계층의 구조를 갖는 시뮬레이션 모델에 있어서 단계적 접근을 위한 모델연결' 방법론과 그 적용 예

Model Coupling Technique for Level Access in Hierarchical Simulation Models and Its Applications

조대호*, 이칠기*
Tae Ho Cho, Chil Gee Lee

Abstract

Modeling of systems for intensive knowledge-based processing requires a modeling methodology that makes efficient access to the information in huge data base models. The proposed *level access mothodology* is a modeling approach applicable to systems where data is stored in a hierarchical and modular modules of active memory cells (processor/memory pairs). It significantly reduces the effort required to create discrete event simulation models constructed in hierarchical, modular fashion for above application. Level access mothodology achieves parallel access to models within the modular, hierarchical modules (clusters) by broadcasting the desired operations (e.g. querying information, storing data and so on) to all the cells below a certain desired hierarchical level. Level access methodology exploits the capabilities of object-oriented programming to provide a flexible communication paradigm that combines port-to-port coupling with name-directed messaging. Several examples are given to illustrate the utility of the methodology.

1. Introduction

The need to explore clusters, and compounds of clusters for massively parallel computers is emphasized in [9], since for a massively parallel computer a true complete connection (each processing element in an N-node network must have N-1 interfaces to other processing elements) is too expensive, and even impossible for large N. The DEVS-Scheme modeling and simulation environment [10, 13, 11] has been employed to study such hierarchical computer architectures [7]. In DEVS-Scheme, the most basic models called atomic models are constructed and tested independently. The System

* 성균관 대학교 전기전자 및 컴퓨터공학부 조교수

Entity Structure (SES) couples them together, in hierarchical fashion, by linking I/O ports for building more complex models called *coupled models*. The coupling information for a coupled model is stored in the corresponding coupling table of the model. Translation methods provide I/O communication links among the models based on the coupling information in the coupling table.

Such I/O port coupling provides a loosely coupled and modular form of model interconnection [10]. Here all the I/O messages passed among the models are mediated by predefined I/O port couplings. So a sender model (model that produces the output for current interaction) just needs to know the proper output port name in order to send an output message. However, flexibility in interconnection is limited in this approach since it requires that the modeller explicitly change the I/O port coupling in order to establish new communication links.

In contrast, other object oriented simulation languages typically sends output messages by specifying the name of the model (object) and corresponding method for receiving the message [11, 1, 5]. The model interaction done this way does not use I/O port coupling and results in a tightly coupled form of model interaction which does not support full modular construction of models. However, structural linkage, performed at model definition time, is not necessary.

A new coupling scheme called *name-directed coupling* combines these two approaches. Within name-directed coupling, model interactions use both the modular model construction capability of I/O port coupling and the flexibility of basic object-oriented message passing.

Level access methodology is a general concept that defines a way of accessing models in a hierarchical and modular models as explained before. Name-directed coupling is an approach to model interconnection designed and implemented by extending DEVS formalism in order to implement level access methodology within DEVS-Scheme modeling and simulation environment.

Level access methodology provides parallel access to models within modular, hierarchical modules by broadcasting the desired operations (e.g. querying information, storing data) to all the models below a certain desired hierarchical level. This desired hierarchical level, expressed as a module name, is determined by the specificity of information in the message about the location of receiver models. The more specific is the location, the lower the hierarchical level of the modules and the narrower the resulting search space becomes. One important aspect of the level access methodology is its invariance to changes in components within the designated level. This advantage is illustrated in Section 5.

This paper is organized as follows. Section 2 provides an overview of the necessary background knowledge. In Section 3, modeling of an analogical reasoning system is provided as an application example. Section 4 gives a formalism for, and detailed design of, name-directed coupling. Finally Section 5 shows how level access methodology can be used within an example automated test facility (ATF) model. Some simulation results are shown to demonstrate the utility of the level access methodology in this context.

