Coupling Permutation and Model Migration Based on Dynamic and Adaptive Coupling Mechanisms

Sunwoo Park[1] **and C. Anthony Hunt**[1,2]
[1]**BioSystems Group, Dept. of Biopharmaceutical Sciences, University of California, San Francisco;**
[2]**Joint Graduate Group in Bioengineering, University of California, Berkeley and San Francisco,**
513 Parnassus Ave., San Francisco, CA 94143-0446, USA
parks@pharmacy.ucsf.edu a.hunt@ucsf.edu

Keywords: DEVS, coupling permutation, dynamic structure changes, model migration

Abstract

We present new coupling mechanisms for dynamic and adaptive structure changes of hierarchical DEVS models that are needed for development of adaptive biological system models. Unlike previous research, the proposed mechanisms describe structure changes at the coupling specification level. It does not require any further atomic level information. With these mechanisms, sophisticated coupling permutation and model migration are described. Evolution and optimization of coupling relationships between components are also feasible. We provide formal specifications of the proposed coupling mechanisms and revised DEVS coupled models. Aided by examples, we also discuss major issues related to structure changes from a coupling perspective.

1. INTRODUCTION

Many physical and biological phenomena occur as a consequence of dynamic or adaptive structure changes. Structure changes often supervene the permutation of the structural relationship between components of a system or the way they interact. Computer network traffic regulation is mainly achieved by dynamically changing traffic flow paths between subsystems of a computer network system (e.g., routers and host computers) based on certain heuristics [1]. A network topology can continuously change by switching a set of flow paths between components to maintain network traffic under an optimal or healthy condition. Mammalian immunization mounts a response when malicious viruses invade the host. Immune cells (e.g., T-cell and B-cell) defend their host system by engulfing and metabolizing the viruses [2]. In a simulation of these phenomena, the former would be an example of dynamic coupling permutations and the latter would be an example of model migration.

π-calculus provides a formal framework for the description of dynamic structure changes related to dynamic coupling permutations [3]. It describes dynamic permutation of communication links between systems by a series of algebraic expressions. *Mobile ambient* extends *π-calculus* to support dynamic structure changes associated with model migration [4]. It introduces the new concept, the *ambient*, that defines the computational boundary within which the systems reside. Migration refers to as a process by which a system moves from one ambient to another. *BioAmbients calculus* improves *Mobile ambient* to cope with dynamic structure changes in biological compartment problems [5]. Since an ambient is a computational boundary, the option of one system migrating into another does not exist.

Dynamic structure changes of DEVS models have been investigated from both perspectives. Barros utilizes a special atomic model to describe dynamic structure changes [6][7]. Detection and execution of structure changes are described inside transition functions. His V-DEVS and DSDEVS leave the detail logics that govern structural changes to the modeler. Uhrmacher describes dynamic structure changes based mainly on a *model transition function* [8][9]. The function allows a model to transit to another model at the current state. Because the second model is able to explore different spatial, state, and behavioral spaces, the changes are realized by simply choosing a different model when a transition occurs. Clearly, the process requires a set of models to which a model could transit. For some problems, it could be difficult or even infeasible to clearly define that set of models. It seems likely that enormous models would be required to represent complex structural changes. Both of the above approaches deal with structural changes at an atomic level. Hu et. al. allow a coupled model to add, remove, and update its couplings and components based on the concept of an *operations*

boundary [10]. They also permit an atomic model to dynamically add and remove its ports. However, this approach does not cover structural changes associated with model migration. Nevertheless, it has practical advantages.

Below, we present a set of new formal specifications of dynamic and adaptive coupling mechanisms that describe the structural changes of a DEVS model at a coupled level. Both dynamic coupling permutation and model migration are described at the coupled level. The method does not require any further information at the atomic model level. We also present revised DEVS coupled models, which incorporate new coupling specifications, and a set of simple examples to facilitate understanding of the proposed coupling mechanisms and their specifications.

2. STATIC COUPLING SPECIFICATION

A coupling is an *influence relation* between objects (e.g., a unidirectional event propagation path between two systems). It consists of an influencer and an influencee in a dyadic relation, and a set of influencers and influencees in a n-ary relation. In this paper, we mainly focus on the dyadic relation and use the term *model* in lieu of *object*.

Definition 1 *A (dyadic) static coupling c is $(m_{src}, p_{src}, m_{tgt}, p_{tgt})$ where m_{src} is an influencer, m_{tgt} is an influencee, p_{src} is a property set of m_{src}, and p_{tgt} is a property set of m_{tgt}.*

Example 1 *Consider a DEVS coupled model in Figure 1. There exists five couplings in M; $Z_{M,1}, Z_{2,M}, Z_{1,2}, Z_{1,3}$, and $Z_{2,3}$. The component M_2 also has four couplings, $Z_{2,21}, Z_{2,22}, Z_{21,2}$, and $Z_{22,2}$. However, those couplings are valid and visible only within M_2.*

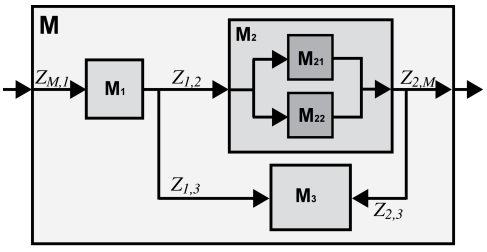

Figure 1: An example of DEVS static coupling

3. DYNAMIC COUPLING SPECIFICATION

A dynamic coupling is an advanced coupling mechanism that enables dynamic permutation of a coupling including its creation, destruction, and updating. It extends a static coupling by appending a new influencer or influencee (m_{new}, p_{new}), a constraint ξ, a permutation function ρ, and a lifespan variable l. A dynamic coupling permutation occurs when ξ is satisfied and $l > 0$ by executing ρ.

