we can consider the GRC a not very complex example, since properties that concern its behavior in time can be expressed by means of OCL, a language that does not allow users to make reference to time directly. In fact time conditions are implicitly captured by means of states. For details concerning the mentioned specifications see [4].

2.2 The Car Speed Regulator

The car speed regulator system, schematically represented in Figure 2, is thoroughly described in [1].

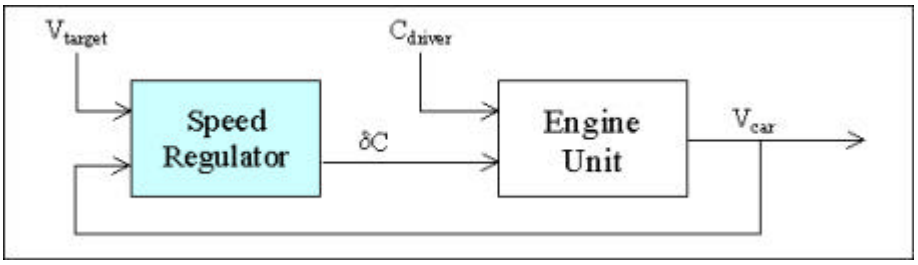

Figure 2: The schema of the car speed regulator .

It requires a software system (the regulator) to compute the value of the torque change that will make the speed of a car approach a predetermined value. The system is equipped with a sensor that provides the knowledge of the actual speed. The behavior of the system is periodic: the actual speed is read at a fixed frequency, and the system has to compute the change in torque before the next read.

Several constraints are posed on the time allowed to stop the regulator upon given events, as well as putting it on standby and resuming upon other events.

The model of the system, written using the DESS extensions of UML, is reported in [5].

The required properties of the system are of two kinds:

- The reaction to a given event is performed within a given time. Reactions involve simple computations and state transitions.

- The system has a periodic behavior, hence in every period the tasks assigned to the system must be completed. In this particular case the task is fixed and is unique (to compute δC), thus the property can be considered quite simple.

In practice the first kind of properties can be embedded in the model by specifying that the permanence in some states (corresponding to some computation or transition) is bounded to some time interval.

The second property can be expressed in several ways. However, given that the system is periodic and can evolve in each period in a very little number of ways, it is even possible to enumerate all the possible evolutions (e.g., by means of sequence diagrams) and express constraints on their duration (again, using sequence diagrams).

2.3 A DESS sample problem

This application example (see Figure 3) is the implementation of a control loop of some real time process. It is constituted of two "activities" that work concurrently on a mono-processor architecture: a Filter and a Corrector. The Filter is periodically triggered at a TA period, and the Corrector is periodically triggered at a TB period. These two activities share a common one slot buffer .Triggering signal can be given by some external timers. The buffer read access by the Corrector and the buffer write access by the Filter are atomic: typically they consist in a main memory word access which is, thanks to the hardware, intrinsically atomic in a mono processor architecture.

The situation can be generalized (as shown in Figure 3) considering several Filters, instead of just one.

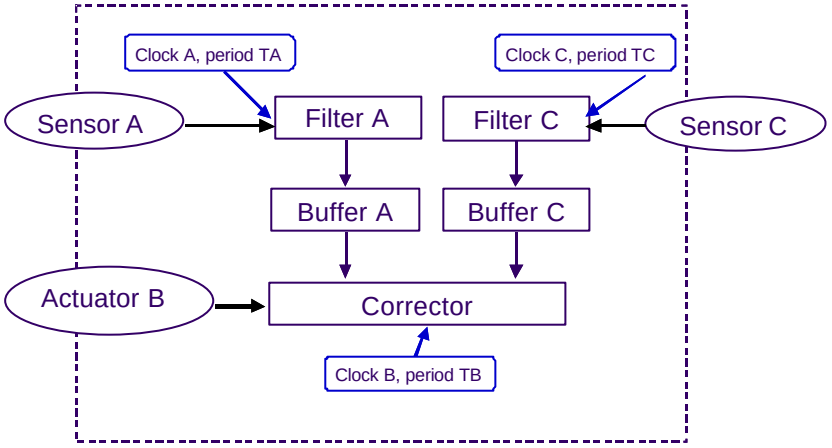

Figure 3: The schema of the DESS example.