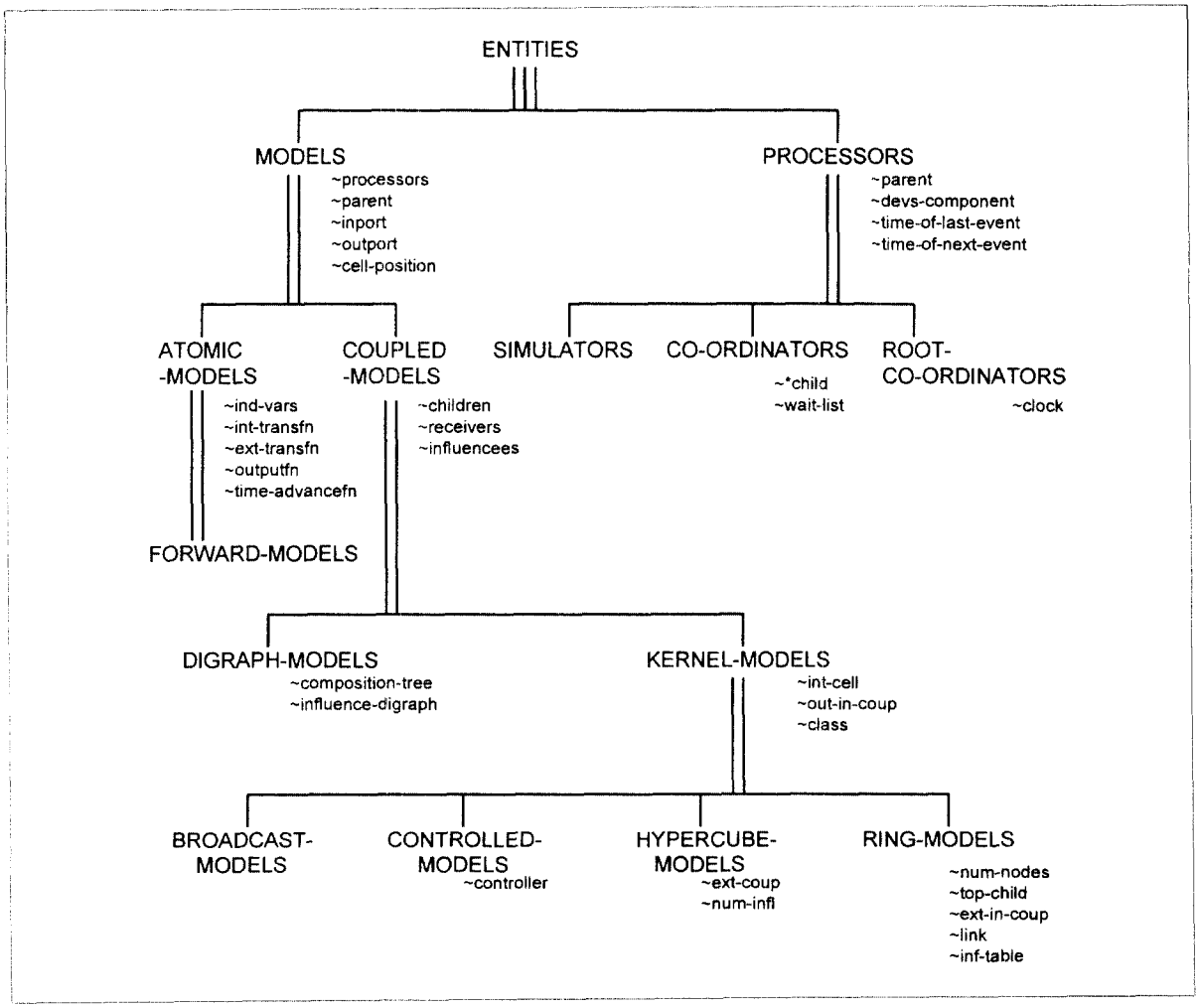
2. DEVS-Scheme simulation environment

DEVS-Scheme is a modular hierarchical discrete event simulation environment implemented in object oriented Scheme language which runs on UNIX and DOS compatible computers. The environment realizes the DEVS formalism [10, 12, 11, 2], a theoretically well-grounded means of expressing hierarchical, modular discrete event simulation models. DEVS-Scheme is implemented as a shell that sits upon Scheme language in such a way that all the underlying Lisp-based and object oriented programming language features are available to the users. The result is a powerful basis for combining AI and simulation techniques. 〈Figure 1〉 shows DEVS-Scheme class hierarchy.

