Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

INTRODUCING VARIABLE PORTS AND MULTI-COUPLINGS FOR CELL BIOLOGICAL MODELING IN DEVS

Adelinde M. Uhrmacher
Jan Himmelspach
Mathias Röhl
Roland Ewald

Institute of Computer Science
University of Rostock, 18051 Rostock, Germany

ABSTRACT

Motivated by the requirements of molecular biological applications, we are suggesting an extension of the DEVS formalism. Starting with DYNDEVS a reflective variant of DEVS which supports dynamic behavior, composition, and interaction pattern, we develop ρ-DEVS. Dynamic ports and multi-couplings are introduced whose combination allows models to reflect significant state changes to the outside world and enabling or disabling certain interactions at the same time. An abstract simulator describes the operational semantics of the developed formalism, and the Tryptophan operon model illustrates the developed ideas and concepts.

1 INTRODUCTION

Variable structure models lend structure to the temporal dimension in describing systems. As the relation of bisimulation has been shown to hold between the original DEVS (Discrete Event Systems Specification) formalism (Zeigler, Praehofer, and Kim 2000) and its variable structure variants (Barros 1997, Uhrmacher 2001), the question arises what benefit to expect from such formalisms and consequently tools. Mostly, it is less the question *whether* a formalism is able to express certain phenomena, but *how* easily this can be done (Kuttler and Uhrmacher 2006). Thus, a bit of a subjective flavor in favoring one or another formalism remains. Variable structures in modeling and simulation traditionally refer to the change of behavior pattern, the change of interaction structure, and the change of composition. Only recently (Hu, Zeigler, and Mittal 2005, Uhrmacher and Priami 2005), the change of interfaces has been discussed in the context of system theoretical approaches toward modeling and simulation. This is not surprising, as in systems theory the distinction between system and environment, and maintaining this distinction, is traditionally emphasized. Thus, a system seems more likely to change its composition, its interaction structure and its behavior pattern, than its interface to its environment, although it might change its communication partners (Uhrmacher 2001). However, some systems are characterized just by that: a plasticity of their interface. Thereby, they signalize significant changes to the external world. These phenomena can particularly be found in the molecular biological domain, where enzymes and proteins change their interface and thereby restrict the type of possible interaction partners.

We will follow this line of thought in introducing variable ports into DYNDEVS (Uhrmacher 2001) and combining them with special types of couplings. To introduce this new formalism, i.e., ρ-DEVS, we will move bottom up the model and the simulator hierarchy. The first part will be dedicated to atomic models and their simulators, the second part to coupled models and their coordinators. A Tryptophan operon model will illuminate the specific features of the introduced formalism. Related work will be discussed before concluding the paper.

2 ρ-DEVS – AN EXTENSION OF DYNDEVS

Starting point for our discussion is DYNDEVS (Uhrmacher 2001). Based on DEVS (Zeigler, Praehofer, and Kim 2000), the DYNDEVS formalism has been developed for describing models whose description entails the possibility to change their own state and behavior pattern. Therefore, model and network transitions, which map the current state of a model into a set of models the model belongs to, have been introduced. Thereby, sequences of models are produced. The idea of DYNDEVS is that models are interpreted as a set of models that are successively generating themselves by model transitions.

Each element of the set represents an incarnation of the model and describes a phase of the evolving modeled dynamic system. Unlike other approaches that emphasize the distinction between controlling and controlled unit, e.g., (Barros 1997), the formalism supports models which adapt their own interaction structure and their own behavior. However, DYNDEVS assumes a static set of ports. To introduce variable ports into DYNDEVS, two different strategies can be followed: one is to support an entirely unconstrained change of ports, so nothing is known about the interface, and the set of models are clued into one unity by the hidden internal continuity of processing. Another possibility would be to specify the set of possible ports in advance, and dynamically determine the set of currently activated ports. We will pursue the first line of thought. Another difference is that whereas DYNDEVS is based on sequential DEVS, ρ-DEVS will be based on PDEVS, which supports time-triggered and situation-triggered events that occur at the same simulation time to be processed concurrently. Furthermore, a port type is introduced which contains the information to realize variable structures at coupled model levels via the network transition function ρ_n, more about this in section 5. The newly introduced function λ_ρ is feeding this port.

