Analysis of Real Time Systems through the ORIS Tool

L.Sassoli, E.Vicario

Dipartimento Sistemi e Informatica - Università di Firenze, Italia
E-mail {sassoli,vicario}@dsi.unifi.it

***Abstract*— This paper gives an overview of the Oris tool. Oris comprises a rich set of modules for building, simulating, analyzing and validating real-time systems described through various TPN formalisms. After an introduction of the Oris framework, we describe a number of Oris plug-ins that have been implemented to support some recently-developed analysis techniques.**

I. Introduction

Oris comprises a rich set of modules for building, simulating, analyzing and validating real-time systems described through various TPN formalisms [1] [2] [3]. In particular, the semantics of Preemptive Time Petri Nets [2] and Stochastic Preemptive Time Petri Nets [3] gives Oris the expressive power needed to deal with complex schedulability problems. As a result, Oris can be used to model complex tasking sets which include different release policies such as recurring, sporadic and one shot. It permits modelling of inter-task dependencies due to the timing of releases, to mutual exclusion on shared resources and dataflow precedence relations. It also enables modelling of internal sequencing of tasks and nondeterministic computation times. This applies to both the case of single and multiple processors systems.

II. A brief Description of the Toolset

In the current version, Oris has a *plug-in* architecture which defines a number of rules and provides a set of extension points that make trivial the integration of new modules (plug-ins) in the environment. Fig. 1 shows a screen of the tool at work, while the dataflow diagram of Figure 2 schematizes the typical utilization of Oris when used in validation. Each bubble represents a data transformation process, labeled with the name of the corresponding Oris module. The figure also shows the data structures that are generated at different stages.

A typical design cycle starts with the user specifying a model. Usually, the model is directly built through the TPN Editor, but it can also be automatically generated if the system under analysis belongs to a specific application context. For instance, Timelines Editor allows specification of a tasking set through the timeline formalism and its translation to the corresponding model. Once the model has been built, the analyst can either simulate or analyze it. Simulation is in two forms: the first consists in animating the model; the second in performing batch simulation. The analysis is carried out through the TPN Analyzer, which is able to generate a graph representing the reachability relation among state classes for different Time Petri Net extensions.

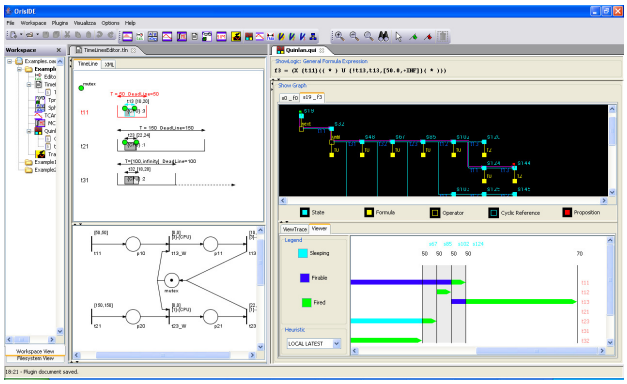

Fig. 1. A screen of Oris employed in a schedulability verification problem

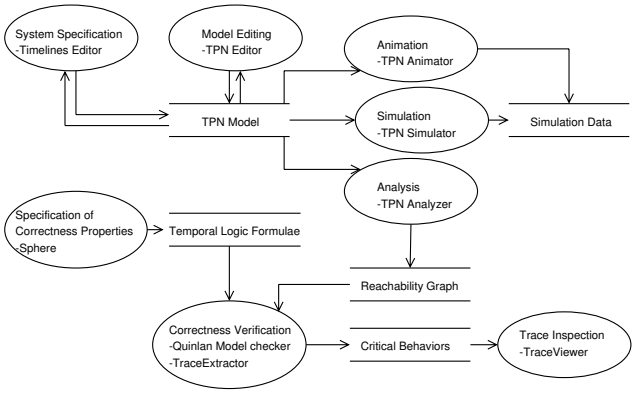

Fig. 2. A dataflow diagram showing a typical usage of Oris modules in different stages of a verification problem.

The enumerated state space contains all information pertaining to model behavior. However, in order to ease inspection of the reachability graph, appropriate methods are needed through which the user can, for instance, identify all the critical traces with respect to requirements pertaining to the logical sequencing or to the quantitative timing of events. To this end, the Oris environment provides a model checking tool which can perform verification with respect to correctness clauses specified in terms of temporal logic. Since the expression of such clauses may be counterintuitive for the designer, we have developed an interactive graphic tool, named Sphere, which helps building temporal logic formulae. Alternatively, automatic extraction of any symbolic execution between two

Third International Conference on the Quantitative Evaluation of Systems (QEST'06)
0-7695-2665-9/06 $20.00

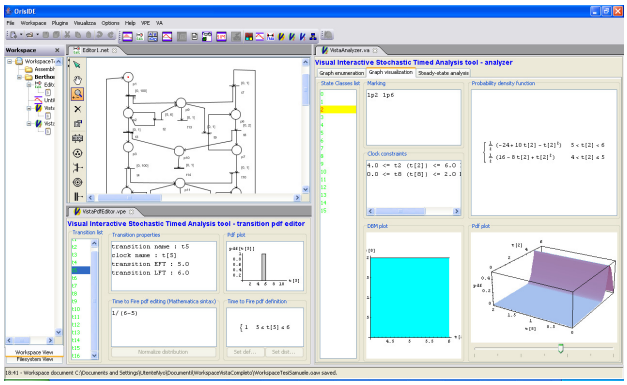

Fig. 3. A screen of Oris-VISTA modules at work

events is made possible through the TraceExtractor module. Finally, the TraceViewer module allows the inspection of the dense variety of timings that can be associated to any symbolic execution sequence, in order to understand the reasons that may lead to specific behaviors.

