FDEVS: A GENERAL DEVS-BASED FORMALISM FOR FAULT MODELING AND SIMULATION

Ernesto Kofman[1,3], Norbert Giambiasi[2] and Sergio Junco[1]

[1] Departamento de Electrónica
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario
Ríobamba 245 Bis – (2000) Rosario – Argentina
E-mail: ekofman@eie.fceia.unr.edu.ar, sjunco@fceia.unr.edu.ar

[2] DIAM – IUSPIM
Université d'Aix-Marseille III
Av. Escadrille Normandie Niemen 13397 – Marseille Cedex 20 – France
E-mail: norbert.giambiasi@iuspim.u-3mrs.fr

[3] CONICET – Argentina

KEYWORDS
Discrete Event Systems, DEVS, Fault Modeling and Simulation.

ABSTRACT

A new formalism for fault modeling and simulation in discrete event systems is introduced. This formalism, called FDEVS and based on the DEVS formalism developed by Zeigler, allows to handle typical classes of fault hypotheses (like behavioral, structural, and delay faults) via the classical technique of fault injection. It also embodies the necessary features to automatically generate and propagate fault hypotheses in the system. It is shown that atomic FDEVS models can be coupled in a modular and hierarchical way, in order to represent complex systems. It is also shown that coupled FDEVS models can be easily simulated in a digital computer. Some examples are presented, in order to illustrate the features of the methodology.

INTRODUCTION

Model-based diagnosis of discrete event systems requires well-defined faulty models and the use of fault simulation techniques to study faulty behaviors.

Since numerous years, fault simulation has constituted a basic tool of CAD–systems for digital circuits (Larsson 1999; Giambiasi et al. 1991; Santucci et al. 1993; Abramovici et al. 1990). In opposition, fault simulation is not widely used in the domain of discrete event systems, whereas analysis of faulty behavior of discrete event systems could take profit from the established knowledge of digital domain. Fault simulation could be used for definition of observation points, validation of test sequences, research of equivalent faults, and identification of fault signature, for instance. Such kind of applications makes fault simulation a very useful aid in designing control or diagnosis systems. In this article, FDEVS is presented, a new general formalism for fault modeling and simulation in discrete event systems, which constitutes a methodology of theoretical interest, as well as a supporting base for the design of any practical software tool.

FDEVS allows the specification and the simulation of faulty discrete event models. It is an extension of the DEVS formalism (Zeigler 1984, 1989, 2000). FDEVS is conceived in order to capture the classical classes of fault hypothesis (behavioral faults, structural faults, etc) (Giambiasi and Sigal 1982) and to allow the simulation of transient, intermittent and permanent faults, and also of delay faults (Perret and Giambiasi 1998; Giambiasi et al. 1996) using a fault injection technique.

Behavioral faults modify the transition functions and /or the output function of a DEVS model. Delay faults modify the lifetime of the states bringing, for example, the transformation of a steady state into a transient state.

Structural faults can be simulated in coupled models. These faults modify the coupling relationships existing among components. This kind of fault hypothesis corresponds, in fact, to dynamic structure DEVS models.

Under these assumptions, in order to represent a faulty behavior of a discrete event system, a classical DEVS model can be specified. But doing so, a DEVS model has to be built for each fault hypothesis. The FDEVS formalism proposed in this paper, providing a framework adapted to this kind of problems, allows describing in a unique model all the faulty behaviors to be simulated. In addition, it permits a clear specification of these faulty behaviors and a formal definition of fault occurrences.

An FDEVS model is defined by a classical DEVS structure equipped with the following additional features:

- A set of faults, which contains the entire possible fault hypothesis and a non-fault element (representing the absence of faults). This set is part of the domain of definition of the transition, output and time advance functions. This means that these functions might be modified by the presence of faults. The set of faults constitutes also part of the image of the transition functions. It implies that the model can also handle the evolution of the faults.
- A fault transition function, which represents the effects of the injection of faults in the model.
- A set of input fault events whose occurrence triggers the fault transition function. Under the classical fault injection hypothesis, this set is the same as the set of above-mentioned faults, and the fault transition function is just an identity function. However, FDEVS can deal with more general cases where the fault that appears might depend on the state of the system.