Definition 2 *A dynamic coupling $c \in C$ is $(m_{src}, p_{src}, m_{tgt}, p_{tgt}, m_{new}, p_{new}, \xi, \rho, l)$ where m_{src}, $m_{tgt}, m_{new} \in M$, $p_{src} \in m_{src}$, $p_{tgt} \in m_{tgt}$, $p_{new} \in m_{new}$, ξ is coupling constraint, $\rho : C \rightarrow C$ is a coupling permutation function, and $l \in \mathbb{R}^+$ is a life-span variable. M is a set of models and C is a set of dynamic couplings.*

M refers to a (*coupling*) *environment set* because m_{src}, m_{tgt}, and m_{new} of $c \in C$ are defined over the set. M provides the boundary at which C is effective and visible. M also refers to a hierarchical (or multi-scale) environment set if it contains at least one sub-environment. It is legitimate to use the sub-environment as either an influencer or an influencee.

ξ is a set of conditions or requirements associated with the occurrence of a coupling permutation. It can be specified in various ways including *IO snooping* or *model reflection*. IO snooping enables detecting a coupling permutation based on information that m_{src} sends or m_{tgt} receives (e.g., event types or the content of an event). Model reflection allows utilizing the model's internal information for coupling permutation detection (e.g., state variables).

Example 2 *For a DEVS model $m \in M$, (i) consider $\xi = \{m.p_{out} : r\}$ where, p_{out} is the output port of m, and r is a certain value. ξ implies that a coupling permutation will occur when m_{src} produces an event that contains the value (or an event type) r through p_{out}. and (ii) consider $\xi = \{m.sv : s_2\}$ where, sv is a state variable of m and s_2 is one of discrete states that sv can explore. ξ implies that a coupling permutation will occur when sv reaches s_2.*

ρ is a function that permutates a coupling. By replacing (m_{src}, p_{src}) or (m_{tgt}, p_{tgt}) by (m_{new}, p_{new}), it changes influence relationships between models. Because ρ is part of the coupling, it could be revised or changed to another permutation function ρ'. It controls the lifetime of the coupling by changing the value of l. l can be specified in various ways. However, it is used mainly as a time variable or a counter. The former specifies the

amount of time the coupling lives and the latter tracks the total number of times that dynamic permutations occur. The coupling is destroyed when $l \leq 0$.

Hereafter, we also use ev_s, ev_t, and ev_n for $(m_{src}, p_{src}), (m_{tgt}, p_{tgt})$, and (m_{new}, p_{new}) to present more concise coupling specifications.

Example 3 *Consider a set of coupling permutation functions with a lifespan counter l. ρ_{switch} changes the influencee to ev_n. ρ_{swap} exchanges ev_t with ev_n. ρ_{spawn} creates a new coupling c', which has ev_n as an inflencee, from an existing coupling c. All ev_n in ρ_{switch} and ρ_{spawn} are identical. ρ_{remove} destroys a coupling regardless of ev_t and ev_n. $\rho_{apoptosis}$ destroys a coupling by assigning the new influencee to ϕ. $\rho_{migrate}$ destroys a coupling after m_{src} migrates into m_{tgt}. We assume that $\rho_{migrate}$ abstracts model migration mechanism, which is $m_{src} \notin m_{new} \rightarrow m_{src} \in m_{new}$. It is legitimate because m_{src}, m_{new}, and m_{tgt} are elements of a dynamic coupling and $\rho_{migrate}$ is a mapping between two couplings.*

- $\rho_{switch} : (ev_s, ev_t, ev_n, \xi, \rho, l) \rightarrow (ev_s, ev_n, ev_n, \xi,\ \rho, l\text{-}1)$
- $\rho_{swap} : (ev_s, ev_t, ev_n, \xi, \rho, l) \rightarrow (ev_s, ev_n, ev_t, \xi, \rho, l\text{-}1)$
- $\rho_{spawn} : (ev_s, ev_t, ev_n, \xi, \rho, l) \rightarrow (ev_s, ev_t, ev_n, \xi, \rho, l\text{-}1) \mid (ev_s, ev_n, ev_n, \xi, \rho, l)$
- $\rho_{remove} : (ev_s, ev_t, ev_n, \xi, \rho, l) \rightarrow \phi$
- $\rho_{apoptosis} : (ev_s, ev_t, \phi, \xi, \rho, l) \rightarrow \phi$
- $\rho_{migrate} : (ev_s, ev_t, ev_n, \xi, \rho, l) \rightarrow \phi$

In lieu of ev_n, we can use a set of influencees $\{ev_{n_1}, \ldots, ev_{n_N}\}$ to describe more advanced dynamic coupling permutations. A coupling permutation function, ρ_{split}, which decomposes a coupling to a set of couplings is defined by $(ev_s, ev_t, ev_{n_1}, \ldots, ev_{n_N}, \xi, \rho, l) \rightarrow (ev_s, ev_{n_1}, ev_{n_1}, \xi, \rho, l\text{-}1) \mid \ldots \mid (ev_s, ev_{n_N}, ev_{n_N}, \xi, \rho, l\text{-}1)$.

