TAXYS : a Tool for the Development and Verification of Real-Time Embedded Systems[*]

Etienne CLOSSE[1], Michel POIZE[1], Jacques PULOU[1], Joseph SIFAKIS[2],
Patrick VENIER[1], Daniel Weil[1], and Sergio YOVINE[2]

[1] France Telecom R&D, 28 chemin du Vieux Chêne, 38243 Meylan cedex, France,
`firstname.lastname@rd.francetelecom.fr`,
[2] UMR VERIMAG, 2 rue Vignate, 38610 Gières, France,
`firstname.lastname@imag.fr`

Abstract. The paper presents a prototype of the TAXYS tool developed within a collaboration between France Telecom R&D and VERIMAG. The connection of the SAXO-RT ESTEREL compiler and of the KRONOS model-checker, together with on-the-fly techniques, brings up the possibility of verifying quantitative timing constraints on real industrial telecommunication systems, such as a GSM radio link and a phone prototype developed by France Telecom. In addition, TAXYS offers a non-ambiguous and user friendly formalism for specifying quantitative timing constraints as well as high-level diagnostic functionalities.

Paper category : B (tool presentation)

Keywords : ESTEREL, Synchronous Languages, Real-Time, Embedded Systems, Model-Checking, Timed Automata.

TAXYS : a Tool for the Development and Verification of Real-Time Embedded Systems*

Etienne CLOSSE[1], Michel POIZE[1], Jacques PULOU[1], Joseph SIFAKIS[2], Patrick VENIER[1], Daniel Weil[1], and Sergio YOVINE[2]

[1] France Telecom R&D, 28 chemin du Vieux Chêne, 38243 Meylan cedex, France,
`firstname.lastname@rd.francetelecom.fr`,
[2] UMR VERIMAG, 2 rue Vignate, 38610 Gières, France,
`firstname.lastname@imag.fr`

The correct behavior of real-time applications depends not only on the correctness of the results of computations but also on the times at which these results are produced. As a matter of fact, violations of real-time constraints in embedded systems are the most difficult errors to detect, because they are extremely sensitive both to the patterns of external events stimulating the system and to the timing behavior of the system itself. Clearly, the development of real-time systems requires rigorous methods and tools to reduce development costs and "time-to-market" while guaranteeing the quality of the produced code (in particular, respect of the temporal constraints).

The above requirements motivated the development of the TAXYS tool, dedicated to the design and validation of real-time telecommunications software. One of the major goal of the TAXYS tool is to produce a formal model that captures the temporal behavior of the whole application which is composed of the embedded computer and its external environment. For this purpose we use the formal model of timed automata [2]. The choice of this model allows the use of results, algorithms and tools available. Here, we use the KRONOS model checker [4] for model analysis.

From the source code of the application, an ESTEREL program annotated with temporal constraints, the TAXYS tool produces on one hand a sequential executable code and on the other hand a timed model of the application. This model is again composed with a timed model of the external environment in order to obtain a global model which is statically analyzed to validate timing constraints. This validation should notably shorten design time by limiting tedious test and simulation sessions.

1 Taxys

The objective of the TAXYS project is to propose a framework for *developing real-time embedded code and verifying its correct behavior with respect to quantitative timing constraints.*

We use ESTEREL [3] as development language of the application. This language provides powerful constructs for management of parallelism and exceptions. It has rigorously defined semantics. ESTEREL programs run in a single

---

thread on a single processor with a non-preemptive interrupt routine and can refer to external data and routines written in C for complex (numerical) computations. Thus, the application is decomposed into a control part, written in Esterel and a functional part written in C, and it is compiled with the Esterel compiler Saxo-rt [5].

The use of synchronous languages for the development of real-time reactive applications relies on a "synchrony assumption" meaning that the application reacts infinitely fast with respect to its environment. This assumption, very convenient in practice, must be validated for a given implementation on a target machine. In practice, validating the synchrony assumption amounts to show that the environment does not take too much lead over the application. This requires the use of a "realistic" synchrony assumption strongly depending on the application, on the speed of the machine and on its interactions with the environment. To interface the real-time system with its environment, we use an external event-handler $\mathcal{H}$, generated by Saxo-rt from an ad-hoc specification [1], and which precisely takes into account the way external events are captured by the interrupt mechanisms and sent to the application.

The behavior of such systems can be modelled by the composition of 3 systems represented as automata : the application automaton $\mathcal{A}$, the external event handler $\mathcal{H}$, which abstracts the behavior of the interrupt routine and buffers external events before they are taken into account by the next synchronous reaction, and the environment model $\mathcal{E}$ which specifies the scenarios in which the application must run [1].