Atomic FDEVS models can be coupled in a modular and hierarchical way to represent complex systems. A coupled FDEVS model, defined in a similar way as a classical coupled DEVS, allows structural fault simulation with the addition of a set of faults that can modify the coupling structure.

Abstract simulators of atomic and coupled FDEVS models are also presented to explain the operational semantics of the FDEVS formalism.

THE FDEVS FORMALISM

Consider a system whose healthy or normal behavior can be represented by the following DEVS structure, with the usual notation (Zeigler 1984, 2000):

$$M_n = \langle X_n, S_n, Y_n, d_{n-int}, d_{n-ext}, l_n, ta_n \rangle$$

Based on that formalism, the following structure can be defined, in order to take faults into account:

$$M_f = \langle X, S, Y, d_{int}, d_{ext}, d_{fault}, l, ta, X_f, F \rangle \text{ , where}$$

$X = X_n \cup \tilde{X}$ is the set of input values, with $\tilde{X}$ representing a set of new input values.

$S = S_n \cup \tilde{S}$ is the set of state values, and $\tilde{S}$ is the set of new states that the model could reach due to the presence of faults.

$Y = Y_n \cup \tilde{Y}$ is the set of output values, and $\tilde{Y}$ represents the set of the new values that the output can take when there is a fault in the system.

$F = F' \cup \{f\}$ is the set of faults, with F' representing the set of all possible faulty hypothesis, and the element f standing for the absence of faults. Notice that a single element of F' can represent a single fault hypothesis or a multiple fault hypothesis.

X_f is the set of event values that represents the sudden appearance (or disappearance) of faults. These events are managed outside the model in order to allow the possibility of inject faults. With this definition, it is possible to model transient or permanent faults.

$d_{int} : S \times F \rightarrow S \times F$ is the internal transition function, which has the restriction $d_{int}(s, f) = (d_{n-int}(s), f)$.

$d_{ext} : S \times F \times \Re_0^+ \times X \rightarrow S \times F$ is the external transition function, satisfying $d_{ext}(s, f, e, x) = (d_{n-ext}(s), f)$ if $x \in X_n$.

The presence of the set F in the right hand of the internal and external transition function allows to deal with systems whose faulty hypothesis evolution has been included in the model (automatic fault generation).

$d_{fault} : S \times F \times \Re_0^+ \times X_f \rightarrow S \times F$ is the fault transition function.

$l : S \times F \rightarrow Y$ is the output function, which verifies the restriction $l(s, f) = l_n(s)$.

$ta : S \times F \rightarrow \Re_0^+$ is the lifetime function (or time advance function), which has the restriction $ta(s, f) = ta_n(s)$.

The operative semantics of the FDEVS is similar to the DEVS. The difference is that any time a fault event occurs (an event in X_f), a transition takes place, which is calculated by the fault transition function. As in the case of external events, this event will not produce any output.

Example: Consider a processor able to perform certain jobs: $\{job_1, job_2, ..., job_n\}$. In absence of faults, each job can be done taking a time $t_p(job_i)$. When the processor finishes a job, it produces an output $y(job_i)$. While the processor is doing a job, it ignores any job arriving. Under these considerations, the DEVS model of the healthy behavior of the system is the following:

$$M_{n,P} = \langle X_n, S_n, Y_n, d_{n\text{int}}, d_{next}, l_n, ta_n \rangle \text{, where}$$

$$X_n = \{job_1, job_2, ..., job_n\}$$
$$S_n = \{job_1, job_2, ..., job_n\} \cup \{f\} \times \Re_0^+$$
$$Y_n = \{y(job_1), y(job_2), ..., y(job_n)\}$$
$$d_{n-int}(job, s) = (f, \infty)$$
$$d_{n-ext}(job, s, e, x) = \begin{cases} (x, t_p(x)) & if \ \ job = f \\ (job, s - e) & otherwise \end{cases}$$
$$l_n(job, s) = y(job)$$
$$ta_n(job, s) = s$$

Consider now two different fault hypotheses:

-With the first fault hypothesis, the processor no longer ignores input events while it is busy. In the corresponding

faulty behavior, when the processor receives a job, it discards the job being processed and starts with the new one.
-With the second fault hypothesis, the processor works at the half of its normal velocity.
Both faults can be simultaneously present (multiple fault hypothesis). The injection of the first one will be modeled by an event called $BFon$, while the other will be modeled by the event $SFon$. The events $BFoff$ and $SFoff$ will be used to deactivate the mentioned faults. Then, the FDEVS model representing the complete behavior is the following:

$M_P = \langle X, S, Y, d_{\text{int}}, d_{ext}, d_{fault}, l, ta, X_f, F \rangle$, where

$X = X_n$ (There are not out of range input events).
$S = S_n,\ Y = Y_n,\ X_f = \{BFon, SFon, BFoff, SFoff\}$
$F = \{BF, SF, BFSF, f\}$
$d_{int}((job, s), f) = ((f, \infty), f)$

$$d_{ext}(job, s, e, x, f) = \begin{cases} (d_{n-ext}(job, s, e, x), f) & if\ \ f = f \\ ((job, s - e), f)\ if\ \ f = SF \wedge job \neq f \\ ((x, t_1), f) & otherwise \end{cases}$$

where $t_1 = \begin{cases} tp(x) & if\ \ f = BF \\ 2tp(x) & otherwise \end{cases}$

$$d_{fault}(job, s, e, f, x_f) = \begin{cases} ((job, s - e), BF)\ if\ \ f = f \wedge x_f = BFon \\ ((job, s - e), BFSF)\ if\ \ f = SF \wedge x_f = BFon \\ ((job, s - e), f)\ if\ \ f = BF \wedge x_f = BFoff \\ ((job, s - e), SF)\ if\ \ f = BFSF \wedge x_f = BFoff \\ ((job, 2(s - e)), SF)\ if\ \ f = f \wedge x_f = SFon \\ ((job, 2(s - e)), BFSF)\ if\ \ f = BF \wedge x_f = SFon \\ ((job, \frac{(s - e)}{2}), f)\ if\ \ f = SF \wedge x_f = SFoff \\ ((job, \frac{(s - e)}{2}), BF)\ if\ \ f = BFSF \wedge x_f = SFoff \\ ((job, s - e), f) \quad otherwise \end{cases}$$

$l(job, s, f) = l_n(job, s)$
$ta(job, s, f) = ta_n(job, s)$

The example shows that a relative complex fault behavior including temporal and behavioral fault hypothesis can be formally represented by an FDEVS structure.

COUPLED FDEVS MODELS

In absence of structural faults, the coupling of FDEVS models can be done in the same way that in classic DEVS. Under this consideration, the faulty and healthy models will have the same structure representing the coupling. The only difference will be that in the healthy model, the input and output sets are subsets of the corresponding sets in the faulty model because the faulty model deals with extended input and output values. Dealing with extended input and output sets implies the extension of the translation functions.

Thus, given a system whose healthy behavior can be described by a coupled DEVS structure:

$$N_n = \langle X_n, Y_n, D, \{M_{nd}\}, \{I_d\}, \{Z_{ni,d}\}, Select \rangle$$

its faulty behavior in absence of structural faults can be represented by the structure:

$$N = \langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, Select \rangle$$

where the input and output sets are defined in the same way as for atomic FDEVS. The models M_{nd} and M_d are the healthy and associated faulty subsystems respectively.

The presence of structural faults implies some change in the original coupling of the system. A structural fault can modify the way in which two models are connected, for example, it can produce the appearance or disappearance of a connection between two models

Thus, with the presence of structural faults, the coupling structure *N* becomes:

$$N_{sf} = \langle X, Y, D, \{M_d\}, \{I_{d,f}\}, \{Z_{i,d,f}\}, Select, d_{fault}, F, X_f \rangle$$
where:

F is the set of structural faults, including an element f that represents the absence of structural faults.

$d_{fault} : F \times X_f \rightarrow F$ is the fault transition function.
For each pair (*d, f*), $d \in D \bigcup N_{sf}$, $f \in F$, the influencer set $I_{d,f} \subset D \cup N_{sf}$ is the set of components that influence over the component *d* when the fault *f* is present. The restriction $I_{d,f} = I_d$ must be satisfied.

