Proceedings of the 2003 Winter Simulation Conference
S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.

FROM TIMED AUTOMATA TO DEVS MODELS

Norbert Giambiasi
Jean-Luc Paillet
Frédéric Châne

LSIS
UMR CNRS 6168
Université Aix-Marseille III
Campus St Jérome
52, Av. Escadrille Normandie Niemen
Marseille, 13397, FRANCE

ABSTRACT

In this paper, we present the formal transformation of Timed Input/Output Automata into simulation models, expressed in the DEVS formalism. This transformation takes place in an approach of a validation of high-level specifications by simulation. The validation is based on the simulation of a coupled model built with the system to be controlled and the control specifications. An example of this approach is given in the paper.

1 INTRODUCTION

Many formalisms have been developed to support verification and validation of formal specifications of control systems (Gajski, Vahid, Narayan, and Gong 1994; Peterson 1981; Alur and Dill 1994; Bolognesi, Lucidi and Trigila 1994).

Timed automata and variants have been used for these kinds of purposes (Alur and Kurshan 1996; Bengtsson, Larsen, Larsson, Pettersson and Wang 1996; Daws, Olivero, Tripakis and Yovine 1996; Henzinger and Ho 1995). Some works are based on the use of simulation technics of timed automata with the objective of establishing a correspondence between the states of a model M1 regarded as an implementation and a specification model M2.

We propose another approach in which we consider that the high level specification of a control system, expressed by a timed automaton, will be verified by simulation of a coupled model built with this specification and the model of the system to be controlled. The simulation models are represented in the framework of the DEVS formalism (Zeigler 1976; Zeigler 1989; Zeigler, Praehofer and Kim 2000; Giambiasi, Escude and Ghosh 2000).

The DEVS formalism has a clean operational semantics and a clean interpretation of the model elements in the real world. DEVS define a method of abstraction of dynamic systems that allows building timed simulation models (deterministic models) with a good accuracy. In other words, DEVS is well adapted to represent timed behavior of real systems. But DEVS does not adapt well to high level specifications of discrete event control systems. This due to the fact that at the first levels of specifications, the model can be undeterministic and the occurrence times of events are not defined precisely. While, Timed Automata are well adapted for these high levels of specification, it is not well adapted to represent a precise timed behavior deduced from the analysis of a real system with explicit state variables.

Consequently, we propose a methodology (Figure 1) based on the use of these two paradigms:

- Timed Automata for the high level specification of the control part,
- DEVS to represent the behavior of the system to be controlled.

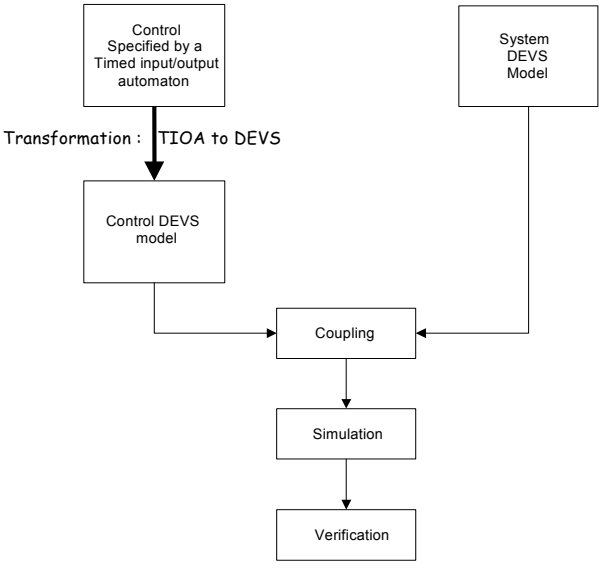

Figure 1: Our Methodology

In order to verify the specification with an accurate timing, we propose to transform T.A. into a DEVS model and to verify it by simulating the coupled model (control part coupled with the model of the controlled system).

Then, in a first step, we have to define a formal transformation of a Timed Automaton into a DEVS model.

In this paper, first we give a brief recall on timed automata and on DEVS formalism. Next, we present the method for the formal transformation of a timed automata into a DEVS model. Finally, we illustrate our approach by a complete example.

2 RECALL

2.1 Timed Automata (Springinveld, Vaandrager and D'Argenio 2001)

Timed Input/Output Automata are a particular class of timed automata (Alur and Dill 1994) with "good" properties, such as deterministic behavior, separation between input and output activity and input enabling.

A Bounded Timed Automata Model, a variant of Alur and Dill's model (Alur and Dill 1994) proposed by Lynch and Al (Gawlick, Segala, S_gaard-Andersen and Lynch 1994) is a finite automaton with a timing annotation. This annotation allows to express timing conditions for the state transitions, to determine values used for updating clocks, and to provide conditions, under which the model can remain in a given state. A TIOA is defined as Bounded Timed Domain Automata BTDA (Springintveld and Vaandrager 1996) together with a partitioning the set of actions into Input and Output actions (or events).