The environment of a real-time embedded system can exhibit different behaviors that must be captured by some non-deterministic model. As Esterel programs are deterministic, we add a non-deterministic instruction *npause* to the Esterel language. The environment can thus be written in the same language as the application. The timing constraints are specified directly by pragmas in the Esterel code of $\mathcal{A}$ and $\mathcal{E}$.

Taxys design flow is shown in Fig. 1. Saxo-rt generates three C-modules which compute $\mathcal{A}$, $\mathcal{H}$ and $\mathcal{E}$ transition functions : the model of the application contains the *embedded code itself*. Kronos [4] explores the system states space by composing on-the-fly $\mathcal{A}$, $\mathcal{H}$ and $\mathcal{E}$. Thus, no intermediate state explosion occurs before composition and only reachable states are computed. If any timing constraint is violated, a trace leading to this error is generated. This trace is then re-executed step by step on the Saxo-rt graphical debugger to provide to the user more precise diagnostics.

2 Timing analysis

We make the following assumption on the temporal behavior of the application: execution time is spent in the functional part to compute C-functions which have been previously instrumented by profiling. The Esterel code is annotated with this information. This hypothesis is true for many reactive applications if the embedded code has been compiled efficiently [5]. We then specify two kinds
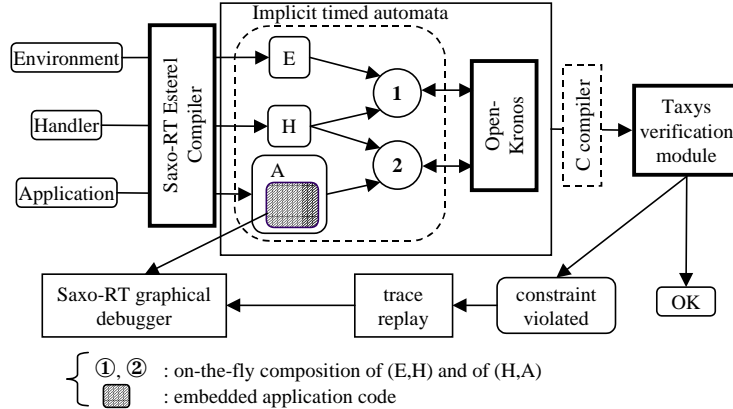
Fig. 1. Taxys Design Flow

of real-time constraints : *throughput* and *deadline* constraints. A throughput constraint is a global constraint and expresses the fact that the system reacts fast enough for a given environment model. The violation of a throughput constraint corresponds to an overflow of $\mathcal{H}$. A deadline constraint is "local" and expresses for example, a maximum delay between a given input and a given output of the system.

This approach is illustrated by the toy example "pulse" on Fig. 2, which is composed of two parallel tasks. The first, triggered by input A, calls filter F. The second, triggered by B, computes some correction G on an actuator using result of function F. F (resp. G) consumes between $Fmin$ (resp. $Gmin$) and $Fmax$ (resp. $Gmax$) CPU time. The buffer size of the external event handler $\mathcal{H}$ is 1.

The *throughput constraint* is specified by the environment model written in timed Esterel (Fig. 5). It is composed of two independent periodic tasks, the first one strictly periodic with a period T_A and the second one with a period T_B jittered by an interval $[0, \varepsilon]$, for some constant ε.

There are two *deadlines constraints* on function F and G (Fig. 3) : $(\mathcal{D}_1)$ F must terminate $d1$ time units after arrival of event A and $(\mathcal{D}_2)$ G must compute value of actuator with data not older than $d2$ time units i.e., G terminates at most $d2$ time units after the arrival of the last event A which was consumed by function F. The annotated application code is given on Fig. 4 : $\mathcal{D}_1$ is specified by the pragma $0 < clock(lastA) < d1$, and $\mathcal{D}_2$ by the two pragmas $Y = clock(lastA)$ (which starts a new clock each time F is executed), and $0 < Y < d2$.