Definition 1 (ρ-DEVS) *An atomic ρ-DEVS model is the structure $\langle m_{init}, \mathcal{M}, X_{sc}, Y_{sc}\rangle$ with $m_{init} \in \mathcal{M}$ the initial model, X_{sc}, Y_{sc} ports to communicate structural changes, $\mathcal{M}$ the least set with the following structure: $\langle X, Y, S, s_0, \delta_{int}, \delta_{ext}, \delta_{con}, \rho_\alpha, \lambda_\rho, \lambda, ta\rangle$, where*

X, Y structured sets of inputs and outputs
S structured set of states
$s_0 \in S$ initial state
$\delta_{int} : S \rightarrow S$: internal transition function
$\delta_{ext} : Q \times X^b \rightarrow S$: external trans. function, with
$Q = \{(s,e) : s \in S, 0 \leq e < ta(s)\}$ state set including elapsed time
$\delta_{con} : S \times X^b \rightarrow S$: the confluent transition function
$\lambda : S \rightarrow Y$: the output function
$ta : S \rightarrow \mathcal{R}^{\geq 0} \cup \{\infty\}$: the time advance function
$\rho_\lambda : S \rightarrow Y_{sc}$: implied structural network changes
$\rho_\alpha : S \times X_{sc} \rightarrow \mathcal{M}$: model transition

and $\mathcal{M}$ is the least set for which the following reachability property holds $\forall n \in \mathcal{M}$:

$$n = m_{init} \vee \exists m_0 = m_{init}, \ldots, m_i = n \wedge \rho_\alpha(s^{m_k}) = s^{m_{k+1}}$$

with $i > 0$, $k = 0, \ldots, i-1$, $s^{m_k} \in S^{m_k}$, $s^{m_{k+1}} \in S^{m_{k+1}}$, $m_0, \ldots, m_i \in \mathcal{M}$.

ρ-DEVS defines inputs, outputs, and states, i.e., X, Y, S, as structured sets, which are structured according to a set of variable names, which, in the case of input and output, denote the *ports* by which inputs are received and outputs are launched. As in DYNDEVS, the model transition ρ_α does not interfere with other transitions, it preserves the values of variables which are common to the states of two successive model incarnations and assigns "default initial" values to the "new" variables (see Uhrmacher 2001). X^b denotes a bag of inputs, as several inputs might arrive concurrently, n and m_k etc. denote incarnations of the model set $\mathcal{M}$. In comparison to DYNDEVS, input and output ports in ρ-DEVS are becoming part of the incarnations and thus can be changed via the model transition. In addition, a special type of port has been introduced. The role of ρ_λ is to fill the port for structural changes, Y_{sc}, that shall occur at the level of the network model. This information will be accessed by the network transition function ρ_n of the parent coupled model (see section 5). As other models might induce structural changes that have an effect on the model incarnation, also an input port X_{sc} has been introduced for these types of requests. This input is considered in applying ρ_α and determining the new model incarnation.

3 SIMULATOR FOR ρ-DEVS ATOMIC MODELS

Following the tradition of DEVS, a simulator is associated with each atomic model. The simulation proceeds by propagating pulses top-down and bottom-up the hierarchy. First, the processes that have an internal or confluent event are activated by a $*$ message, the outputs are generated and propagated bottom-up the hierarchy. Afterward, an xy message is propagated top-down, triggering the execution of events. After the state transitions have been executed, a structure change message sc is generated and propagated bottom-up. A top-down sent sc message will trigger the execution of the model transition ρ_α, which might or might not lead to a new model incarnation. In case no structural changes are implied, the model transition will return the old model incarnation (Uhrmacher 2001).

Algorithm 1 Pseudo Code of ρ-DEVS Simulator

```
when receive *, xy or sc message
  if message is * or xy message
    if message is * message
      $xy$ := $m.\lambda(s)$
      send xy message to parent
      wait for receive xy message
      if isEmpty (xy message)
        $m.s$ := $m.\delta_{int}(m.s)$
      else
        $m.s$ := $m.\delta_{con}(m.s, xy, t - tole)$
      fi
    else
      $m.s$ := $m.\delta_{ext}(m.s, xy, t - tole)$
    fi
    $yr$ := $m.\rho_\lambda(m.s)$
    send sc(yr) message
  fi
  wait for $sc$ message
  $m$ := $m.\rho_\alpha(m.s, sc)$
  tonie := t + ta(m.s)
  tole := t
  send done (XYports(m), tonie) message
end when
```

Each simulator has a link to the current incarnation of the model m (4). In addition it keeps track of the time by storing the time of last event $tole$ and time of next event $tonie$ (21,20). After executing internal, external, and confluent event of the current incarnation of the model, it is checked whether a structural request concerning the network structure has to be launched by invoking $m.\rho_\lambda$ (15). The structural output is sent to the parent (16). After having received structural input, the model transition function ρ_α is invoked to generate the new incarnation (if necessary), which will include also new input and output ports (19). The later is an information which is also of interest for the network, same as the time of next event. The done message includes the time of next event and the set of ports of the current incarnation. Before introducing the coupled model of ρ-DEVS, let us first see how a concrete model would look like. Therefore, we will introduce a biological example.