All the classes in DEVS-Scheme are subclasses of the universal class *entities* which provides tools for manipulating objects in these classes. The inheritance mechanism ensures that such general facilities need only be defined once and for all. Models and processors, the main subclasses of entities, provide the basic constructs needed for modeling and simulation. Models are further specified into more

specific classes. Class processors, on the other hand, have three specializations: simulators, coordinators, and root-coordinators which serve to handle all the simulation needs. The class *atomic-models* realizes the atomic-level of the DEVS model formalism as explained in section 2.1. The class *coupled-models* is a major class which embodies the hierarchical model composition constructs of the DEVS formalism. *Digraph-models* and *kernel-models*, the subclasses of coupled-models, enable specialization of coupled-models in a specific way. Digraph-models provide a means of specifying coupled-models which are composed of a finite set of explicitly given components with explicitly specified coupling. On the other hand, kernel-models provides a means of specifying coupled-models containing arbitrary numbers of components coupled in a uniform manner. Different specialized classes of kernel-models realize different internal and external coupling schemes. Greater detail on DEVS-Scheme is in [12, 13, 11].

DEVS-Scheme can be used to develop DEVS models and save them in a model base for later use. To organize such



〈Figure 1〉 Class hierarchy in DEVS-Scheme

models into a model base, a model base management system is highly desirable. The SES (system entity structure) formalism is one such tool for model base management. ESP-scheme is a realization of the SES formalism developed by Zeigler [10] in a LISP-Based, object-oriented programming environment. The ESP-Scheme supports specification of the structure of a model, pruning the structure to the reduced one, and transforming the structure to a simulation model by synthesizing components models in the model base developed by using DEVS-Scheme [6].

2.1. DEVS formalism

In the DEVS environment, a system has a time base, inputs, states, outputs, and functions. The system functions determine next states and outputs based on the current states and inputs. In the formalism, an atomic model is defined by a structure:

$$M = \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a \rangle$$

where X is an external input set, S is a state variable set, Y is an external output set, δ_{int} is an internal transition function, δ_{ext} is an external transition function, λ is an output function, and t_a is a time advance function.

Several atomic models can be coupled to build a more complex model, called a coupled model. A coupled model tells how to couple several models together to form a new model. The DEVS formalism also defines a coupled model in modular form as a structure:

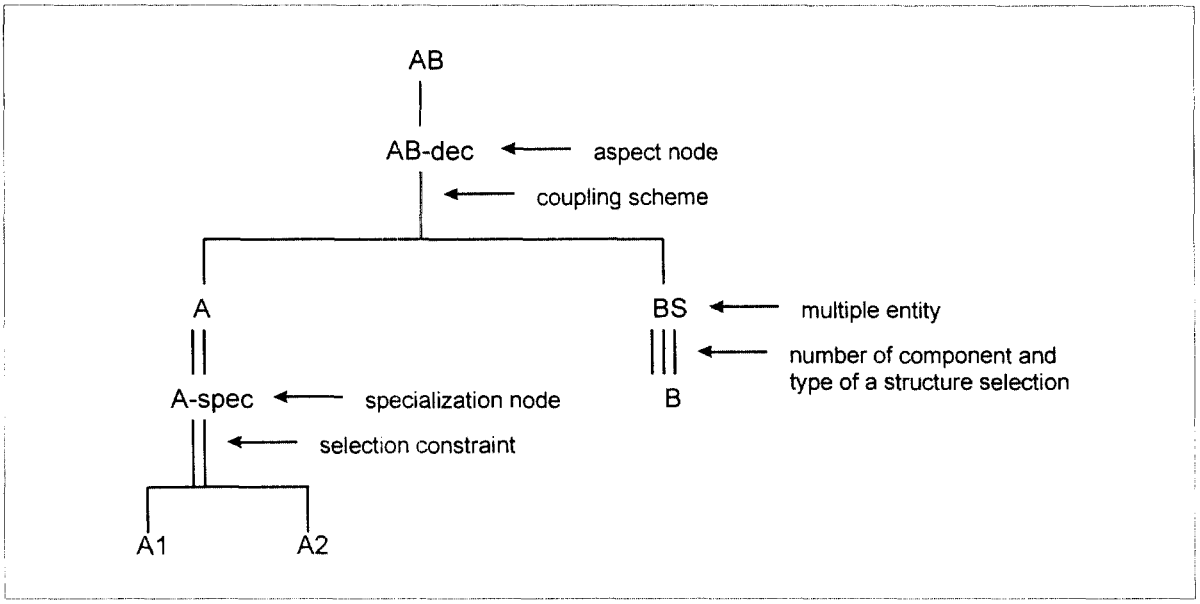
$$DN = \langle D, \{M_i\}, \{I_i\}, \{Z_{i,j}\}, select \rangle$$