Couplings are categorized into five groups. A dynamic coupling is (*i*) *persistent* if l is ∞, (*ii*) *non-permutable* if ev_t is equal to ev_n, (*iii*) *permanent* if both (*i*) and (*ii*), (*iv*) *static* if ξ is ϕ, ρ is ϕ, and l is ∞, and (*v*) *dynamic*, otherwise. If there exists no confusion, a dynamic coupling $(ev_s, ev_t, ev_n, \xi, \rho, l)$ is simplified to $(ev_s, ev_t, ev_n, \xi, \rho)$, $(ev_s, ev_t, \xi, \rho, l)$, (ev_s, ev_t, ξ, ρ), or (ev_s, ev_t) depending on the case above.

4. ADAPTIVE COUPLING SPECIFICATION

An adaptive coupling is an extended dynamic coupling such that either ev_s, ev_t, or ev_n is initially unspecified but is resolved by a binding function φ when the binding constraint ψ is satisfied. An unspecified event handler is denoted by a wildcard character $'*'$. The coupling produces a dynamic coupling if φ successfully maps $(*, *)$ to the pair of an influencer or an influencee in M and its property set (m_r, p_r).

4.1. Proactive Coupling Specification

A *proactive coupling* is an adaptive coupling such that ev_s or ev_t is unspecified and φ resolves it when M is updated after any dynamic or adaptive activity. Model migration is such an activity. From the coupling perspective, model migration is an activity by which a model traverses from one coupling environment to another environment based on certain heuristics.

Definition 3 *A proactive coupling $c \in C$ is (i) $(*, *, m_{tgt}, p_{tgt}, m_{new}, p_{new}, \xi, \rho, l, \psi, \varphi)$ when (m_{src}, p_{src}) is unspecified, and (ii) $(m_{src}, p_{src}, *, *, m_{new}, p_{new}, \xi, \rho, l, \psi, \varphi)$ when (m_{tgt}, p_{tgt}) is unspecified. In addition to Definition 2, ψ is a binding constraint. $\varphi : (M \cup \{*\}) \times (P \cup \{*\}) \rightarrow (M \cup \{*\}) \times (P \cup \{*\})$ is a binding function that maps $(*, *)$ to (m_{src}, p_{src}) or (m_{tgt}, p_{tgt}). P is a set of property sets. $*$ represents unspecified information.*

Example 4 *Consider a proactive coupling $c = (*, *, m_2, p_2, m_3, p_3, \xi, \rho, l, \psi, \varphi)$, two coupling environment sets $M = \{m_1, m_2, m_3\}$, $N = \{m_4, m_5\}$, and ψ is $m_n \neq \phi$. Let m_n and p_n be the model that migrates to M and its property set, respectively, and both are initially ϕ. φ resolves the unspecified influencer and its property set to any of $(m_n, p_n), (m_1, p_1)$, or $(*, *)$.*

$$\varphi = ev_r = (m_r, p_r) = \begin{cases} (m_n, p_n), & m_n = Type_A \\ (m_1, p_1), & m_n = Type_B \\ (*, *), & otherwise. \end{cases}$$

In proactive coupling specification, ev_s or ev_t of ρ can also be represented by $'*'$. When φ resolves unspecified ev_s or ev_t, $'*'$ in ρ is also identified. proactive versions of the permutation functions in Example 3 are listed as follows.

- $\rho_{switch} : (*, ev_t, ev_n, \xi, \rho, l) \xrightarrow{\varphi} (ev_r, ev_n, ev_n, \xi, \rho, l\text{-}1)$
- $\rho_{swap} : (*, ev_t, ev_n, \xi, \rho, l) \xrightarrow{\varphi} (ev_r, ev_n, ev_t, \xi, \rho, l\text{-}1)$
- $\rho_{spawn} : (*, ev_t, ev_n, \xi, \rho, l) \xrightarrow{\varphi} (ev_r, ev_t, ev_n, \xi, \rho, l\text{-}1) \mid (ev_r, ev_n, ev_n, \xi, \rho, l)$
- $\rho_{remove} : (*, ev_t, \phi, \xi, \rho, l) \xrightarrow{\varphi} \phi$
- $\rho_{apoptosis} : (*, ev_t, ev_n, \xi, \rho, l) \xrightarrow{\varphi} \phi$

 ISBN 1-56555-304-7

- ρ_{split} : $(*, ev_t, ev_{n_1}, \ldots, ev_{n_N}, \xi, \rho, l) \xrightarrow{\varphi} (ev_r, ev_{n_1}, ev_{n_1}, \xi, \rho, l\text{-}1) | \ldots | (ev_r, ev_{n_N}, ev_{n_N}, \xi, \rho, l\text{-}1)$

4.2. Reactive Coupling Specification

A reactive coupling is an adaptive coupling when ev_n is unspecified. ev_n is resolved by φ when ψ is satisfied.