2.1.1 Finite Automaton or Finite Labeled Transition System

A labeled transition system (LTS) is a rooted, edge labeled multigraph. Formally, a LTS is an algebraic structure

$$\mathbf{A} = <Q, E, \Sigma, src, act, trg, q^0>,$$

where:

- Q is a set of states,
- E is a set of transitions,
- Σ is a set of events.
- src: $E \rightarrow Q$ which associates a source state to each transition,
- act: $E \rightarrow \Sigma$ which associates an event to each transition,
- trg: $E \rightarrow Q$ which associates a target state to each transition.

Initial conditions:

- q^0 is the initial state.

So we write:

δ: $q \rightarrow^a q'$, the transition with :

$$src(\delta) = q \; ; \; act(\delta) = a \; ; \; trg(\delta) = q'$$

Definition 1: An LTS A is said *lean* if each transition is fully determined by its source, event and target, as follows:

$$src(\delta) = src(\delta') \wedge act(\delta) = act(\delta') \wedge trg(\delta) = trg(\delta') \Rightarrow \delta = \delta'$$

Definition 2: An LTS A is said *deterministic* if it satisfies the following property:

$$src(\delta) = src(\delta') \wedge act(\delta) = act(\delta') \Rightarrow \delta = \delta'$$

Definition 3: An LTS A is a *finite automaton* if both Q and E are finite.

2.1.2 Bounded Time Domain Automata

A bounded time domain automaton (BTDA) is an extension of finite automaton model and it is a variant of timed automaton model.

A timed automaton is typically a (finite) automaton extended with a set of clocks.

For the BTDA model, we have the following definitions :

Definition 4: A clock is a variable x included in a domain dom(x) of the form $I \cup \{\infty\}$, where I is an interval over R bounded with values $\in$ Z. Let C be the set of clocks.

Definition 5: A term over C is an expression generated by the grammar z::=x|n|z+n, where $x \in C$ and $n \in R$. Let T(C) be the set of all terms over C.

Definition 6: A constraint over C is a Boolean combination φ of inequalities of the form $z \leq z'$ or $z < z'$ with $z,z' \in$ T(C). The Boolean constants are used to indicate if the constraint is realized or not. Let F(C) be the set of all these formulas.

Definition 7: A (simultaneous) assignment over C is a function $A : C \rightarrow T(C)$. Let M(C) be the set of all assignments.

Definition 8: A clock valuation is a vector that assigns to each clock $x \in C$ a value in dom(x). Let V(C) be the set of all valuations over C.

A timing annotation for a given automaton A is a tuple:

$$\mathbf{T} = <C, Inv, G, A, v^0>$$

where :

- C is a finite set of clocks.
- Inv: $Q \rightarrow F(C)$ associates an invariant to each state. The automaton A can remain in a state as long as its invariant remains true.

- G: E → F(C) associates a guard to each transition. A transition may be taken if the guard, a clock constraint, is satisfied by the current valuation of clocks.
- A: E → M(C) associates an assignment to each transition such as :

$$src(\delta) \wedge G(\delta) \Rightarrow \Lambda_{x \in C}(A(\delta)(x) \in dom(x)) \wedge Inv(trg(\delta))[A(\delta)]$$

holds for each $\delta \in E$.
Initial conditions :

- $v^0 \in V(C)$ is the initial valuation of clocks. We require v^0 satisfy $Inv(q^0)$ and $\forall x \in C$, $v^0(x) \in Z^{\infty}$.

Definition 9: The addition of a clock value v(x) with a value $d \in R^{>0}$ is defined as follow :

$$(v \oplus d)(x) = v(x)+d \quad \text{if } (v(x)+d) \in dom(x)$$
$$(v \oplus d)(x) = \infty \quad \text{otherwise}$$

Definition 10: A bounded time domain automata (BTDA) is a pair B = (A,T), where A is a finite automaton with $\Sigma_A \cap R = \varnothing$, and T is a timing annotation for A.

Definition 11: The operational semantics OS(B) of a BTDA B is the lean LTS A, which is specified by:

- $Q_A = \{(q,v) \in Q_B \times V(C_B) \mid v \models Inv_B(q)\}$,
- $\Sigma_A = \Sigma_B \cup R^{>0}$,
- $Q^0_A = (Q^0_B, v^0_B)$,
- And $\rightarrow^a$ is the smallest relation that satisfies the two following rules :

$\forall (q,v),(q',v') \in Q_A$, $a \in \Sigma_B$, $\delta \in E_B$ and $d \in R^{>0}$,

$$\frac{\delta: q \rightarrow^a q',\ v \models G_B(\delta), v' = v \circ A_B(\delta)}{(q,v) \rightarrow^a (q',v')}$$

$$\frac{q = q';\ v' = v \oplus d;\ \forall 0 \le d' \le d : v+d' \models Inv_B(q)}{(q,v) \rightarrow^d (q',v')}$$

We refer to the states of BTDA B as locations, to avoid confusion between the state of the BTDA and those of its operational semantics. So we have :

- discrete states $\in Q_B \times V(C_B)$,
- locations $\in Q_B$.