For each triple (*i, d, f*) where $d \in D \cup N_{sf}$, $f \in F$ and $i \in I_{d,f}$, the function $Z_{i,d,f}$ is the translation function.
$Z_{i,d,f} : X \times F \rightarrow X_d$, if $i = N_{sf}$.
$Z_{i,d,f} : Y_i \times F \rightarrow Y$, if $d = N_{sf}$.
$Z_{i,d,f} : Y_i \times F \rightarrow X_d$, if $d \neq N_{sf}$ and $i \neq N_{sf}$.
The restriction $Z_{i,d,f}(u, f) = Z_{i,d}(u)$ must be always satisfied.

Example: Consider a queue, whose behavior is modeled by the following DEVS structure:

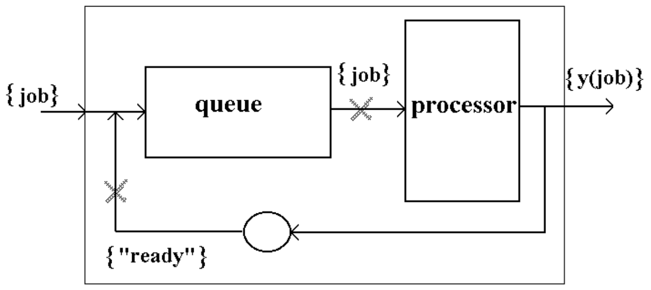
$M_Q = \langle X, S, Y, d_{\text{int}}, d_{ext}, l, ta \rangle$, where
$X = \{job_1, job_2, ..., job_n\} \bigcup \{"ready"\} = J \bigcup \{"ready"\}$
$S = J^+ \bigcup \{f\} \times \{free, busy\} \times \{0; \infty\}$
$Y = J$
$d_{int}(q \bullet job, pst, s) = (q, "busy", \infty)$

$$d_{ext}(q, pst, s, e, x) = \begin{cases} (x \bullet q, pst, \infty) & if\ x \in J \wedge pst = "busy" \\ (x \bullet q, pst, 0) & if\ x \in J \wedge pst = "free" \\ (q, "free", \infty) & if\ x = "ready" \wedge q = f \\ (q, "free", 0) & otherwise \end{cases}$$

$$l(q \bullet job, pst, s) = job$$

$$ta(job, pst, s) = s$$

No behavioral fault hypothesis is done for the queue. The FDEVS model will coincide with the DEVS model since the sets related to faults are not defined. Consider now that this queue is connected to the processor whose FDEVS model (the structure M_P) was given above. The jobs coming out of the queue arrive to the processor and when the processor finishes a job, the output is translated to the queue as a "ready" message.



In absence of structural faults, the structure of the coupling is the following:

$N = \left\langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, Select \right\rangle$, where:

$X = \{job_1, job_2, ..., job_n\}$

$Y = \{y(job_1), y(job_2), ..., y(job_n)\}$

$D = \{P, Q\}$

The structures M_d were defined before.

$I_P = \{Q\}, I_Q = \{P, N\}, I_N = \{P\}$

$Z_{P,Q}(y) = "ready"$

The other translation functions are just identity functions and the function $Select$ can be any function.

Consider now that the connections between the queue and the processor can be broken. The events *QPd* and *QPc* respectively disconnect and reconnect the output of the queue to the input of the processor. The events *PQd* and *PQc* do the same with the connection from de processor to de queue. Then, the associated model to represent this structural faulty behavior is the following:

$$N_{sf} = \left\langle X, Y, D, \{M_d\}, \{I_{d,f}\}, \{Z_{i,d,f}\}, Select_{sf}, d_{fault}, F, X_f \right\rangle$$

$$F = \{QP, PQ, QPPQ, f\}$$

$$X_f = \{QP_d, QP_c, PQ_d, PQ_c\}$$

$$d_{fault}(f, x_f) = \begin{cases} QP & if\ (f = f \wedge x_f = QP_d) \vee (f = QPPQ \wedge x_f = QP_c) \\ PQ & if\ (f = f \wedge x_f = PQ_d) \vee (f = QPPQ \wedge x_f = PQ_c) \\ QPPQ & if\ (f = QP \wedge x_f = PQ_d) \vee (f = PQ \wedge x_f = QP_d) \\ f & if\ (f = QP \wedge x_f = QP_c) \vee (f = PQ \wedge x_f = PQ_c) \\ f & otherwise \end{cases}$$

The influencer sets which are modified by the faults are: $I_{P,QP} = I_{P,QPPQ} = \{\ \}, I_{Q,PQ} = I_{Q,QPPQ} = \{N\}$.