Definition 4 *A reactive coupling $c \in C$ is $(m_{src}, p_{src}, m_{tgt}, p_{tgt}, *, *, \xi, \rho, l, \psi, \varphi)$. In addition to Definition 2, ψ is a binding constraint. $\varphi : (M \cup \{*\}) \times (P \cup \{*\}) \rightarrow (M \cup \{*\}) \times (P \cup \{*\})$ is a binding function that maps $(*, *)$ to (m_{new}, p_{new}). P is a set of property sets. $*$ represents unspecified information.*

In the reactive coupling specification, ev_n of ρ can also be represented by $'*'$. When φ resolves unspecified ev_n, $'*'$ in ρ is also identified. Reactive versions of the permutation functions in Example 3 are listed as follows.

- ρ_{switch} : $(ev_s, ev_t, *, \xi, \rho, l) \xrightarrow{\varphi} (ev_s, ev_r, ev_r, \xi, \rho, l\text{-}1)$
- $\rho_{swap} : (ev_s, ev_t, *, \xi, \rho, l) \xrightarrow{\varphi} (ev_s, ev_r, ev_t, \xi, \rho, l\text{-}1)$
- $\rho_{spawn} : (ev_s, ev_t, *, \xi, \rho, l) \xrightarrow{\varphi} (ev_s, ev_t, ev_r, \xi, \rho, l\text{-}1) \mid (ev_s, ev_r, ev_r, \xi, \rho, l)$
- $\rho_{remove} : (ev_s, ev_t, \phi, \xi, \rho, l) \xrightarrow{\varphi} \phi$
- $\rho_{apoptosis} : (ev_s, ev_t, *, \xi, \rho, l) \xrightarrow{\varphi} \phi$
- ρ_{split} : $(ev_s, ev_t, ev_{*_1}, \ldots, ev_{*_N}, \xi, \rho, l) \xrightarrow{\varphi} (ev_s, ev_{r_1}, ev_{r_1}, \xi, \rho, l\text{-}1) | \ldots | (ev_r, ev_{r_N}, ev_{r_N}, \xi, \rho, l\text{-}1)$

5. COUPLING EXECUTION

Definition 5 *Coupling execution $\Re(c, M)$ is an abstract function that executes the coupling $c \in \{C_s, C_d, C_a\}$ over the coupling environment set M. C_s is a set of static couplings. C_d is a set of dynamic couplings. C_a is a set of adaptive couplings. $C = C_s \cup C_d \cup C_a$.*

$\Re(c \in C_s, M) : C_s \rightarrow C_s$ returns c. $\Re(c \in C_d, M) : C_d \rightarrow \times_{k=1 \ldots N} C_d$ returns $\{c_k\}_{k=1}^{N}$ produced by ρ if ξ is satisfied and $l > 0$, $\{\phi\}$ if ξ is satisfied but $l \leq 0$, or c if ξ is not satisfied. $\Re(c \in C_a, M) : C_a \rightarrow C_a \times C_d$ returns $\{c, c_d \in C_d\}$ if ψ is satisfied and φ maps $(*, *)$ to $(m_r \in M, p_r \in P)$ or $\{c, \phi\}$ if ψ is not satisfied. Also, $\Re(c \in C_d, M)$ and $\Re(c \in C_a, M)$ updates C_d to $C_d \cup \{c_k\}_{k=1}^{N}$ and $C_d \cup c_d$, respectively. C_s and C_a are intact.

Example 5 *Consider the proactive coupling c in Example 4. The coupling execution $\Re(c, M) = \Re(c, \{m_1, m_2, m_3\})$. When $m_4 \in N$ migrates into M, N reduces to $\{m_5\}$ and M grows to $\{m_1, m_2, m_3, m_4\}$. Since m_4 is m_n and ψ is satisfied, φ resolves $(*, *)$ to (m_4, p_4) if m_4 is $Type_A$ or to (m_1, p_1) if m_4 is $Type_B$. In the formal case, $\Re(c, M)$ produces $\{c, (m_4, p_4, m_2, p_2, m_3, p_3, \xi, \rho, l)\}$. It shows how an existing model establishes a new coupling with the incoming model. In the latter case, $\Re(c, M)$ produces $\{c, (m_1, p_1, m_2, p_2, m_3, p_3, \xi, \rho, l)\}$. It shows how existing models interact with each other when m_4 migrates into M. In case the migration does not occur or φ resolves $(*, *)$ to $(*, *)$, $\Re(c, M)$ produces $\{c, \phi\}$.*

Definition 6 *Coupling execution $\Re(C, M)$ is a set of $\Re(c, M)$ for all $c \in C$ over the coupling environment set M. That is, $\Re(C, M) = \Re(\{c_1, c_2, \ldots, c_N\}, M) = \{(\Re(c_1, M), \Re(c_2, M), \ldots, \Re(c_N, M)\}$.*

$\{(\Re(c_1, M), \Re(c_2, M), \ldots, \Re(c_N, M)\}$ is equivalent to $\Re(c_1, M) | \Re(c_2, M) | \ldots | \Re(c_N, M)$. If $\Re(C, M)$ is independent from the execution order of $\Re(c, M)$ for each $c \in C$, $\Re(c_1, M) | \Re(c_2, M) | \ldots | \Re(c_N, M)$ is also equivalent to $\Re(c_1, M) \Re(c_2, M) \ldots \Re(c_N, M)$. We can consider $C = \{c_1, c_2, \ldots, c_k\}$ as a sequence of couplings $w = c_1 c_2 \ldots c_k$.

M becomes non-permutable if it is not changed after $\Re(c, M)$ or $\Re(C, M)$. In that case, $\Re(c, M)$ and $\Re(C, M)$ are simplified to $\Re(c)$ and $\Re(C)$, respectively.