2.1.3 Timed Input/Output Automata (Gawlick, Segala, S_gaard-Andersen and Lynch 1994)

We recall some basic definitions on timed I/O automaton (TIOA).

In addition, some constraints will be added in order to allow the transformation of a TIOA into a discrete event simulation model. In fact, for this transformation a TIOA must be *deterministic*, with *isolated output* for excluding autonomous choice between two or more different outputs or between performing output and accepting input. Finally, it must be *input enabling* i.e. each input is enabled only in the interior of the invariant of each location, this means that inputs are enabled as long as time can progress.

Definition 12: A timed input/output automaton (TIOA) is a pair M = (B,P), where B is a BTDA and P=(I,O) is a partitioning of Σ_B in input events and outputs events with the following properties :

- (Determinism) If src(δ) = src(δ'), act(δ)=act(δ') and G(δ) ^ G(δ') is satisfiable then δ = δ'.
- (Isolated outputs) If src(δ) = src(δ'), act(δ) ∈ O and G(δ) ^ G(δ') is satisfiable then δ = δ'.
- (Input enabled) each input event is enabled only in the interior of the invariant i.e. only when the invariant is verified.

The operational semantics OS(A) of a TIOA A is the same as the one of BTDA B.

2.2 Discrete Event Specification Formalism: DEVS

According to the literature on DEVS (Zeigler 1976; Zeigler 1989), the specification of a discrete event model is a structure, M, given by:

$$\mathbf{M} = < \mathbf{X}, \mathbf{S}, \mathbf{Y}, \delta int, \delta ext, \lambda, \mathbf{D} >$$

where **X** is the set of the external input events, **S** the set of the sequential states, **Y** the set of the output events, δint is the internal transition function :

- δint : S → S defines the state changes caused by internal events,

δext is the external transition function :

- δext: $Q_M \times X_M \rightarrow S$ specifies the state changes due to external events,

λ is the output function :

- λ :S → Y

and the function D is the lifetime of the states:

- D:S → $R^{\geq 0} \cup \{\infty\}$. For a given state, s, D(s) represents the time interval during which the model will remain in the state s if no external event occurs.

A state may be viewed as passive when its lifetime is assumed to be infinite or active when the lifetime interval is assumed to be a finite real positive number, Zeigler (Zeigler 1976) introduces the concept of total states, TS, of a model as:

$$\mathbf{TS = \{(s,e) : s \in S,\ o < e < D(s)\}}$$

where **e** represents the elapsed time in state s. The concept of total state is fundamental in that it permits one to specify a future state based on the elapsed time in the present state. Potential benefits may lie in its ability to implement event filtering (Ghosch and Meng-Lin 1989; Ghosch and Giambiasi 1999).

A key contribution of DEVS lies in decomposing the traditional transition function into two sub-functions: internal transition function and external transition function. The internal transition function permits one to capture the autonomous evolution of the model. The external transition function reflects the evolution of the model corresponding to externally induced input events. The output function is defined only for active states and is executed only when the elapsed time in a given state equals its lifetime. When a model is in an active state sk, it sends an output event as defined by the output function, at the end of the lifetime of the current active state. From the simulation perspective, this implies that the output function is executed prior to the internal transition function.

2.3 Coupled Models

A coupled DEVS Model, DN, is a structure :

$$\mathbf{DN} = \langle X,Y,M,EIC,EOC,IC \rangle$$

Where :

- X = input events set
- Y = output events set
- M = DEVS components set
- EIC $\subseteq$ DN.INxM.IN : external input coupling relation
- EOC $\subseteq$ M.OUTxDN.OUT : external output coupling relation
- IC $\subseteq$ M.OUTxM.IN : internal coupling relation
- DN.IN and DN.OUT refer to the input and output ports of the coupled model.
- M.IN and M.OUT refer to the input and output ports of component Models.

EIC, EOC, IC specify the connections between the set of models M and input and output ports X,Y.

3 FROM TIOA TO ATOMIC DEVS MODEL

For this transformation, we make two hypotheses.

i. The timestamp of a transition over an output event has a unique value (not define in a time interval).
ii. The conditions of guards of two different transitions, with the same input event and from the same state must be expressed using different clocks. Formally:
Let e1,e2 $\in$ E such that :
   - src(e1) = src(e2),
   - act(e1) = act(e2) and
   - let X1, X2 the subset of clocks used by G(e1) and G(e2) respectively.

$$\Rightarrow \mathbf{X1 \cap X2 = \varnothing}$$

In this case the two guards G(e1) and G(e2) cannot be true at the same date, according to the constraint of determinism of the automaton.

3.1 Untimed DEVS Syntactic Transformation

We begin by a definition of the syntactic transformation of a TIOA into a DEVS model.

Relationships between sets :

- S = Q, the set of discrete states is the same in the two models.
- X_M = I, the set of input events of the TIOA model represents the set of input event variables in the DEVS model.
- Y_M = O, the set of output events in the TIOA represents the set of output event variables in the DEVS model.