4 THE EXAMPLE - THE TRYPTOPHAN OPERON

The Tryptophan (Trp) operon is one of the most extensively studied systems for the examination of the prokaryotic gene regulation including the spatio-temporally progressing phenomena of gene repression, transcriptional attenuation, and post translational enzyme feedback inhibition. In the following, we will focus on the gene repression (Figure 1). The Trp operon comprises a promoter and an operator region and genes responsible for coding of the 5 enzymes that are needed to synthesize Trp. The promoter region has a binding site where the RNA polymerase can bind. This is the enzyme responsible for the transcription of genes. As long as the repressor is inactive, it cannot bind to the operator of the Trp operon. However, if the corepressor Trp binds to the inactive repressor protein, the allosteric repressor protein will change its shape and become activated. In this state, the repressor protein can bind to the operator region of the operon. With the active repressor protein bound to the operator region, RNA polymerase is unable to bind to the promoter region of the operon. Thus, the transcription of the five genes into mRNA will be disabled.

4.1 Repressor Model in Devs

In the following, we will describe the repressor model in classic DEVS. Its input and output ports allow it to receive Trp molecules and to dock to the operon (Figure 2, line 2,3). It is statically coupled to an operon model and to a cytoplasm model. The latter contains and updates the Trp molecules. An repressor's state reflects the basic states, being activated and not being activated (free), trying to dock, and repressing (6). A time period is associated with each phase. For example, the repressor remains in the state free until an external event occurs. With the phase repressing, i.e., repr, a time span is associated which is calculated

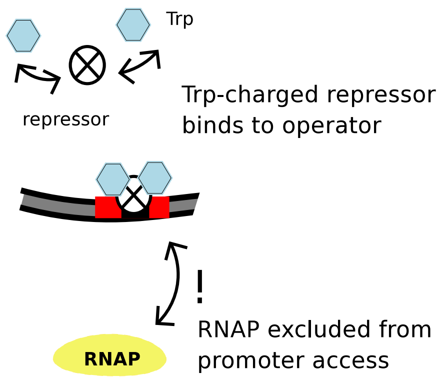
Trp
repressor
Trp-charged repressor binds to operator
!
RNAP
RNAP excluded from promoter access

Figure 1: Transcriptional Repression

as an exponential distribution of the mesoscopic constants, i.e., `expRand(reprT)`, according to Gillespie (Gillespie 1976) (36). In the phase active, i.e., act, backward and forward reaction are competing with each other (11,17,19). If the backward reaction wins, the Trp is released into the cytoplasm, if the forward reaction wins it turns in the state trying to dock (20) and sends a docking request to the operon via the lambda function (44).

In the biological system, no Trp molecule will bind to the repressor unless the repressor is able to accept it. However, this information, e.g., about being free, repressed, or activated, is stored as part of the internal state of the repressor model and is not visible from outside. Produced outputs are broadcasted in DEVS, i.e., if several input ports are coupled to one output port, the produced output, i.e., Trp, is cloned and will reach all input ports. Thus, the Trp is handled in DEVS as a non-consumable information. To ensure a correct processing, the cytoplasm model has to keep track of the repressors and their states. The cytoplasm will select randomly one from the group of repressors being in phase free to which the Trp will be directed. So even though the Trp messages is sent to all repressors, only one will be the correct addressee. This information is included in the Trp event (29). The same applies to the docking request (36).

Thus, each Trp sent and each docking accepted leads to $n-1$ unnecessary transitions if n repressors exist. This decreases the performance of the simulation significantly. In addition, the cytoplasm model has to maintain an internal model about its environment. It has to store a list of repressors with their names and the information whether a repressor is currently free and able to accept Trp. This puts an additional and unintuitive burden on the modeling. In the new formalism, this additional effort in modeling and simulation can be avoided.

```
Repressor :=
  X := {(trpIn, opIn) | ... }
  Y := { (trpOut, opOut) | ... }
  S := { ($\sigma$, forwT, backT) |
         $\sigma$ $\in$
           {free, act, tryToDock, repr}
        forwT, backT $\in \mathcal{R}$ }
  s_0 := (free, $\infty$, $\infty$)

  ta ($\sigma$, forwT, backT) :=
    if $\sigma$ = act then min(forwT, backT)
    else forwT

  $\delta_{int}$ ($\sigma$, forwT, backT) :=
    case $\sigma$ of
      act:
        if backT < forwT then
          (free, $\infty$, $\infty$)
        else
          (tryToDock, $\epsilon$, $\infty$)
      tryToDock:
        (act, expRand(FT), expRand(BT))
      repr:
        (act, expRand(FT), expRand(BT))
    end case

  $\delta_{ext}$ (($\sigma$, forwT, backT), elapT, X) :=
    if X = (trp, nil) then
       if $\sigma$ = free $\wedge$ trp.name = myName then
         (act, expRand(FT), expRand(BT))
       else
         ($\sigma$, forwT-elapT, backT-elapT)
    else
      if X = (nil , dockAccepted) then
        if $\sigma$ = tryToDock $\wedge$
          dockAccepted.name = myName  then
         (repr, expRand(reprT), $\infty$)
        else
         ($\sigma$, forwT-elapT, backT-elapT)

  $\lambda$ ($\sigma$, forwT, backT) :=
      case $\sigma$ of
        act:
          if backT < forwT then (Trp, nil)
          else (nil, dock?)
        repr: (nil, undock!)
      end case
```