The other sets and functions will not present any modification.

The same faulty behavior could have been defined modifying the translation function instead of the influencer sets (saying for example that the result of some translation function in the presence of some fault is "no event").

CLOSURE UNDER COUPLING OF FDEVS MODELS

The property of closure under coupling allows the hierarchical composition of models. In order to prove that the FDEVS formalism is closed under coupling, the atomic FDEVS model defined by a coupled structure will be constructed.

The coupled structure:

$N = \left\langle X_N, Y_N, D, \{M_d\}, \{I_{d,f}\}, \{Z_{i,d,f}\}, Select, d_{fault_N}, F_N, X_{f_N} \right\rangle$

of the subsystems:

$M_d = \left\langle X_d, S_d, Y_d, d_{int_d}, d_{ext_d}, d_{fault_d}, l_d, ta_d, X_{f_d}, F_d \right\rangle$

defines an FDEVS model:

$M = \left\langle X, S, Y, d_{int}, d_{ext}, d_{fault}, l, ta, X_f, F \right\rangle$ where:

$X = X_N, Y = Y_N$,

$F = (\underset{d \in D}{\times} F_d) \times F_N$, $X_f = (\underset{d \in D}{\times} X_{f_d}) \times X_{f_N}$

$S = \underset{d \in D}{\times} Q_d$ with $Q_d = \{(s_d, e_d)\}$ where $s_d \in S_d$ and e_d is the elapsed time from the last transition of subsystem M_d to the last transition of system N $(e_d \geq 0)$. (It is said that N has a transition when some submodel has a transition).

The fault transition function is $d_{fault}(s, f, e, x_f) = (s', f')$ where:

$x_f = (...., x_{f_d}, ..., x_{f_N})$, where only for component $d=k$ is $x_{f_d} \neq f$

$s = (..., (s_d, e_d), ...)$

$f = (..., f_d, ..., f_N)$

$s' = (..., (s'_d, e'_d), ...)$

$f' = (..., f'_d, ..., f'_N)$

being, $(s'_d, e'_d) = \begin{cases} (s_d, e_d + e) & if \quad d \neq k \\ (s'_k, 0) & if \quad d = k \end{cases}$

and $(s'_k, f'_k) = d_{fault_k}(s_k, f_k, e_k, x_{f_k})$ if $k \neq N$

or $f'_N = d_{fault_N}(x_{f_N}, f_N)$ if $k=N$.

and then $f'_d = \begin{cases} f_d & if \quad d \neq k \\ f'_k & if \quad d = k \end{cases}$

The external transition function is $d_{ext}(s, f, e, x) = (s', f')$ where:

$\forall d / N \in I_{d,f} \wedge x_d \neq f, \quad (s^*_d, f^*_d) = d_{ext\,d}(s_d, f_d, e_d + e, x_d)$ with $x_d = Z_{N,d,f_N}(x)$

and then,

$$(s'_d, e'_d) = \begin{cases} (s^*_d, 0) & if \quad N \in I_{d,f} \wedge x_d \neq f \\ (s_d, e_d + e) & otherwise \end{cases}$$

$$f'_d = \begin{cases} f^*_d & if \quad N \in I_{d,f} \wedge x_d \neq f \\ f_d & otherwise \end{cases}$$

The lifetime function is $ta(s, f) = min\{s_d \mid d \in D\}$, where $s_d = ta_d(s_d, f_d) - e_d$ (time remaining to the next event in submodel M_d).

The internal transition function is $d_{int}(s, f) = (s', f')$ where:

Let $IMM(s, f) = \{d \mid d \in D \wedge s_d = ta(s, f)\}$ be the set of imminents (candidates for next transition).