where D is a set of component model names; for each i in D, M_i is a component model, I_i is the set of influencees of i, $Z_{i,j}$ is an output translation function (the *i-to-j* output translation), and *select* is a tie-breaking function. Such a coupled model can itself be employed in a larger coupled model.

2.2. SES formalism

A SES is a structural knowledge representation scheme that contains knowledge of decomposition, taxonomy, and coupling relationships of a system necessary to direct model synthesis.

There are three types of nodes in the SES - *entity*, *aspect*, and *specialization* - which represent three types of knowledge about the structure of systems. The entity node, having several aspects and/or specializations, corresponds to a model component that represents a real world object. The aspect node (a single vertical line in the labeled tree of 〈Figure 2〉) represents one decomposition of an entity. Thus the children of an aspect node are entities, distinct components of the decomposition. The specialization node (within a double vertical line in the labeled tree of 〈Figure 2〉) represents a way in which a general entity can be categorized into special entities. A *multiple entity* represents the set of all members of an entity class and it is a special entity that consists of a collection of homogeneous components. Such components are a *multiple decomposition* of the multiple entity. The aspect of such a multiple entity is called a *multiple aspect* (triple vertical lines in the labeled tree of 〈Figure 2〉). A substructure of a SES is extracted for use as the skeleton for a model. This substructure is called pruned entity structure (PES) and the extraction operation of a PES from the SES is called *pruning (or pruning operation)*. There are only aspect type nodes and entity type nodes in a PES. All the multiple decomposition nodes and specialization nodes in SES are traversed and processed in pruning process. The process done at the specialization node is to select a special entity (either A1 of A2) and the process done at the multiple aspect is to decide the number of multiple decomposition (number of components in BS) and type of kernel models (broadcast models, controlled-models, etc.) that this multiple decomposition forms. For more on DEVS formalism and SES formalism refer to [10, 11].