III. New capabilities: analysis of sTPN models

Time Petri Nets do not allow the characterization of feasible behaviors with a measure of probability, an essential step towards dependability and performance evaluation. In order to overcome this limitation, in [4] stochastic Time Petri Nets (sTPNs) have been introduced. They extend the formalism of Time Petri Nets by associating the static firing interval of each transition with a (dense) probability density function. The analysis of sTPN models requires the derivation of a density function which characterizes the probability of individual timings comprised within the boundaries of a time domain. This leads to the introduction of stochastic state classes, which extend state classes and their reachability relation by enriching dense firing domains with a state probability density function. The resulting stochastic state class graph can be used for the verification of correctness properties pertaining to the logical sequencing and to the qualitative timing of events. In addition, since it is also a stochastic process, it can be analyzed to derive performance measures.

Fig. 3 shows a screen of the new set of Oris modules, called VISTA (Visual Interactive Stochastic Timed Analyzer), which has been implemented to support the theory introduced in [4]. Fig. 4 reports a dataflow diagram describing the three different phases in the analysis of sTPNs through ORIS. Using the VISTA-Editor, a TPN is extended into a sTPN by associating the static firing interval of each transition with a probability density function. Then, starting from the state class graph of the corresponding TPN, the VISTA-Analyzer enumerates the stochastic state class graph associated to the sTPN model, which is the basic step to enable performance evaluation. Finally, the VISTA-TraceAnalyzer can be used to estimate the probability distribution associated to the min-max duration for any execution sequence identified as a path in the stochastic

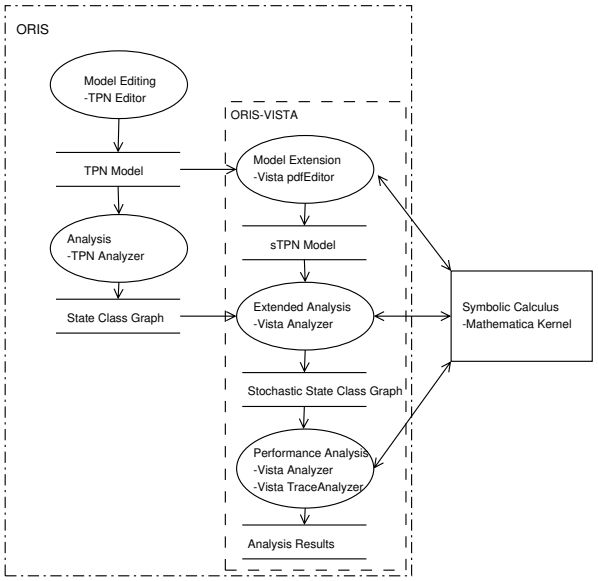

Fig. 4. A diagram representing the components required for the definition and the analysis of sTPN models.

state class graph. Fig. 4 also shows a difference among VISTA and other components of Oris. While most of Oris modules have a Java user interface calling C++ implementations to perform the required computations, the set of VISTA modules call the Mathematica Kernel [5] to perform the symbolic calculus required in the analysis of sTPN proposed in [4]. The bridge between VISTA and Mathematica is made of JLink libraries [6]. These allow the instantiation of the Mathematica Kernel as a Singleton object [7], whose methods can be invoked to perform the required computations.

IV. Conclusions

We have briefly described the Oris environment for specifying, simulating, analyzing and validating reactive real-time systems. Being a research tool, the Oris framework undergoes continuous development. Oris can be obtained by contacting the authors through the website http://www.stlab.dsi.unifi.it/oris/index.html.

References

[1] E. Vicario, "Static analysis and dynamic steering of time dependent systems using time petri nets," *IEEE Trans. on Soft. Eng.*, August 2001.

[2] G.Bucci, A. Fedeli, L. Sassoli, and E. Vicario, "Timed state space analysis of real time preemptive systems," *IEEE Trans.on Soft.Eng.*, vol. 30, no. 2, pp. 97–111, February 2004.

[3] G. Bucci, L. Sassoli, and E. Vicario, "Correctness verification and performance analysis of real time systems using stochastic preemptive time petri nets," *IEEE Trans. on Soft. Eng.*, no. 11, pp. 913–927, November 2005.

[4] G. Bucci, R.Piovosi, L. Sassoli, and E. Vicario, "Introducing probability within state class analysis of dense time dependent systems," *Proc. of the 2st Int. Conf. on the Quant. Evaluation of Sys.(QEST)*, September 2005.

[5] W. Research, "Mathemathica 5.1," *www.wolfram.com*.

[6] W. C. Technologies, "Java toolkit: J/link," *http://www.wolfram.com/solutions/mathlink/jlink/*.

[7] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design patterns," *Addison-Wesley*.