Example 6 *Consider $C = \{c_1, c_2, c_3\}$ in which each coupling is independent from the others. The coupling execution of C over a non-permutable set M is $\Re(C, M) \equiv \Re(C) \equiv \Re(\{c_1, c_2, c_3\}) \equiv \{\Re(c_1), \Re(c_2), \Re(c_3)\} \equiv \Re(c_1) | \Re(c_2) | \Re(c_3) \equiv \Re(c_1) \Re(c_2) \Re(c_3)$.*

6. DEVS MODELS WITH COUPLING SPECIFICATIONS

To demonstrate the flexibility and integrability of our coupling mechanisms, we present a collection of DEVS coupled model specifications. We use P-DEVS specification as an existing DEVS formalism [11][12][13]. It is not required to revise an atomic model specification because coupling information appears only in a coupled model specification. Revised coupled model specifications are listed as follow.

Definition 7 *A coupled model with dynamic coupling N is $\langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{d,i}\} \rangle$. X is a set of input events. Y is a set of output events. D is a set of components. For each d in D, M_d is a parallel DEVS model in [11]. For each $d \in D \cup \{N\}$, I_d is the influencee set of d. For each $i \in I_d$, $Z_{d,i}$ is a function, d-to-i output translation with (i) $Z_{d,i} : X \xrightarrow{X_k, \xi, \rho, l} X_i$, if $d = N$, (ii) $Z_{d,i} : Y_d \xrightarrow{Y_k, \xi, \rho, l} Y$, if $i = N$, and (iii) $Z_{d,i} : Y_d \xrightarrow{X_k, \xi, \rho, l} X_i$, if $d \neq N \wedge i \neq N$, where $k \in D \cup \{N\}$. $X \xrightarrow{X_k, \xi, \rho, l}$*

X_i *is* $(N, X, M_i, X_i, M_k, X_k, \xi, \rho, l)$. $Y_d \xrightarrow{Y_k,\xi,\rho,l} Y$ *is* $(M_d, Y_d, N, Y, M_k, Y_k, \xi, \rho, l)$. $Y_d \xrightarrow{X_k,\xi,\rho,l} X_i$ *is* $(M_d, Y_d, M_i, X_i, M_k, X_k, \xi, \rho, l)$.

Definition 8 *A coupled model with proactive coupling* N *is* $\langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{d,i}\}\rangle$, *where* X, Y, D, M_d, *and* k *are as in Definition 7, and* $Z_{d,i}$ *is a function, d-to-i output translation with (i)* $Z_{d,i} : X \xrightarrow{X_k,\xi,\rho,l,\psi,\varphi} X_*$, *if* $d = N$, *(ii)* $Z_{d,i} : Y_* \xrightarrow{Y_k,\xi,\rho,l,\psi,\varphi} Y$, *if* $i = N$, *and (iii)* $Z_{d,i} : Y_* \xrightarrow{X_k,\xi,\rho,l,\psi,\varphi} X_i$ *or* $Y_d \xrightarrow{X_k,\xi,\rho,l,\psi,\varphi} X_*$, *if* $d \neq N \wedge i \neq N$, *where* $* \in D \cup \{N\}$ *is an unspecified component.* $X \xrightarrow{X_k,\xi,\rho,l,\psi,\varphi} X_*$ *is* $(N, X, M_*, X_*, M_k, X_k, \xi, \rho, l, \psi, \varphi)$. $Y_* \xrightarrow{Y_k,\xi,\rho,l,\psi,\varphi} Y$ *is* $(M_*, Y_*, N, Y, M_k, Y_k, \xi, \rho, l, \psi, \varphi)$. $Y_* \xrightarrow{X_k,\xi,\rho,l,\psi,\varphi} X_i$ *is* $(M_*, Y_*, M_i, X_i, M_k, X_k, \xi, \rho, l, \psi, \varphi)$. $Y_d \xrightarrow{X_k,\xi,\rho,l,\psi,\varphi} X_*$ *is* $(M_d, Y_d, M_*, X_*, X_k, \xi, \rho, l, \psi, \varphi)$.

Definition 9 *A coupled model with reactive coupling* N *is* $\langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{d,i}\}\rangle$, *where* X, Y, D, M_d, *and* $*$ *are as in Definition 7 and 8, and* $Z_{d,i}$ *is a function, d-to-i output translation with (i)* $Z_{d,i} : X \xrightarrow{X_*,\xi,\rho,l,\psi,\varphi} X_i$, *if* $d = N$, *(ii)* $Z_{d,i} : Y_d \xrightarrow{Y_*,\xi,\rho,l,\psi,\varphi} Y$, *if* $i = N$, *and (iii)* $Z_{d,i} : Y_d \xrightarrow{X_*,\xi,\rho,l,\psi,\varphi} X_i$, *if* $d \neq N \wedge i \neq N$. $X \xrightarrow{X_*,\xi,\rho,l,\psi,\varphi} X_i$ *is* $(N, X, M_i, X_i, M_*, X_*, \xi, \rho, l, \psi, \varphi)$. $Y_d \xrightarrow{Y_*,\xi,\rho,l,\psi,\varphi} Y$ *is* $(M_d, Y_d, N, Y, M_*, Y_*, \xi, \rho, l, \psi, \varphi)$. $Y_d \xrightarrow{X_*,\xi,\rho,l,\psi,\varphi} X_i$ *is* $(M_d, Y_d, M_i, X_i, M_*, X_*, \xi, \rho, l, \psi, \varphi)$.