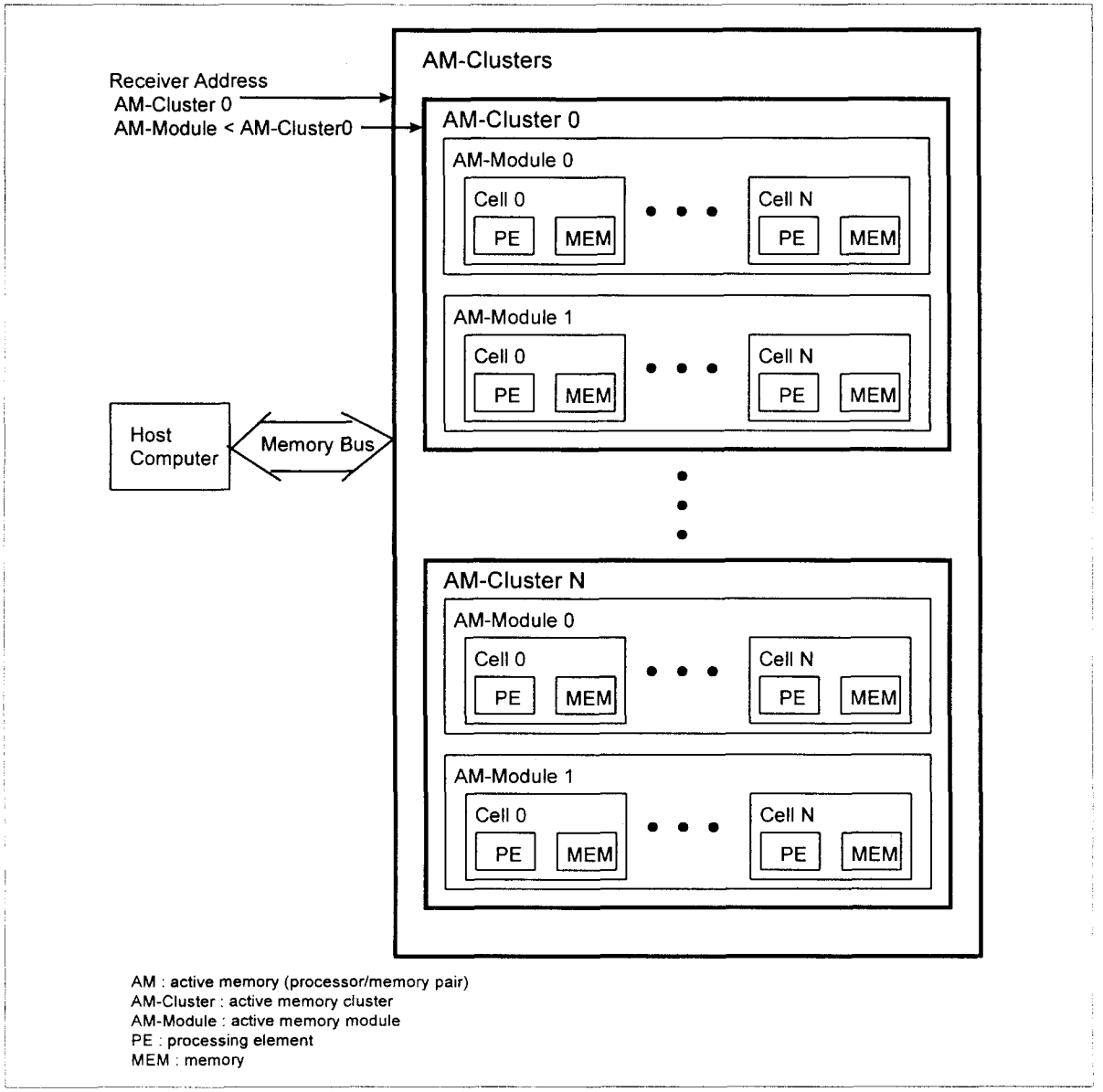
〈Figure 2〉 System Entity Structure(SES)