Let $d^* = Select(IMM(s, f))$ (M_{d^*} is the model whose internal transition function will be executed).

$$(s'_d, f'_d) = \begin{cases} d_{int\,d^*}(s_{d^*}, f_{d^*}) & if \quad d = d^* \\ d_{ext\,d}(s_d, f_d, e_d + ta(s, f), x_d) & if \quad d^* \in I_{d,f_N} \wedge x_d \neq f \\ (s_d, f_d) & otherwise \end{cases}$$

where $x_d = Z_{d^*,d,f_N}(l_{d^*}(s_{d^*}, f_{d^*}))$ and

$$e'_d = \begin{cases} 0 & if \quad d = d^* \vee (d^* \in I_{d,f_N} \wedge x_d \neq f) \\ e_d + ta(s, f) & otherwise \end{cases}$$

Finally, the output function is:

$$l(s, f) = \begin{cases} Z_{d^*,N,f_N}(l_{d^*}(s_{d^*}, f_{d^*})) & if \quad d^* \in I_{N,f_N} \\ f & otherwise \end{cases}$$

ABSTRACT SIMULATOR FOR FDEVS MODELS

A hierarchical structure of simulation will be proposed, following the idea of the DEVS simulator of (Zeigler et al. 2000). This schema of simulation is not the most efficient because it produces an important traffic of messages between the processors associated to each submodel. However it is very simple and it can be carried to a more efficient "flat code".

The FDEVS simulator will be composed by simulators associated to each atomic model and coordinators associated to each coupled structure. A root coordinator manages the global time advance and sends the input events and the fault events to all the simulators and coordinators in the system.

```
FDevs-simulator
variables:
  parent          //parent coordinator
  tl              //time of last event
  tn              //time of next internal event
  FDEVS           //associated model with total state
                    (s, e) and fault state f.
  y               //current output value of the model
when receive i-message  (i, t)  at time t
  tl = t - e
  tn = tl + ta(s,f)
when receive *-message (*, t) at time t
  y = $\lambda$(s,f)
  send y-message (y, t) to parent coordinator
  (s,f) = $\delta_{int}$ (s,f)
  tl = t
  tn = tl + ta(s,f)
when receive x-message (x, t) at time t with input value x
  e = t - tl
  (s,f) = $\delta_{ext}$ (s,f,e,x)
  tl = t
  tn = tl + ta(s,f)
when receive $\phi$-message (xf, t) at time t with value xf
  e = t - tl
  (s,f) = $\delta_{fault}$ (s,f,e,xf)
  tl = t
  tn = tl + ta(s,f)
end FDevs-Simulator
```

```
FDevs-coordinator
variables:
  FDEVN           //the associated network
                    with fault state f
  parent          // parent coordinator
  tn              // time of next internal event
  event-list      // list of elements (d, $tn_d$)
  d*              // selected imminent child
when receive i-message  (i, t)   at  time t
  for-each d in D do
        send i-message (i, t) to child d
  tn = min { $tn_d$ | $d \in$ D}
when receive *-message  (*, t)  at  time t
  d* = Select{d/ $tn_d$ = min {$tn_j$ | $j \in$ D}}
  send *-message (*, t)  to d*
  tn = min {$tn_d$ | $d \in D$}
when receive x-message (x, t)
  receivers = {r | $r \in D$, $N \in I_{r,f}$, $Z_{N,r,f}(x) \neq \Phi$}
  for-each r in receivers
        send x-messages ($x_r$, t) with $x_r = Z_{N,r,f}(x)$ to r
  tn = min {$tn_d$ | $d \in D$}
when receive y-message ($y_{d^*}$, t) from d*
  if  $d^* \in I_{N,f}$ & $Z_{d^*,N,f}(y_{d^*}) \neq \Phi$ then
        send y-message ($y_N$, t)  with
                    $y_N = Z_{d^*,N,f}(y_{d^*})$ to parent
  receivers = {r | $r \in D$, $d^* \in I_{r,f}$, $Z_{d^*,r,f}(y_{d^*}) \neq \Phi$}
  for-each r in receivers
        send x-messages ($x_r$, t) with $x_r = Z_{d^*,r,f}(y_{d^*})$ to r
when receive $\phi$-message  (xf, t)
  f = $\delta_{fault}$ (f,xf)
end FDevs-coordinator
```

```
FDevs-root-coordinator
variables:
t                 //current simulation time
```

```
child              //direct subordinate
input-event-list   //list of input events ($x_i$, $tx_i$) sorted by $tx_i$
fault-event-list   //list of fault events ($o_k$, $xf_k$, $tf_k$) sorted by $tf_k$
$t = t_0$
$i$=1
$k$=1
send initialization message ($i$, $t$) to subordinate
$t$ =min {$tn_{child}$ , $tx_i$ , $tf_k$ }
loop
  if $t$ =$tn_{child}$ then
        send *-message (*, $t$) to child
  elseif $t$ = $tx_i$ then
        send x-message ($x_i$, $t$) to child
        $i$=$i$+1
  elseif $t$ = $tf_k$ then
        send ϕ-message ($xf_k$, $t$) to object $o_k$
        $k$=$k$+1
  $t$ =min {$tn_{child}$ , $tx_i$ , $tf_k$ }
until end of simulation
end FDevs-root-coordinator
```