Figure 2: Repressor Model in Classic DEVS

4.2 Repressor Model in ρ-DEVS

Based on the different phases of the model, we can distinguish between the following incarnations:

freeInc in which the repressor does not produce any outputs but waits for the arrival of Trp molecules,

actInc in which the repressor cannot receive any Trp (because they are already bound), and in which it can release Trp to the cytoplasm and is able to communicate with the operon

tryDockInc the short phase of waiting for an acceptance from the operon, if accepted, the repressor will start repressing, if not it falls back to the state of being activated

reprInc where the repressor is docked to the operon

In this case, we have four distinguished phases, and the repressor is defined by
repressor :=
$\langle freeInc, \{freeInc, actInc, tryDockInc, reprInc\}, X_{sc}, Y_{sc}\rangle$
with $X_{sc} := \emptyset, Y_{sc} := \emptyset$ and $freeInc$, $actInc$, $tryDockInc$, and $reprInc$ being structures:
$\langle$ X, Y, S, s_0, $\delta_{int}, \delta_{ext}, \delta_{con}, \rho_\alpha, \rho_\lambda, \lambda, ta\rangle$.

Please note that the output function ρ_λ is not needed, nor are the input and output ports for variable structure messages (X_{sc}, Y_{sc}). Functions that are not needed do not appear in the example and are assumed to be undefined, $\rho_\alpha(s, nil)$ is abbreviated by $\rho_\alpha(s)$. δ_{con} is defined by a successive execution of the internal and external transition function, i.e., $\delta_{ext}(\delta_{int}(s), xy)$ by default.

```
$freeInc$ := $\langle$
  X   := { (trpIn) | trpIn $\in$ {trp, nil} }
  Y :=  $\emptyset$
  S :=  { ($\sigma$) | $\sigma$ $\in$ {free, act} }
  $s_o$ := (free)
  ta($\sigma$) := $\infty$
  $\delta_{ext}$($\sigma$, elapT, trp) := act
  $\rho_\alpha$($\sigma$) := if $\sigma$ = act then $actInc$
$\rangle$
```

Figure 3: Repressor Model in ρ-DEVS - Incarnation "freeInc"

In the first phase (Figure 3) only input ports are available in order to let Tryptophan dock to the repressor.

In the activated phase (Figure 4), no Tryptophan can dock to the repressor, however, the repressor can release Trp to the cytoplasm and can announce its docking request to the operon. In the first case, it will reach the phase free again. In the latter, it will change its phase to tryToDock and switch to incarnation tryDockInc, respectively.

The phase tryToDock is a phase only introduced for synchronization (Figure 5). Although the readiness of the repressor to dock is signalized by generating the input port, i.e., opIn, the dynamics of docking is part of the repressor and thus initiated by it, so it sends via its output ports a docking request (Figure 4, line 20), which might or might not reach the operon. The latter is the case if the operon's input port, repIn, is not available since another repressor is already bound to the operon. If the operon is accepting the docking request, the repressor will move into the phase repressing and switch to incarnation $reprInc$ (Figure 6). The

```
$actInc$ := $\langle$
   X := $\emptyset$
   Y :=   { (trpOut, opOut) |
              trpOut $\in$ { trp, nil },
              opOut $\in$ { dockRequest, nil }}
   S := { ($\sigma$, forwT, freeT) |
              $\sigma$ $\in$ { act, tryToDoc, free }
              forwT, freeT $\in$ R
   s$_o$ := (act, expRand(FT), expRand(BT))
   ta ($\sigma$, forwT, freeT) :=
      if $\sigma$ = act then min(forwT, freeT)
   $\delta_{int}$ ($\sigma$, forwT, freeT) :=
      if freeT < forwT then (free, $\infty$, $\infty$)
      else (tryTodock, $\infty$, $\infty$)
   $\rho_\alpha$ ($\sigma$, forwT, freeT) :=
      if $\sigma$ = free then $freeInc$ else $tryDockInc$
   $\lambda$ ($\sigma$, forwT, freeT) :=
      if $\sigma$ = act $\wedge$
         freeT < forwT then (trp, nil)
      else (nil, dockRequest)
$\rangle$
```

Figure 4: Repressor Model in ρ-DEVS - Incarnation “actInc”

```
$tryDockInc$ := $\langle$
   X := { (opIn) | opIn $\in${dockAccepted, nil }}
   S := { ($\sigma$) |
              $\sigma$ $\in${tryToDock, repr, act }}
   s$_o$ := (tryToDock)
   ta ($\sigma$) := $\epsilon$
   $\delta_{int}$ ($\sigma$) := (act)
   $\delta_{ext}$ (($\sigma$), elapT, dockAccepted) := (repr)
   $\rho_\alpha$ ($\sigma$) :=
      if $\sigma$ = repr then $repInc$
      if $\sigma$ = act then $actInc$
$\rangle$
```

Figure 5: Repressor Model in ρ-DEVS - Incarnation “tryDockInc”

docking could be modeled more intuitively if the formalism would support a synchronous communication, as e.g., is the case in the stochastic π-CALCULUS. However, interactions between models described in the DEVS formalism and its variants are traditionally asynchronous.