Relationships between functions :

$\forall ei \in E$:

- External transition function -δext-:

if act(ei) $\in$ I then δext ((src(ei),e),act(ei)) = trg(ei)

- Internal transition function -δint-:

If act(ei) $\in$ O the δint(src(ei)) = trg(ei)
and λ(src(ei)) = act(ei)

Example: See the untimed automaton in Figure 2.

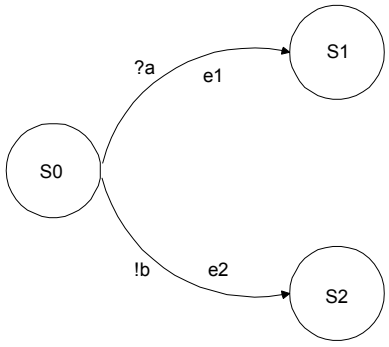

Figure 2: A Simple Untimed Automaton

Defined by :

- Q = {s0,s1,s2}
- E = {e1,e2}
- Σ = {I={a},O={b}}
- **src**:
  src(e1) = src(e2) = s0
- act:
  act(e1) = a; act(e2) = b
- trg:
  trg(e1) = s1; trg(e2) = s2

becomes, in DEVS :

- S = Q = {s0,s1,s2}
- XM = {a}
- YM = {b}
- e1 :

$$act(e1) = a \in I \quad \Rightarrow \delta ext((src(e1),e),act(e1)) = trg(e1)$$
$$\Rightarrow \delta ext((s0,e),a) = s1$$

- e2 :

$$act(e2) = b \in O \quad \Rightarrow \delta int(src(e2)) = trg(e2)$$
$$\text{and} \quad \lambda(src(e2)) = act(e2)$$
$$\Rightarrow \delta int(s0) = s2$$
$$\text{and} \quad \lambda(s0) = b$$

and the state graph (Figure 3) :

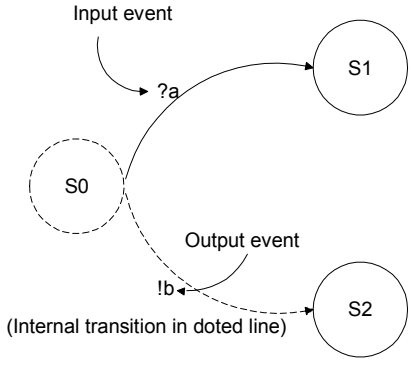

Figure 3: State Graph of the Resulting DEVS Model

This first step gives an untimed DEVS(which can be not deterministic).In order to obtain a complete transformation, we have to introduce new state variables called temporal state variables, a valuation function and to express the lifetime function.

Definition 13: A temporal state variable is a variable x included in a domain dom(x) = J ∪ {∞}, where J is an interval over R. Let C_M is the set of state variables.

Definition 14: A term over C_M is an expression generated by the grammar z::=x|n|z+n, where $x \in C_M$ and $n \in R$. Let T_M (C) be the set of all terms over C_M.

Definition 15: A constraint over C_M is a Boolean combination φ of inequalities of the form $z \leq z'$ or $z < z'$ with $z,z' \in T_M (C_M)$. The Boolean constants are used to indicate if the constraint is realized or not. Let $F_M (C_M)$ be the set of all these formulas.

Definition 16: A (simultaneous) assignment over C_M is a function : $C_M \rightarrow T_M (C_M)$. Let $M_M (C_M)$ be the set of all assignments.

Definition 17: A temporal state variable valuation v is a vector that returns for each state variable $x \in C_M$ a value in dom(x). Let V_M (C_M) be the set of all valuations over C_M.

Definition 18: We define the discrete (or sequential) state of a DEVS model as follow :

$$Sd = \{(s,v) \in S_M \times V_M\}$$

3.2 Lifetime Function -D-

According to the first hypothesis, all transition guards of input events are fully determined and can occurred only on one value determined by either an Invariant or a Guard.

$$D : S \times V_M (C_M) \rightarrow R$$

$\forall s_i \in S$:

- if $Inv(s_i) = \varnothing$ then $D(s_i) = \infty$
- else
  the invariant and guard are included in $F_M(C_M)$.

Guards and invariants are Boolean combinations of terms of the form $z \leq z'$ or $z < z'$ and $z = z'$.

Let Op={≤,<,=}:

1. A formula $z_i \; Op \; z_i' \wedge z_j \; Op$ is equivalent to :

$$\min((z_i' - z_i),(z_j' - z_j))$$

2. A formula $z_i \; Op \; z_i' \vee z_j \; Op \; z_j'$ is equivalent to :

$$\max((z_i' - z_i),(z_j' - z_j))$$

3. A combination of this two type of formula is a combination of min-max.
4. Let GM(ei) = the result of the combination of this three operations on guard.
5. Let InvM(ei) = the result of the combination of this three operations on the invariant.
6. Let D(si) = the result of the min of the combination of this operations on invariant and Guard.

For a given state D(si) depends on the values of the temporal state variables.