Definition 10 *A coupled model with universal coupling* N *is* $\langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{d,i}\}\rangle$, *where* X, Y, D, M_d, *and* $*$ *are as in Definition 7 and 8, and* $Z_{d,i}$ *is a function, d-to-i output translation with* $Z_{d,i} \in (Z^s_{d,i} \cup Z^d_{d,i} \cup Z^p_{d,i} \cup Z^r_{d,i})$. $Z^s_{d,i}$ *is equivalent to the static translation function* $Z_{d,i}$ *in [11].* $Z^d_{d,i}$ *is* $Z_{d,i}$ *in Definition 7.* $Z^p_{d,i}$ *is* $Z_{d,i}$ *in Definition 8.* $Z^r_{d,i}$ *is* $Z_{d,i}$ *in Definition 9.*

7. DEVS EXAMPLE: ADAPTIVE EVENT FLOW MANAGEMENT

Consider a coupled model N having a queue M_1 and two processors M_2, and M_3. M_1 accepts and allocates jobs to M_2 or M_3. A processor serves them with its resources and reports its current resource availability level to M_1. The level is represented by a percentage scale. When M_1 detects that the level of the processor, which is connected to M_1, is lower than 20%, it switches to an alternative processor. However, if both processors are lower than 20%, M_1 does not accept any new job from N by removing the coupling between N and M_1. When the resource level of the current processor becomes more than 20%, it accepts new jobs again by recreating the coupling. We assume M_1 is initially connected to M_2 and resource availability levels of both processors are 100%.

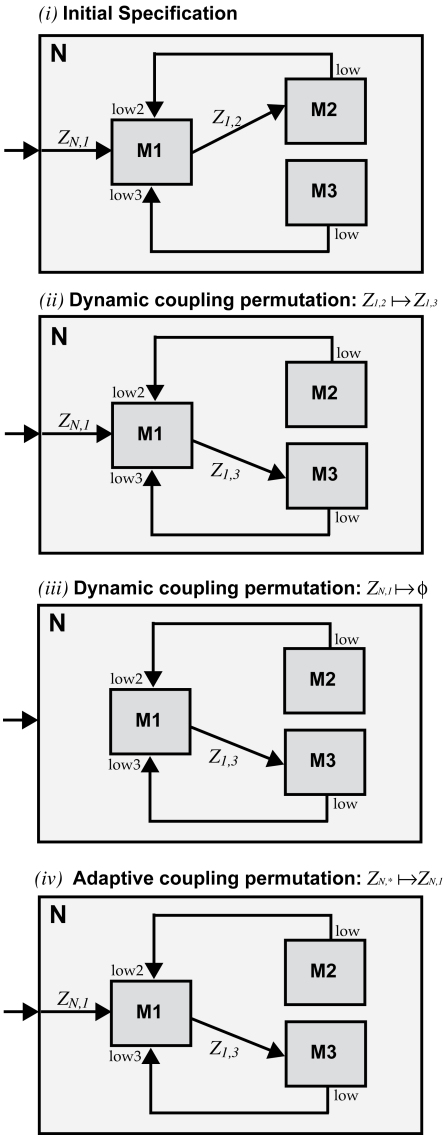

Figure 2: Adaptive event flow management model

7.1. Model Specification

To realize the scenario, we define a coupled model based on the specification in Definition 10. The adaptive event flow management model N is $\langle X, Y, D, M, I, Z\rangle$ with $X = \{(in, job_1), (in, job_1), \ldots, (in, job_k))\}$, $Y = \phi$, $D = \{1, 2, 3\}$, $M = \{M_1, M_2, M_3\}$, $I = \{I_N, I_1, I_2, I_3\}$, and $Z = \{Z_{N,1}, Z_{1,2}, Z_{2,1}, Z_{3,1}, Z_{N,*}\}$.

$I_N = \{1\}$, $I_1 = \{2\}$, $I_2 = \{1\}$, $I_3 = \{1\}$. $Z_{N,1} = (N, in, M_1, in, \phi, \phi, M_1.full{=}true, \rho_{remove}, \infty)$. $Z_{1,2} = (M_1, out, M_2, in, M_3, in, M_1.low_2{=}true \veebar M_1.low_3{=}true, \rho_{swap}, \infty)$. $Z_{N,*} = (N, in, *, *, \phi, \phi, *.full{=}true, \rho_{set}, \infty, \psi, \varphi)\}$. $Z_{2,1} = (M_2, low, M_1, low_1)$. $Z_{3,1} = (M_3, low, M_1, low_2)$. $\psi = (M_1.full \neq true \wedge Z_{N,1} \notin Z)$. φ is (M_1, in) if $M_1.full \neq true$. Otherwise, it is $(*, *)$. $\veebar$ is a binary XOR operator, which returns *true* if only one operand is *true*. In this specification, there exists two static couplings ($Z_{2,1}$ and $Z_{3,1}$), two dynamic couplings ($Z_{N,1}$ and $Z_{1,2}$), and one adaptive coupling ($Z_{N,*}$).

We assume $M_2.low$ and $M_3.low$ are *true* when their resource level is lower than 20%, $M_2.low$ and $M_3.low$ are propagated to $M_1.low_2$ and $M_1.low_2$, respectively, and $M_1.full$ is $M_1.low_2 \wedge M_1.low_3$.