The model could be easily visualized as a composite state automaton, which emphasizes the structuring aspect of the temporal plane. However, the set of model incarnations is not necessarily finite and can not always be as easily predefined as in the above case. This becomes apparent at the level of the coupled model of the Tryptophan model (see section 7).

```
$reprInc$ := $\langle$
   X  := $\emptyset$
   Y  := { (opOut) | opOut $\in$ {undock, nil} }
   S :=  { ($\sigma$) | $\sigma$ $\in$ { repr, free } }
   s$_o$ := (repr)
   ta ($\sigma$) := expRand(reprT)
   $\delta_{int}$ ($\sigma$) := (act)
   $\lambda$ ($\sigma$) := (undock!)
   $\rho_\alpha$ ($\sigma$) := if $\sigma$ = act then $actInc$
$\rangle$
```

Figure 6: Repressor Model in ρ-DEVS - Incarnation “reprInc”

5 ρ-DEVS NETWORKS

To support the hierarchical and modular modeling, we introduce a network structure in ρ-DEVS. A structural change means a change of interaction and composition structure as in DYNDEVS. In addition, ports can change and multi-couplings are introduced.

Definition 2 (ρ-Devs Networks) *A reflective, higher order network, a ρ-NDEVS, is the structure $\langle n_{init}, \mathcal{N}, X_{sc}, Y_{sc}\rangle$ with $n_{init} \in \mathcal{N}$ the start configuration, X_{sc}, Y_{sc} ports to communicate structural changes, and $\mathcal{N}$ the least set with the following structure $\{\langle X, Y, C, MC, \rho_N, \rho_\lambda\rangle$ where*

X	*set of structured inputs*
Y	*set of structured outputs*
C	*set of components which are of type ρ-DEVS*
MC	*set of multi-couplings*
$\rho_N : \mathbf{S}^n \times X_{sc} \rightarrow \mathcal{N}$	*network transition*
$\rho_\lambda : \mathbf{S}^n \rightarrow Y_{sc}$	*structural output function*

with $\mathbf{S}^n = \times_{d\in C} \oplus_{d\in C} Y^d_{sc}$ and $\mathcal{N}$ is the least set for which the following reachability property holds $\forall n \in \mathcal{N}$:

$$n = n_{init} \vee \exists n_0 = n_{init}, \ldots, n_i = n \wedge \rho_\alpha(s^{n_k}) = s^{n_{k+1}}$$

with $i > 0$, $k = 0, \ldots, i-1$, $s^{n_k} \in S^{n_k}$, $s^{n_{k+1}} \in S^{n_{k+1}}$, $n_0, \ldots, n_i \in \mathcal{N}$.

In addition, similar to the definition in DYNDEVS, the ρ-NDEVS has to satisfy the following constraint: The application of ρ_N preserves the state and structure of models which belong to the composition of the “old” network and the “new” one. C is the set of components. Components which are newly created are initialized. The initial state of a component is given by the model n_{init} being in its initial state s_{init} (Uhrmacher 2001). The outputs Y^d_{sc} of the components in C form the quasi-state the structural output function ρ_λ and the network transition ρ_N are based upon. The structural output function defines, given the component's structural outputs, what shall be made available to the coupled model further up, and the network transition takes this information as well as structural input information X_{sc} into account to determine the next network incarnation.

Another difference to DYNDEVS is the introduction of *multi-couplings*. The idea behind multi-couplings is to make use of the information of the components' available ports and allow a dynamic coupling between models. In this definition, the names of ports become central. Couplings are defined as 1:n, n:1, or n:m relationships between sets of components. Taking part in these couplings is based on the availability of ports.

Definition 3 (Couplings) *A multicoupling $mc \in MC$ is defined as a tuple*

$$\begin{aligned} mc \ = \ \langle \ & \{(C_{src}.port) | C_{src} \in C\}, \\ & \{(C_{tar}.port) | C_{tar} \in C\}, \\ & select \ \rangle \\ \text{with } & select : 2^C \rightarrow 2^C. \end{aligned}$$

As couplings are directed we distinguish between the components that form the source or the target of events, i.e., C_{scr}, C_{tar}. The existence of ports, i.e., $port$, implies the existence of couplings. The function select determines how the values are distributed. If more than one input port are linked to an output port in regular DEVS, each output will be cloned and sent to all connected input ports. This standard strategy is meaningful if information shall be broadcasted, however, it is not a good strategy for consumable resources like molecules. For the latter, a random selection strategy can now be utilized. For example, we combine the output port of the cytoplasm model responsible for delivering Tryptophan, i.e., cyt.trpOut, with all repressors that are currently able to bind Tryptophan, which is reflected by having an input port to accept Tryptophan, i.e., r.TrpIn, and assign a random strategy:

mc = ⟨ { cyt.trpOut }, { r.trpIn } |
r ∈ $\{rep_1, \ldots, rep_n\}$}, ranSelOne ⟩.