3   Experimental results

We used Taxys for verifying the Esterel code for the communication mode of a GSM terminal developed by Alcatel (815 Esterel lines and 48000 C lines).
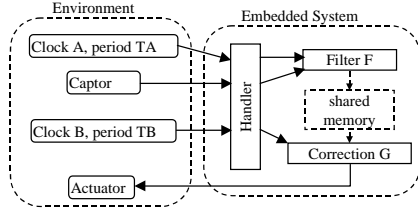
Fig. 2. The "pulse" example

Fig. 3. Deadline constraints

```
loop
 await A ;%{# Y=clock(last A) %}
 call F();%{# Fmin<CPU<Fmax %}
 %{# 0<clock(last A)<d1 %}
 end loop
||
 loop
 await B ;
 call G();%{# Gmin<CPU<Gmax %}
 %{# 0<Y <d2 %}
 end loop
```
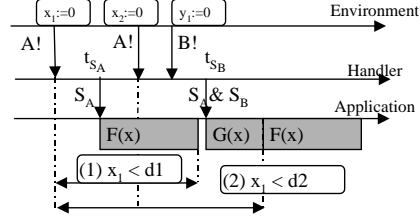
Fig. 4. Application code

```
loop
 npause;%{# TA<ca< TA; ca:=0}
 emit A;
 end loop
||
 loop
 npause;%{# TB<cb<TB+ε;cb:=0}
 emit B;
 end loop
```

Fig. 5. Environment code

We found 4 scenarios leading to deadline violations caused by a wrong scheduling between two C-functions [1].

We present here results obtained on a digital phone prototype carrying simultaneously voice and data produced by a graphic tablet, implemented on a 32 MIPS Digital Signal Processor. Audio data are processed at 8kHz and their processing consumes 3900 CPU cycles over the 4000 CPU cycles available every 125μs. Graphic tablet data are compressed by a vectorization algorithm which consumes sporadically between 15000 and 20000 CPU cycles. 6 experiments were carried out with the *same* ESTEREL code for the application but with different environment models and handler buffer sizes. $\mathbf{ISDN_1}$ and $\mathbf{ISDN_2}$ with an environment model composed of two strictly periodic and independent tasks (the first carrying audio data at 8kHz and the second the graphic tablet data at 100Hz). $\mathbf{ISDN_3}$ and $\mathbf{ISDN_4}$ with the second task being aperiodic and emitting bursts at rates varying in a non-deterministic manner between 25 and 100Hz. $\mathbf{ISDN_5}$ and $\mathbf{ISDN_6}$ with a third additional periodic task modelling switching between several audio modes. In all cases, the application $\mathcal{A}$ consists of 3000 C lines and 258 ESTEREL lines, and the environment $\mathcal{E}$ of 120 ESTEREL lines.

Results presented in table 3 show that a buffer size of at least 6 is necessary for absorbing the sporadic task. We observe that the number of symbolic states explored by KRONOS increases exponentially with the "degree" of non-determinism of the environment. Therefore, to cope with state explosion due to

environment non-determinism, it is necessary to find appropriate environment model approximations preserving the verified properties.

Table 1. TAXYS experimental results

Name	Buff. size	Symb. states	Verif. time	Diagnostic
ISDN$_1$	5	2 200	1.27 s	buffer overflow
ISDN$_2$	6	10 849	5 s	OK
ISDN$_3$	5	15 894	6.29 s	buffer overflow
ISDN$_4$	6	633 472	10 mn 47 s	OK
ISDN$_5$	5	22 695	13.6 s	buffer overflow
ISDN$_6$	6	$> 10^7$	?	aborted

4 Conclusion

We have presented an original approach for specifying, designing and validating real-time embedded systems. This approach is implemented in an entirely automated tool applicable to industrial size examples. Specifications are written in a user friendly and compositional formalism which does not require from the user any knowledge about timed automata or temporal logic. Its limitations are mainly those of model-checking techniques. Any advance in these techniques can be taken into account, transparently for the user. Furthermore, because the embedded code is effectively executed during validation, the validation is trustworthy and is therefore particularly suited to safety critical applications.

References

1. V. Bertin, M. Poize, J. Pulou, J. Sifakis, *Towards Validated Real-Time Software*, 12th Euromicro Conference on Real-Time Systems, Stockholm, Sweden, June 2000
2. R. Alur, D. Dill, *A theory of timed automata*, Theoretical Computer Science, 126:183–235, 1994. Elsevier.
3. G. Berry, G. Gonthier, *The* ESTEREL *Synchronous Programming Language : Design, Semantics, Implementation*, Science of Computer Programming, vol. 19-2, pp. 87-152, 1992.
4. C. Daws, A. Olivero, S. Tripakis and S.Yovine. The tool KRONOS. In Hybrid Systems III, Verification and Control, Lecture Notes in Computer Science 1066, Springer-Verlag, 1996.
5. D. Weil, V. Bertin, E. Closse, M. Poize, P. Venier, J. Pulou, *Efficient Compilation of* ESTEREL *for Real-Time Embedded Systems*, Proceeding of CASES'2000, pp. 2-8, San Jose, November 2000.