7.2. Model Execution

There exist five main distinct execution phases in this example. *The first phase* is initial execution before any dynamic coupling permutation occurs (step i in Figure 2). N runs the same as does a normal P-DEVS coupled model. *The second phase* is the first occurrence of a dynamic coupling permutation triggered by the low resource level of M_2 (step ii in Figure 2). As a consequence, $Z_{1,2}$ is changed to $Z_{1,3}$ with $(M_1, out, M_3, in, M_2, in, M_1.low_2{=}true \veebar M_1.low_3{=}true, \rho_{swap}, \infty)$. N is revised with $Z = \{Z_{N,1}, Z_{1,3}, Z_{2,1}, Z_{3,1}, Z_{N,*}\}$ and $I_1 = \{3\}$. *The third phase* is to switch back to M_2 when M_3 is lower than 20% and M_2 is higher than 20%. $Z_{1,3}$ is changed back to $Z_{1,2}$ with $(M_1, out, M_2, in, M_3, in, M_1.low_2{=}true \veebar M_1.low_3{=}true, \rho_{swap}, \infty)$. N is revised back to the original specification with $Z = \{Z_{N,1}, Z_{1,2}, Z_{2,1}, Z_{3,1}, Z_{N,*}\}$ and $I_1 = \{2\}$. The second and third phases will be repeated unless both M_2 and M_3 are lower than 20%. *The fourth phase* is the removal of $Z_{N,1}$ when M_1 detects both processors are $full$ (step iii in Figure 2). By removing the coupling, M_1 no longer accepts jobs from N. In the case where M_1 is connected to M_3, N is revised with $Z = \{Z_{1,3}, Z_{2,1}, Z_{3,1}, Z_{N,*}\}$ and $I_N = \phi$. *The fifth phase* is reconstruction of $Z_{N,1}$ when the processor recovers its resource above 20% after serving its jobs (step iv in Figure 2). The adaptive coupling $Z_{N,*}$ recreates $Z_{N,1}$. In case M_1 is connected to M_3, N is revised with $Z = \{Z_{N,1}, Z_{1,3}, Z_{2,1}, Z_{3,1}, Z_{N,*}\}$ and $I_N = \{1\}$.

The preceding example shows how dynamic and adaptive couplings work at a DEVS M&S level. During execution, the original DEVS specification evolves. In many modeling problems, under the current DEVS paradigm, it would be a challenge to find the optimal coupling relationship between components. With the proposed coupling mechanisms and specifications, we can handle such tasks more clearly and efficiently. They enable DEVS coupled models to evolve and optimize their couplings from their initial specifications.

7.3. Model Migration

Now, we consider the issues of model migration. Assume N accepts a processor $M_4 \notin N$ through its input port $assist$. After M_4 migrates to N, it creates a coupling with M_1 and serves as the third processor to reduce the computational load of N.

To support this scenario, consider a new dynamic coupling $Z_{4,N} : (M_4, out, N, assist,\ N, assist, \xi, \rho_{migrate}, l)$ with the revised N containing an additional input port $assist$ (step v in Figure 3). When ξ is satisfied, $\rho_{migrate}$ performs model migration and destroys the coupling between M_4 and N (step vi in Figure 3). It is legitimate because M_4 and N are elements of $Z_{4,N}$ and $\rho_{migrate}$ can change them (i.e., $M_4 \notin N \rightarrow M_4 \in N$). See Example 3 for more information on $\rho_{migrate}$. $D \in N$ is updated to $\{1, 2, 3, 4\}$ after $\rho_{migrate}$ completes. To describe adaptive coupling construction between M_4 and M_1, also consider a new proactive coupling $Z_{*,1} : (*, *, M_1, misc, \phi, \phi, \phi, \phi, \psi, \varphi)$ where ψ is $(Z_{k,1}{=}\phi)$ and φ resolves $(*, *)$ to (k, low). We assume M_1 has an additional input port $misc$ for this example. $Z_{*,1}$ creates a static coupling $(M_k, low, M_1, misc)$ after M_k migrates into N (step vii in Figure 3). We also redefine the permutation function ρ_{swap} by $\mathcal{R}(\rho_{swap}(k, i), \rho_{swap}(k, j))$ where $\mathcal{R} : (\rho, \rho) \rightarrow \rho$ is a random select function, and $\rho_{swap}(k, i)$ and $\rho_{swap}(k, j)$ are binary permutation functions for $i, j, k \in D \backslash 1$. After migration completes, $Z_{*,1}$ creates $Z_{4,1}$. $Z_{1,4}$ is created when both M_2 and M_3 have resources less than 20%. N is revised with $D = \{1, 2, 3, 4\}$, $M = \{M_1, M_2, M_3, M_4\}$, $Z = \{Z_{N,1}, Z_{1,3}, Z_{2,1}, Z_{3,1}, Z_{4,1}, Z_{N,*}, Z_{*,1}\}$, $I = \{I_N, I_1, I_2, I_3\}$ and $I_4 = \{1\}$ if M_4 is connected to M_1 but M_1 is connected to M_4.