In the moment a Trp is launched via the output port cyt.trpOut, one of the repressors currently connected via the input port trpIn will be chosen randomly as its addressee. Which of the repressors are currently connected is determined by the existence of the input port trpIn.

6 COORDINATOR FOR ρ-DEVS

With each coordinator an incarnation of the current network is associated, i.e., n. Unlike in DYNDEVS, port information has to be updated and the network transition is based on the information provided by the output ports $ysc_1, ...ysc_n$ of its components. Also structural change requests can propagate through top down and bottom up the hierarchy (see also Himmelspach and Uhrmacher 2004). Requests

Algorithm 2 ρ-DEVS Coordinator

```
when receive *, x or sc message
  if message is * or x message
    if message is * message
      send * to d ∈ IMM = {c ∈ n.C|tonie(c)=t}
      wait for $xy_d$ messages from all children
          send xy$_{toParent}$ message according to
            MC
      wait for xy$_{fromParent}$ message from parent
    fi
    forward xy messages  to
    $\cup xy_d \backslash$xy$_{toParent}\cup$xy$_{fromParent}$ to
       d ∈ IMM ∪ INF according to MC
    wait for $sc_d$ message from all children
    yr := n.$\rho_\lambda(ysc_1, ...ysc_m)$
    send sc(yr) to parent
  fi
  receive sc message from parent
  send sc message to children
  wait for done messages from any activated
      children
  n := n.$\rho_N$($ysc_1, ...ysc_m$, sc)
  tonie := t + minimum (tonie$_n$)
  tole := t
  send done(XYports(n), tonie) message
end when
```

that are not directed to the network itself are sent to the coordinator's parent, so it can be processed by the next higher level. The role of the function ρ_N is to take the different requests from the components, which are available via their structure change ports Y_{sc}, and the input that has reached the coordinator via X_{sc}, and to invoke, based on this information, the network transition, producing a new network incarnation. Afterward, the time of next event and the set of ports associated with the network incarnation is sent to the parent via the done message.

7 THE TRP OPERON MODEL

The coupling between cytoplasm, the individual repressor models, and mRNA models are realized by a special type of coupling: multi-couplings which randomly select an input port each time an output is generated by the cytoplasm. Cytoplasm, Enzyme, Operon, and mRNA are all coupled models, so the model is hierarchically structured. In contrast to earlier versions of the model (Degenring, Röhl, and Uhrmacher 2004), we do not use a micro model to describe the individual enzymes, but describe the pool of Tryptophan enzymes at macro level, by one model that contains all enzymes. The number of repressor models is constant, however, the number of mRNA models variates over time.

The model of the Trytophan can be described at the level of the network as a sequence of network incarnations: $\langle n_1, \{n_1, n_2, n_3 \ldots\}, X_{sc}, Y_{sc}\rangle$ with X_{sc}:=$\emptyset$, Y_{sc}:=$\emptyset$ and n_i structured by $\langle X, Y, C, MC, \rho_N, \rho_\lambda\rangle$. Due to the introduction of multi-couplings the network structure

```
$n_x$ := ⟨ X:= nil, Y:= nil
  C:= { rep$_1$, ... rep$_n$, mRNA$_1$, ... mRNA$_x$, enz,
      op, cyt }
  MC:=
   {⟨ {cyt.rnapOut},{op.rnapIn},default⟩,
    ⟨{cyt.IGPOut},{enz.IGPIn},default⟩,
    ⟨{op.rnapOut},{cyt.rnapIn}, default⟩,
    ⟨{mrna.trpOut |
      mrna ∈ {mRNA$_1$ ... mRNA$_x$}},
      {cyt.trpIn}, default⟩),
    ...
    ⟨{r.trpOut |
      r ∈ {$rep_1$ ... $rep_n$}},{cyt.trpIn},default⟩,
    ⟨{cyt.trpOut}, {r.trpIn} |
                    r ∈ {$rep_1$,...,$rep_n$}},
                        ranSelOne}⟩,
    ⟨{r.opOut | r ∈ {$rep_1$ ... $rep_n$}},
     {op.repIn}, default⟩,
    ⟨{op.repOut,
     {r.opIn | r ∈ {$rep_1$,...,$rep_n$}}, ranSelOne}⟩
 $\rho_n$ :=
         if ⟨ add, C, ?mRNA ⟩ then $n_{x++}$
         if ⟨ remove, C, ?mRNA ⟩ then $n_{x--}$
```

Figure 7: Network Incarnation n_x

only changes with the amount of mRNA available (Figure 7).