CONCLUSIONS

The main contribution of the paper is FDEVS, a very general DEVS-based formalism for fault modeling and simulation in discrete event systems, allowing different kinds of fault hypothesis and automatic fault generation. It has been shown that FDEVS permits the representation of faulty behavior in large, complex systems, via the hierarchical composition of faulty models of their low-level components or subsystems. An associated hierarchical, abstract simulation structure has also been proposed in this article.

In some application fields, a particularization of FDEVS featuring predefined tools close to the particular problems of the domain could be more useful, because improving the easy of manipulation by end-users. In many applications, users are only interested in fault simulation and not in fault generation; in some other cases, only few classes of faults are of interest. In order to simplify the work of end-users, current research is oriented to the definition of FDEVS-subformalisms, via the proposition of specific functions and tools to simplify the description of faulty behaviors. In particular, current work on these points focuses on:

- obtaining different formalisms, sub-sets of FDEVS, oriented to classical classes of fault simulation problems (simulation of stuck-at-fault, for instance)
- developing a problem-oriented library of application tools
- developing a procedure for flattening user descriptions in order to accelerate simulations through the use of a one-level discrete-event technique.

REFERENCES:

Larsson, M. 1999. “Behavioral and Structural Model Based Approaches to Discrete Diagnosis”. Linkoping Studies in Science and Technology. Dissertations, No. 608.

Giambiasi, N.; J.F.Santucci and A.L.Courbis. 1991. “Test pattern generation for behavioral descriptions in VHDL”. In *Proceedings of the Euro-VHDL 91*, Stockholm, Sweden.

Santucci, J.F.; A.L.Courbis and N.Giambiasi.1993. “Behavorial testing of digital circuits”, *Journal of Microelectronic System Integration*, Vol. 1, N° 1, S. K. Tewsbury Editor, Plenum Press, New York.

Abramovici, M.; M.A.Breuer and A.D.Friedman. 1990 “*Digital Systems Testing and Testable Design*”. Computer Science press .

Zeigler B. 1984. “*Multifaceted Modeling and Discrete Event Simulation*”. Academic Press, London.

Zeigler B. 1989. “DEVS representation of dynamical systems”, *Proceedings of the IEEE*, Vol. 77, pp. 72-80.

Giambiasi, N. and M.Sigal. 1982. “A deductive Method for Non Classical Logic Faults Simulation”, In *Proceedings of ICCC 82*, New York.

Perret, J. and N.Giambiasi. 1998. "Fault Modelling and Fault Simulation using Statecharts". In *Proceedings of AVCS*, Amiens, France.

Giambiasi, N.; C. Frydman and M. Smaili. 1996. "Event Driven Simulation with Fuzzy Delay", *Artificial Intelligence, Simulation, and Planning in High Autonomy Systems*, La Jolla, USA.

Zeigler, B., H.Praehofer and T.G.Kim. 2000. “*Theory of Modeling and Simulation second ed.*”. Academic Press, New York.