8. CONCLUSION

We have described a set of new coupling mechanisms that enable building dynamic and adaptive structural changes at the coupling specification level. Among those changes, dynamic coupling permutations and adaptive model migrations are discussed from the perspectives of both pure coupling and DEVS M&S. These advances are needed for development of adaptive biological models. Unlike previous research, the proposed mechanisms do not need an atomic level description, which is particularly important when modeling in a biomedical context. They sustain a high degree of flexibility and to be applicable to virtually every DEVS formalism. It also enables performing scenario-based DEVS M&S. In addition to the formal specifications of the coupling mechanisms, we have also presented abstract coupling execution machines, a set of revised DEVS specifications, and

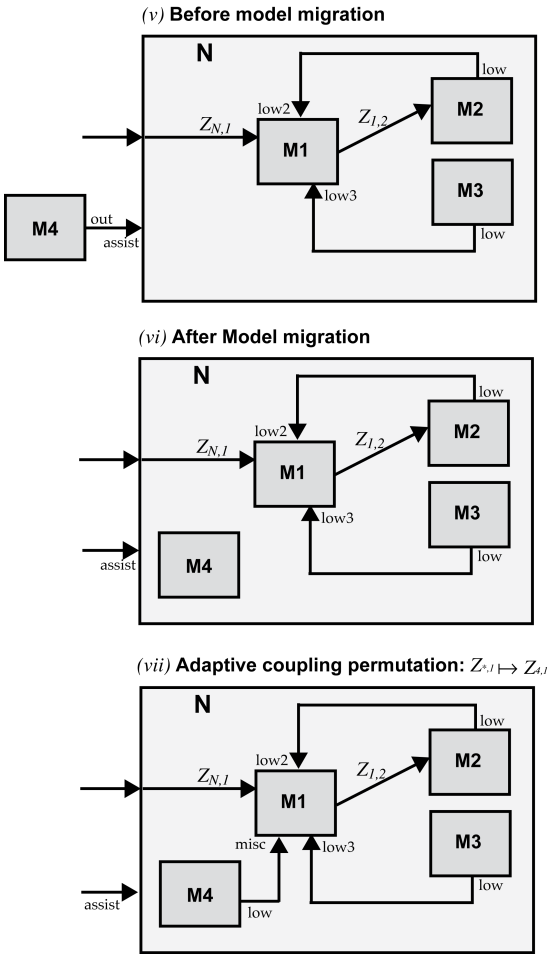

Figure 3: Adaptive event flow management model with model migration

examples.

ACKNOWLEDGEMENTS

This research was funded in part by CDH Research Foundation R21-CDH-00101 and CAH[1]. We are grateful for the Computational and Systems Biology Postdoctoral Fellowship funding provided to Sunwoo Park by the CDH Foundation. We thank Prof. Bernard P. Zeigler, Sean HJ Kim, and other members of the BioSystems Group for helpful discussion and commentary.

REFERENCES

[1] Halabi, S. 2000. *Internet Routing Architectures*, 2nd ed. Cisco Press, 2000.

[2] Roitt, I.; Brostoff, J.; and Male, D. 1998. *Immunology*. Mosby, 1998.

[3] Miller, R. 1999. *Communicating and mobile systems: the π-calculus*. Cambridge University Press, 1999.

[4] Cardelli, L. and Gordon, A. D. 2000. "Mobile ambients." *Theoretical Computer Science*. 240. 177–213.

[5] Regev, A.; Panina, E. M.; Silverman, W.; Cardelli, L.; and Shapiro, E. 2004. "Bioambients: an abstract for biological compartments." *Theoretical Computer Science*. 325. No. 1. 141–167.

[6] Barros, F. J.; Medes, M. T.; and Zeigler, B. P. 1994. "Variable DEVS - variable structure modeling formalism: An adaptive computer architecture application." in *AI, Simulation, and Planning in High Autonomy Systems, Proceedings of the Fifth Annual Conference on Distributed Interactive Simulation Environments*. 185–191.

[7] Barros, F. J. 1996. "The dynamic structure discrete event system specification formalism." *Transactions of the Society for Computer Simualation International*. 13. No. 1. 35–46.

[8] Uhrmacher, A. M. 2001. "Dynamic structures in modeling and simulation: A reflective approach." *ACM Transactions on Modeling and Computer Simulation (TOMACS)*. 11. No. 2. 206–232.

[9] ——. 2001. "A system theoretic approach to constructing test beds for multi-agent systems." *Discrete event modeling and simulation technologies: a tapestry of systems and AI-based theories and methodologies*. 315–339.

[10] Hu, X.; Zeigler, B. P.; and Mittal, S. 2005. "Variable structure in DEVS component-based modeling and simulation." *Simulation*. 81. No. 2. 91–102.

[11] Chow, A. C. and Zeigler, B. P. 1994. "Parallel devs: A parallel, hierarchical, modular modeling formalism." in *Proceedings of the 1994 Winter Simulation Conference*. Tew, J. D.; Manivannan, S.; Sadowski, D. A.; and Seila, A. F., Eds.

[12] Chow, A. C.; Zeigler, B. P.; and Kim, D. H. 1994. "Abstract simulator for the parallel devs formalism." in *Proceedings of the Fifth Annual Conference on I, Simulation, and Planning in High Autonomy Systems*. 157–163.

[13] Zeigler, B. P.; Praehofer, H.; and Kim, T. G. 2000. *Theory of Modeling and Simulation*, 2nd ed. Academic Press, 2000.

[1]CAH is a trustee of the CDH Research Foundation.