Because of the multi-coupling, we can now distinguish between information being broadcasted and consumable resources being exchanged between models. Maintaining endomorphic models, i.e., models about models, is no longer needed as internal states are signalized by ports being available to the environment. At the level of simulation, events are no longer processed unnecessarily. However, an overhead is induced due to propagating the variable structure requests top-down and bottom-up the processor hierarchy and updating the internal lists of couplings and components. Thus, depending on the model, the efficiency gain in simulation might be lost or even inverted. However, these cases are difficult to predict, as they depend not only on model structure and dynamics, but also on the actual implementation and composition of the simulation engine (Himmelspach and Uhrmacher 2006). In the worst case, the benefit of the proposed approach will be restricted to facilitate modeling.

8 DISCUSSION AND RELATED WORK

Traditional DEVS does not support variable interaction, composition, and behavior pattern, although DEVS was probably one of the first modeling formalisms for simulation in which the need to support variable structures was emphasized (Zeigler 1986). Meanwhile, a variety of extensions of the formalisms exist, e.g., (Pawletta et al. 1996, Barros 1997, Uhrmacher 2001), which support the change of structure as specific events, and likely even more implementations support variable structures. Prominent in DEVS is the clear distinction between system and environment by encapsulating the attributes and methods of a model. The interaction of a model with its surrounding models is based on static ports, even in DEVS variants that support variable structures.

However, recently the introduction of variable ports has been suggested (Uhrmacher and Priami 2005, Hu, Zeigler, and Mittal 2005). In (Hu, Zeigler, and Mittal 2005), to facilitate a selective 1:1 communication between workflow model components, Zeigler and his colleagues showed an example implementation of variable interfaces in DEVSJAVA. To help maintaining a consistent modeling and simulation in DEVSJAVA, operational boundaries have been defined as discussed in (Uhrmacher 2001). Zeigler and his colleagues allow a model component to change interfaces of any model that resides in the same coupled model. In our approach this is not necessary. A comfortable modeling of changing interactions due to changing interfaces has been ensured by introducing multi-couplings.

The traditional couplings as they are supported in DEVS are suitable if a structures like the Tryptophan Synthase enzyme shall be modeled. The alpha and beta subunit of the Tryptophan Synthase form a covalent structure and interact directly via a 1:1 coupling. Whereas the beta unit signalizes the availability of serine to alpha, the alpha subunit tunnels the produced Indole via the tunnel to beta. Not only this static channeling but also dynamic channeling, e.g., observed in the glycolysis, require a direct communication which is nicely reflected by the standard coupling in DEVS (Degenring, Röhl, and Uhrmacher 2004). However, if more than one model is connected to the output of a model the output will be cloned and reach all. This is similar to the idea of broadcasting events in STATECHARTS (Harel and Naamad 1996). If information is communicated, this makes perfect sense, however, less so when consumable resources like Tryptophan molecules are communicated along couplings. The question is whether DEVS provides us with any means to solve this problem. In (Zeigler, Praehofer, and Kim 2000), Zeigler introduces the Z function which translates the output values of an output port into values for an input port. The set of couplings to which a specific Z is applied is extensionally defined, which hampers an efficient use in combination with dynamic ports. In addition, the intention of introducing Z is type conversion between output and inputs rather than different interaction semantics. The latter is the purpose of the introduced multi-couplings in combination with variable ports.

For some formalisms, variable ports do not come as an extension but are natural, e.g., in π-CALCULUS (Priami 1995). It is based on *names* and uses a small set of operators to create terms which are referred to as *processes*. Two concurrent processes interact using a name they share the knowledge of. Interactions occur using a name, over which one process acts as a sender, while the other receives. The message being transmitted is again a name, which the receiver henceforth knows and may use in further interactions. Such message-passing allows to describe networks with evolving connectivity in an elegant manner. Our introduction of multi-couplings and variable ports in ρ-DEVS was largely inspired by the concepts of π-CALCULUS and the comparison of the two formalisms (Uhrmacher and Priami 2005). Even more importance to interfaces is given in current extensions of the stochastic π-CALCULUS, like BETA-BINDERS (Priami and Quaglia 2005). BETA-BINDERS is a formal language that merges the basic features of classical process calculi with the intuition that, in order to better model biological entities, simple concurrent processes can be wrapped by borders with explicit interaction sites. The enclosing borders mimic biological membranes and are equipped with typed sites that resemble the motifs of molecules. Here, the possibility to interact does not depend on the announced names of the sites but on the compatibility of the site types.

9 CONCLUSION

Prominent in DEVS is the clear distinction between system and environment by encapsulating the attributes and methods of a model and constraining the interaction between system and environment to typically static ports, even in DEVS variants that support variable structures. The knowledge of a system ends at its ports, it does not even know with whom it is interacting. Only the superior coupled model is privy to that information. From outside nothing is known about the internal state, if the model has not communicated it before. To hide this information causes an undesirable overhead in modeling and simulating certain phenomena of dynamic systems.