The behavior of the system is subject to several constraints. Among others:

- each data output sample must be emitted no later than 3 t.u and no before than 4. t.u. from the last output sample;
- every time an output sample B is generated, this sample must be computed from A and C sample which are not separated by a delay greater than 5 t.u.;

where t.u. stands for time units. It is not relevant how long is a t.u.

It is easy to see that the properties required in this case are more difficult to express than the former ones:

- There is no unique clock which governs the behavior of the system and which can provide a reference for the properties.
- We have sets of constraints which must all hold together. Several of these constraints refer to different objects' states considered at different times.

It is quite clear that OCL and state diagrams are not sufficient to model this system's constraints. Maybe it could be possible to add an "observer" object which reacts to every event in the rest of the system, and summarizes in its state all the states of the other objects: then constraints could be expressed only with respect to this object. However such an object would probably be so complex (its set of states being the product of the states of all the other objects) that even though we were able to express properties concerning its behavior, we would hardly be sure that such properties are expressed correctly.

As a consequence, in this case we need a more powerful linguistic tool to express constraints. We are currently writing such constraints using timed temporal logic TCTL [7]. Note that modeling this system, e.g. by means of the timed automata of Kronos [8] is not difficult. By the way, Kronos provides model-checking facilities that will allows the verification of the model's properties.

3. Considerations and conclusions

In this paper we have considered three sample real-time systems. Although these systems are similar with respect to size, structure, and "modeling complexity", the effort required to express the models' properties was very different in the three cases.

As discussed above, this is due to the "complexity" of the requirements concerning the real-time behavior of the three systems. Here we have no pretensions to provide a formal classification scheme for the complexity of real-time behaviors, or for the properties that express such behaviors. Nevertheless, we can make some observations:

- The most simple system of the lot, the car speed regulator, is characterized by periodic behavior, with constraints concerning only the duration of single state transitions. In this case most properties directly correspond to a model item (typically the specification of a transition). The only other relevant property concerns the completion of a computation in a period. This is relatively easy to express both with formal methods (e.g., non-zenoness of a timed automata), and informally (e.g., as a set of sequence diagrams).
- The most complex system is characterized by several clocks, and constraints that involve several states of different objects at different points in time. In such a case, it is difficult to express properties without resorting to some sort of temporal logic.
- Interestingly, we found an intermediate case, the GRC, whose desired properties can be specified in terms of sets of states considered at the same time, which can be satisfactorily specified by means of state diagrams and OCL statements.

These findings confirm that the DESS approach described in section 1 is correct, as long as it is very flexible: systems which are complex with respect to the classification above will be translated and verified by means of formal methods, while simpler cases will be developed without going through the formal phase.

Acknowledgments

The work described here was partly supported by MURST funding of ITEA project DESS. Work by Vieri Del Bianco was partly supported by CiaoLab (http://www.ciaolab.com).

References

[1] Case Study: a Car Speed Regulator, http://www-leti.cea.fr/leti/UK/Pages/Tech_info/These/sivoes2001.htm

[2] L. Lavazza, "An introduction to the DESS approach to the specification of real-time software", CEFRIEL report, April 2001.

[3] Heitmayer C.L., Jeffords R.D., Labaw B.G., Comparing different approaches for Specifying and verifying Real-Time Systems, In *Proc. 10th IEEE Workshop on Real-Time Operating Systems and Software* (New York May 1993), 122-129.

[4] L. Lavazza, G. Quaroni, M. Venturelli, *Combining UML and formal notations for modelling real-time systems*, Technical report CEFRIEL, November 2000.

[5] V. del Bianco, L. Lavazza, M. Mauri, An application of the DESS modeling approach: The Car Speed Regulator, CEFRIEL Technical Report, April 2001.

[6] *OMG Unified Modeling Language Specification*. Version 1.3, June 1999. http://www.omg.org.

[7] R. Alur, C. Courcoubetis, and D. L. Dill. Model checking in dense real-time. *Information and Computation*, 104(1):2{34, 1993.

[8] Yovine S. Kronos: A verification tool for real-time systems. In *Springer International Journal of Software Tools for Technology Transfer*, Vol. 1, N. 1/2, October 1997.