3.3 Operational Semantics OS(M)-

Definition 19: The operational semantics OS(M) of a DEVS M is specified by:

- **Discrete states:**

$$Sd = \{(s,v) \in S_M \times V_M\}$$

- **Total states:**

$$Q_M = \{(Sd,e) = ((s,v),e) \in S_M \times V_M \times R \mid 0 \leq e \leq D(s)\}$$

- **Initial state:**

$$Q^0_M = ((s^0,v^0_B),0)$$

$\forall ei \in E$ in the TIOA:

if **act(ei) ∈ I** and $v(C_M) \models G_M(ei)$ then
$v = v+e$ and $v' = v_O A(ei)$ and
$\delta ext((src(ei),v(C_M)),act(ei)) = (trg(ei),v')$ and
$e:=0$

if **act(ei) ∈ O** and $v(C_M)|=G_M(ei)$ and $e = D(src(ei))$ then
$v = v+e$ and $v'=v_O A(ei)$ and
$\delta int((src(ei),v(C_M))) = (trg(ei),v')$ and
$\lambda(trg(ei)) = act(ei)$ and
$e:=0$

4 EXAMPLE AND SIMULATION

4.1 Filling System Example

We consider that the DEVS model of the system to be controlled is in the model base. This system, a filling system, is composed of a tank with a valve, a conveyor and barrels (Figure 4).

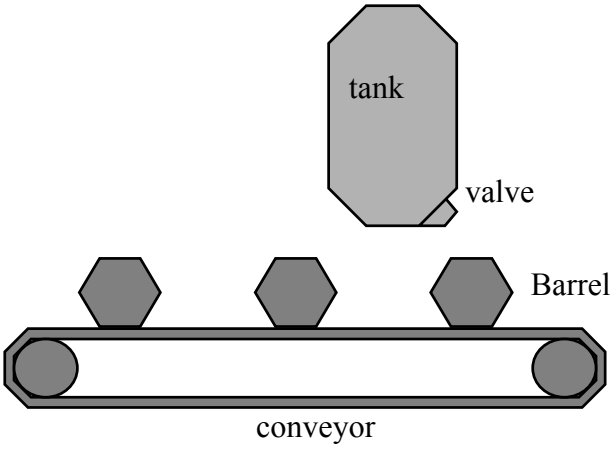

Figure 4: A Simple Filling System

This filling system model has two discrete event inputs:

- Control of the valve : VAL = {Open, Close}
- Control of the conveyor : MOT = {Start, Stop}

And two sensor outputs :

- Barrel level : BL = {Full}
- Barrel position : BP = {Good}

The DEVS model of the filling system is in Figure 5. One possible specification of the control system by a timed automaton is given in Figure 6.

This Timed Automaton is defined by :

- Q = {
  Init,Adcon0,Adcon1,Stop,Filling0,Filling1,Error}
- E = {e1,e2,e3,e4,e5,e6,e7,e8}
- Σ = {I = {Bp.Good, Bl.Full },
  O={Mot.Start, Mot.Stop, Val.Open, Val.Close}}

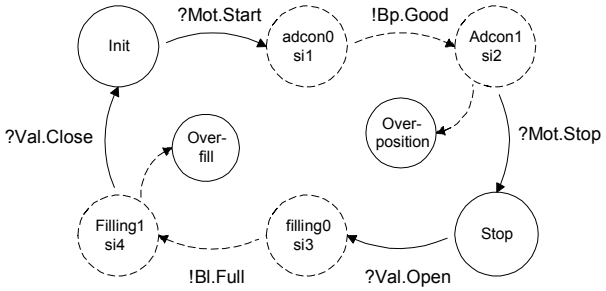

Figure 5: DEVS Model of the Filling System

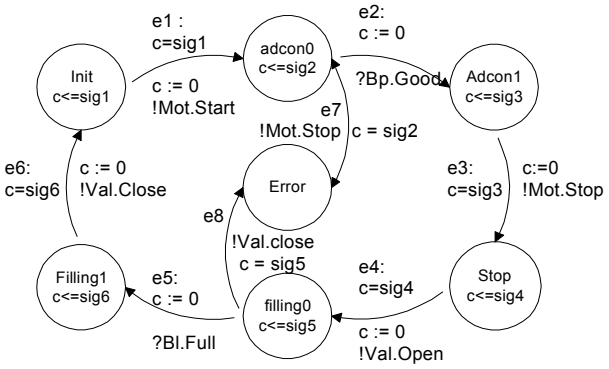

Figure 6: Timed Automaton of the Control System

- src:
  src(e1) = Init; src(e2) = Adcon0;
  src(e3) = Adcon1; …
- act:
  act(e1) = Mot.Start; act(e2) = Bp.Good;
  act(e3) = Mot.Stop; …
- trg:
  trg(e1) = Adcon0; trg(e2) = Adcon1;
  trg(e3) = Stop; …
- q^0 = Init

Timing annotation :