The combination of variable ports together with multi-couplings provides one solution to this problem. Via variable ports, internal state changes can be signalized to the outside world. Due to the use of multi-couplings, this signalizing is directly reflected at the interaction level, as multi-couplings are no longer defined extensionally but intensionally based on the existence of ports. Rooted in the formalism DYN-DEVS, the formalism ρ-DEVS has been defined, which is reflective in nature and takes dynamic interfaces into account. An abstract simulator has been developed. The Trp operon model has been used to illustrate the basic ideas of the introduced concepts and their value in providing additional structure for modeling and reducing artificial overhead in modeling and simulation likewise. With multi-couplings, we are heading into the direction of putting more emphasis on the interaction between models. With this, we reflect current trends as they appear in areas like process algebras as well. Requirements of application areas like cell biology will motivate further explorations into interaction patterns, e.g., to support a synchronous interaction.

REFERENCES

Barros, F. 1997. Modeling Formalism for Dynamic Structure Systems. *ACM Transactions on Modeling and Computer Simulation* 7 (4): 501–514.

Degenring, D., M. Röhl, and A. Uhrmacher. 2004. Discrete Event, Multi-Level Simulation of Metabolite Channeling. *BioSystems* 75 (1-3): 29–41.

Gillespie, D. T. 1976. A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *Journal of Computational Physics* 22:403–434.

Harel, D., and A. Naamad. 1996, October. The STATEMATE semantics of statecharts. *ACM Transactions on Software Engineering and Methodology* 5 (4): 293–333.

Himmelspach, J., and A. M. Uhrmacher. 2004, October. Processing dynamic PDEVS models. In *Proceedings of the 12th IEEE International Symposium on MASCOTS*, ed. D. DeGroot and P. Harrison, 329–336. Volendam, The Netherlands: IEEE Computer Society.

Himmelspach, J., and A. M. Uhrmacher. 2006. Sequential processing of pdevs models. In *Proceedings of the EMSS 06*.

Hu, X., B. Zeigler, and S. Mittal. 2005. Variable structure in devs component-based modeling and simulation. *Simulation* 81 (2).

Kuttler, C., and A. Uhrmacher. 2006. Multi-level modeling in systems biology by discrete event approaches. *IT Themenheft Systems Biology*.

Pawletta, T., B. Lampe, S. Pawletta, and W. Drewelow. 1996. A new approach for simulation of variable structure systems. In *Proc. 41th KoREMA*, Volume 2, 57–62. Opatija, Croatia.

Priami, C. 1995. Stochastic π-calculus. *Computer Journal* 6:578–589.

Priami, C., and P. Quaglia. 2005. Beta binders for biological interactions. *Transactions on Computationa Systems Biology*.

Uhrmacher, A. 2001. Dynamic Structures in Modeling and Simulation - A Reflective Approach. *ACM Transactions on Modeling and Simulation* 11 (2): 206–232.

Uhrmacher, A., and C. Priami. 2005, December 4-7. Discrete event systems specification in Systems Biology - a discussion of stochastic pi-calculus and DEVS. In *Proc. of the Winter Simulation Conference*.

Zeigler, B. 1986. Toward a simulation methodology for variable structure modelling. In *Modelling and Simulation Methodology in the Artificial Intelligence Era*, ed. M. Elzas, T. Ören, and B. Zeigler, 195–210. North-Holland,.

Zeigler, B., H. Praehofer, and T. Kim. 2000. *Theory of Modeling and Simulation*. London: Academic Press.

AUTHOR BIOGRAPHIES

ADELINDE M. UHRMACHER is an Associate Professor at the Department of Computer Science at the University of Rostock and head of the Modeling and Simulation Group. Her research interests are in modeling and simulation methodologies, particularly agent-oriented modeling and simulation and their applications. Her e-mail address is ⟨`lin@informatik.uni-rostock.de`⟩.

JAN HIMMELSPACH holds a MSc in Computer Science from the University of Koblenz. His research interests are on designing flexible and efficient simulation systems. He is currently a research scientist at the Modeling and Simulation Group at the University of Rostock. His e-mail address is ⟨`jh194@informatik.uni-rostock.de`⟩.

MATHIAS RÖHL holds a MSc in Computer Science from the University of Rostock. His research interests are on component-based modeling and agent-oriented simulation. He is currently a research scientist at the Modeling and Simulation Group at the University of Rostock. His e-mail address is ⟨`mroehl@informatik.uni-rostock.de`⟩.

ROLAND EWALD holds an MSc in Computer Science from the University of Rostock. His research interests are in parallel and distributed simulation. He is currently a research scientist at the Modeling and Simulation Group at the University of Rostock. His e-mail address is ⟨`re027@informatik.uni-rostock.de`⟩.