- C = {c}
- Inv:
  Inv(Init) = (c<=sig1); Inv(Adcon0) = (c<=sig2);
  Inv(Adcon1) = (c<=sig3); Inv(Stop) = (c<=sig4);
  Inv(Filling0) = (c<=sig5); Inv(Filling1) = (c<=sig6);
  Inv(Error) = ∅
- G :
  G(e1) = (c = sig1); G(e2) = ∅;
  G(e3) = (c = sig3); G(e4) = (c = sig4);
  G(e5) = ∅; G(e6) = (c = sig6);
  G(e7) = (c = sig2); G(e8) = (c = sig5)
- A:
  A(e1) = A(e2) = A(e3) = A(e4) = A(e5) = A(e6) = (c:= 0)
  A(e7) = A(e8) = ∅
- v^0 = {0}

Now, in order to verify this specification by simulation, we transform this model into a DEVS atomic model :

Step 1 : untimed DEVS :

- S=Q= {Init,Adcon0,Adcon1,Stop,Filling0,Filling1,Error}
- X_M = I ={Bp.Good, Bl.Full}
- Y_M = O ={Mot, Val}
- s° = q° = Init
- C_M = {c}
- v° (C_M) = {0}
- Act(e1) ∈ O
  ⇒ δint(src(e1)) = trg(e1)
  and λ(trg(e1)) = act(e1)
  ⇒ δint(Init) = Adcon0
  and λ(Adcon0) = /Mot.Start
- Act(e2) ∈ I
  ⇒ δext ((src(e2),e),act(e2)) = trg(e2)
  ⇒ δext ((Adcon0,e),Bp.Good) = Adcon1

…

Step 2 : Transformation in atomic DEVS :

We add state variable sets :

$$C_M = \{c\}$$

and the initial valuation of state variable c :

$$v^\circ (C_M) = \{0\}$$

Operational semantics :

Total states :

Q_M = {((Init,v),e), ((Adcon0,v),e), ((Adcon1,v),e), ((Stop,v),e), ((Filling0,v),e), ((Filling1,v),e), ((Error,v),e)}

Q° = ((Init,v),e)

Lifetime Function :

Inv(Init) = (c<=sig1) ⇒ D(Init) = sig1-v(c)
Inv(Adcon0) = (c <= sig2) ⇒ D(Adcon0) = sig2-v(c)
Inv(Adcon1) = (c<=sig3) ⇒ D(Adcon1) = sig3-v(c)
Inv(Stop) = (c<=sig4) ⇒ D(Stop) = sig4-v(c)
Inv(Filling0) = (c <= sig5) ⇒ D(Filling0) = sig5-v(c)
Inv(Filling1) = (c <= sig6) ⇒ D(Stop) = sig6-v(c)
Inv(Error) = ∅ ⇒ D(Error) = ∞

Then internal and external transition functions become:

Act(e1) ∈ O ⇒ if v |= G_M(e1) and e = D(src(e1))
⇒ if v |= (c = sig1) and e = D(Init)
⇒ if v |= (c = sig1) and e = sig1-v(c)
⇒ v = v+e and v' = v $_O$ A(e1)
⇒ δint((Init,v)) = (Adcon0,v')
and λ((Init,v')) = /Mot.Start ;

Act(e2) ∈ I ⇒ if v |= ∅
⇒ v = v+e and v' = v $_O$ A(e2)
⇒ δext (((Adcon0,v),e),Bp.Good) = (Adcon1,v')

…

In this case c has the same behavior than e, c is unused. The final DEVS state graph is in Figure 7.

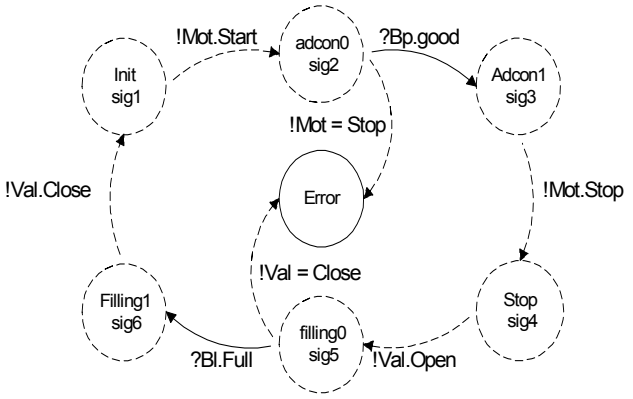

Figure 7: Final State Graph of the DEVS Model of the Control System

4.2 Coupled Model and Simulation

The coupled model is built with the control model and the system model by connecting the input/output ports of the two basic models (Figure 8).

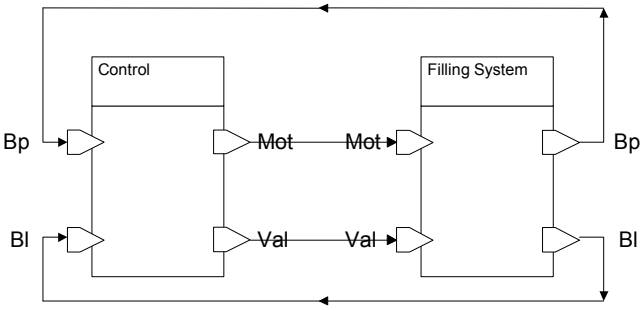

Figure 8: Coupled Model of the Control and the Filling System

Simulation :

Global Clock T = 0.

v°(C_M) = {0}.

Let the DEVS model of the system (Figure 9) :

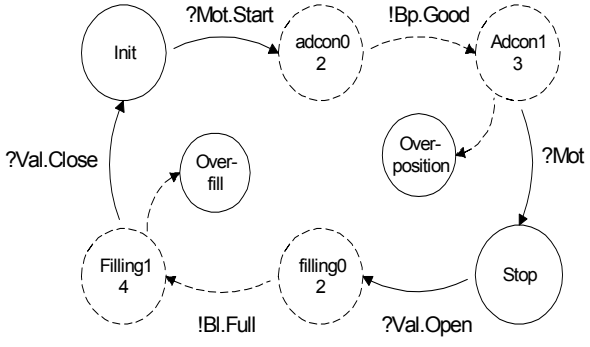

Figure 9: DEVS Model of the Filling System

The following tables give the timed evolution of the coupled model (Table 1).

Table 1: Simulation Table

System DEVS Model

Event Date (t.u.)	Current state	Timelife D	External transition function(ext)	Internal transition function(int)	Trans. Date	Output function (O)	Output	Next state
0	Init	Infinite	ext((init,e),?Mot.Start)		1			Adcon0
1	Adcon0	si1 = 2		int(Adcon0)	3	O(Adcon0)	!Bp.Good	Adcon1
3	Adcon1	si2 = 3		int(Adcon1)				Over-pos.
"	"	"	ext((Adcon1,e),?Mot.Stop)		4			Stop
4	Stop	Infinite	ext((Stop,e),?Val.Open)		6			Filling0
6	Filling0	si3 = 2		int(Filling0)	8	O(filling0)	!Bl.Full	Filling1
8	Filling1	si4 = 4		int(Filling1)				Over-fill
"	"	"	ext((Filling1,e),?Val.Close)		11			Init
	Over-Pos.	Infinite						
	Over-Fill	Infinite						

Control DEVS Model

Event Date (t.u.)	Current state	Timelife D	External transition function(ext)	Internal transition function(int)	Trans. Date	Output function (O)	Output	Next state
0	Init	sig1 = 1		int(Init,v)	1	O(Init,v)	!Mot.Start	(Adcon0,v)
1	Adcon0	sig2 = 3	ext(((Adcon0,v),e),?Bp.Good)		3			(Adcon1,v)
"	"	"		int(Adcon0,v)		O(Adcon0,v)	!Mot.stop	(Error,v)
3	Adcon1	sig3 = 1		int(Adcon1,v)	4	O(Adcon1,v)	!Mot.stop	(Stop,v)
4	Stop	sig4 = 2		int(stop,v)	6	O(stop,v)	!Val.Open	(Filling0,v)
6	Filling0	sig5 = 1		int(Filling0,v)	8	O(Filling0,v)	!Val.Close	(Filling1,v)
"	"	"	ext(((Filling0,v),e),?Bl.Full)					(Error,v)
8	Filling1	sig6 = 3		int(Filling1,v)	11	O(Filling1,v)	!Val.Close	(Init,v)
	Error	Infinite						

5 CONCLUSION

In this paper, we have presented the formal transformation of a TIOA into a DEVS model in order to verify by simulation, high level specifications given by a TIOA.

The proposed methodology seems to be realistic because it allows the simulation of very complex coupled models, for which a formal proof is quite impossible. In addition, going from high level specifications to the low level design need to define deterministic models, which is done for the transformation of a TIOA into DEVS

REFERENCES

Alur, R. and Dill, D.L. 1994. A theory of timed automata, *Theorical computer science*, Vol. 126, No 2, pp 183-235.

Alur, R. and Kurshan, R.P. 1996. Timing analysis in COSPAN. In Alur et al. 1996, pages 220-231.

Bengtsson, J., Larsen, K.G., Larsson, F., Pettersson, P. and Wang Yi. 1996. UPPAAL: a tool suite for the automatic verification of real-time systems. In Alur et al. 1996, pages 232-243.

Bolognesi, T., Lucidi, F. and Trigila, S. 1994. A timed full LOTOS with time/action tree semantics, in: T. Rus and C. Rathay (eds), Theories and Experiences for Real-Time System Development, Amast Series in Computing, *World Scientific*, Singapore, pp. 205-237.

Daws, C., Olivero, A., Tripakis, S. and S. Yovine, S. 1996. The tool kronos. In Alur et al. 1996, pages 208-219.

Gajski, D., Vahid, F., Narayan, S. and Gong, J. 1994. Specification and Design of Embedded Systems, *Prentice-Hall*, Englewood Cliffs, NJ.

Ghosch, S. and Meng-Lin Yu 1989. A preemptive scheduling mechanism for occurate behavioral simulation of digital designs; *IEEE trans. on Computers*, vol. 38, Nov.

Ghosch, S. and Giambiasi, N. 1999. "LANGUAGE BARRIERS IN HARDWARE DESIGN: On the Need for Consistency between the VHDL Language Constructs and Underlying Hardware Design", IEEE Circuits and Devices, Vol. 15, No. 5, September, pp. 25-40.

Giambiasi, N., Escude, B. and Ghosch, S. 2000. "GDEVS: A Generalized Discrete Event Specification for Accurate Modeling of Dynamic Systems*". Transactions of the SCS*, 17(3) pp. 120-134.

Gawlick, R., Segala, R., S_gaard-Andersen, J.F. and Lynch ,N. 1994. Liveness in timed and untimed systems. In S. Abiteboul and E. Shamir, editors, Proceedings 21th ICALP, Jerusalem, volume 820 of *Lecture Notes in Computer Science*. Springer-Verlag. A full version appears as MIT Technical Report number MIT/LCS/TR-587.

Henzinger, T.A. and Ho, P.-H. 1995. HyTech: TheCornell HYbrid TECHnology Tool. In U.H. Engberg, K.G. Larsen, and A. Skou, editors, Proceedings of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems, Aarhus, Denmark, volume NS-95-2 of BRICS Notes Series, pages 29{43. Department of Computer Science, University of Aarhus, May.

Peterson, J.L. 1981. Petri Net Theory and the Modeling of Systems, *Prentice-Hall*, Englewood Cliffs, NJ.

Springinveld, Jan, Vaandrager, Frits and D'Argenio, Pedro R. 2001. Testing Timed Automata, *Theorical Computer Science,* Vol. 254, pp. 225-257.

Springintveld, J.G. and Vaandrager, F.W. 1996. Minimizable timed automata. In B. Jonsson and J. Parrow, editors, Proceedings of the Fourth International Symposium on Formal Techniques in Real Time and Fault Tolerant Systems (FTRTFT'96), Uppsala, Sweden, volume 1135 of *Lecture Notes in Computer Science*, pages 130-147. Springer-Verlag.

Zeigler, B. P. 1976. Theory of Modeling and Simulation, *John Wiley*, New York.

Zeigler, B.P. 1989. DEVS Representation of Dynamical system, in: *Proc. of the IEEE*, Vol.77, pp.72-80.

Zeigler, B. P., Praehofer, H., and Kim, T.G. 2000. *Theory of Modeling and Simulation,* Integrating Discrete Event and Continuous Complex Dynamic System, 2nd Edition.

AUTHOR BIOGRAPHIES

NORBERT GIAMBIASI is a Professor of the University of "Aix-Marseille III" since 1981. In October 1987, he created the LERI in a new engineering school where he was the director of Research and Development. He created recently the "Laboratoire des Sciences de l'Information et des Systemes" (LSIS), UMR CNRS 6168. He wrote a book on C.A.D and he is an author of more than 150 international publications. He is a member of the program committee of several international conferences (Micad, RFIA, CIAM, N'Euro 88, Expert Systems and their Applications, IMACS, ESS, ESM....) and president of 'Neural-network and their Applications'. He has been a scientific manager of more than 50 research contracts (with E.S Dassault, Thomson-Cimsa, Bull, Siemens, Cnet, Esprit, Euréka, Usinor, etc.). His main current interests converge on: formalism for specification of hybrid simulation models, distributed simulation, discrete event simulation of hybrid systems, CAD systems and design automation. He can be contacted by e-mail at <norbert.giambiasi@univ.u-3mrs.fr>.

JEAN-LUC PAILLET is a "Maitre de Conferences" of the Université de Provence (Aix-Marseille I). He supervised the works of the "Modeling and Verifying Digital Systems" team in the "Laboratoire d'Informatique de Marseille" (LIM) until 2000. He is presently member of the "Laboratoire des Sciences de l'Information et des Systemes" (LSIS). After attending an initial education in Mathematics, he published works in mathematical logics, and then worked on logics applied to Computer Sciences. In 1990, he defended the "Habilitation a diriger des recherches" in Computer Sciences at the "Université de Provence". His principal contributions in the Computer Sciences are: defining a functional semantics for the microprocessors, designing a BDD-based Tautology Checker "TACHE", defining the functional algebra "P-calculus" for modeling the synchronous behaviors, and defining a mathematical model and a formal language for high level specifications of discrete event systems, named DECM (Discrete Event Calculus Model). He can be contacted by e-mail at <jean-luc.paillet@cmi.univ-mrs.fr>.

FREDERIC E. CHANE is a D.E.A. M.C.A.O. student at the University Aix-Marseille III where he rejoin the "Laboratoire des Sciences de l'Information et des Systemes" (LSIS) in Marseille, France. He had just finished his studies in computer sciences and it is his first paper. He can be contacted by e-mail at <fchane@club-internet.fr>.