3. Application to the analogical reasoning system

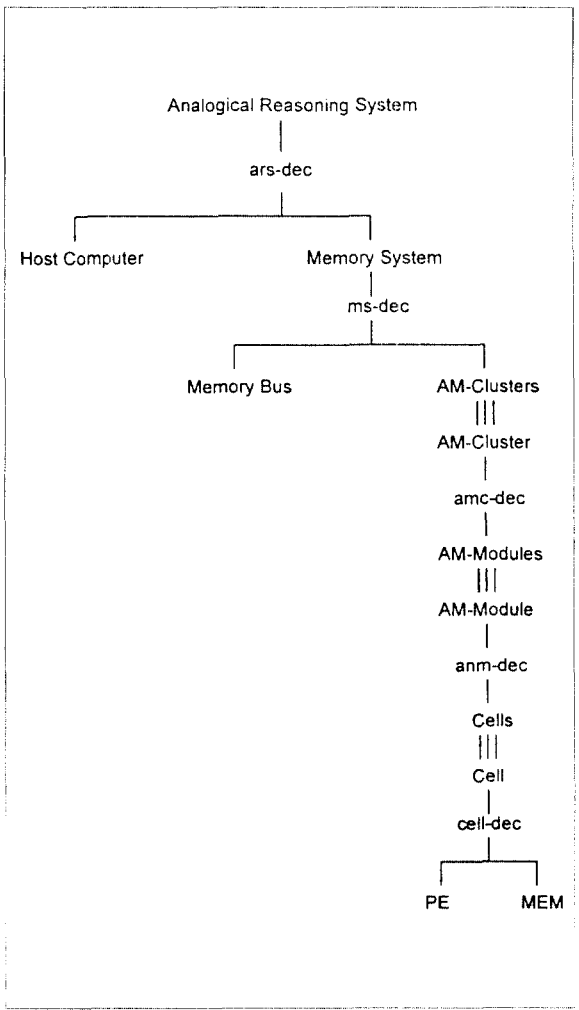
We now illustrate level access methodology with an application to the analogical reasoning, paradigm for problem solving within the field of artificial intelligence (AI). A system which employs analogical reasoning operates by transferring knowledge from past problem-solving cases to new problems that are similar to the past cases. The past cases known to the system are referred to as analogs. One of the key components of an analogical reasoning system is the associative memory. In order for analogical reasoning to be effective, the system must store a large amount of domain knowledge in the form of analogs. Effectively accessing these memories provides a means of retrieving analogs in an efficient manner. The retrieval process can be done in parallel by assigning one processor to each analog, as in the active memory architecture (e.g. massively parallel architecture like the connection machine [4], MIMD distributed memory architecture [8]). During associative retrieval, each analog is examined and matched against the input (called the target) by its own processor and those analogs sufficiently similar to the target are returned [8].

All memory cells within the cluster (AM-Clusters) in 〈Figure 3〉 are organized as the associative memory of the analogical reasoning system explained above. There are several active memory modules (AM-Modules) in an AM-Cluster. Every cell in an AM-Module is composed of a processing element (PE) and a memory (MEM). If it is known that the information to be retrieved is stored within the module AM-Cluster0, then the Host Computer sends query request messages to all the AM-Modules in AM-Cluster0, i.e., to all the cells within the AM-Cluster0 by specifying the destination address as AM-Cluster0. Likewise, if the information to be retrieved is stored within AM-Module1 of AM-Cluster0, then destination address of the message becomes AM-Module1 〈 AM-Cluster0 and the message is broadcast to all the cells within the AM-Module1 〈 AM-Cluster0. Thus, the access level and the module name are specified by AM-Module1 〈 AM-Cluster0. The "〈" symbol represents a hierarchical relationship between a compound module and its sub-modules in the hierarchical,

modular system. That is, the module left of the "〈" symbol is a child of the module right of the "〈" symbol. 〈Figure 4〉 is the SES for the system in 〈Figure 3〉. Since the hierarchical level of the system, which also represents a specific module, can be selected to be receiver of messages, the cells that are not involved in a retrieval operation are free from the operation and can be used for other processing. Thus, the area of cells that should be searched for the data retrieval is determined by how much the Host Computer knows about those analogs sufficiently similar to the target.



〈Figure 3〉 Associative memory of analogical reasoning system



〈Figure 4〉 System Entity Structure of analogical reasoning system

The analogical reasoning example illustrates a call-and-answer type of interaction commonly found in many distributed architecture systems. Such interaction involves varying degrees of messaging specificity between senders and receivers. For example in a client server architectures a client might seek to locate a file resident on one or more members of a group of servers. To do this, it might broadcast the file description to the group along with its return address. A server can acknowledge hosting the file in point-to-point communication fashion with the client. Subsequent file operation requests can also be sent in point-to-point fashion from the client to the host server.

Name-directed coupling and level access methodology provide convenient modeling constructs to capture such call-and-answer interactions.

4. Hierarchical name-directed coupling

As explained in Section 1 name-directed coupling is achieved by combining two different model interconnection approaches. These are port-to-port coupling and name-directed message passing exemplified in object oriented programming systems. Level access methodology within DEVS-Scheme environment is achieved by applying name-directed coupling to a hierarchical model. Such coupling is called *hierarchical name-directed coupling*. To implement it, new DEVS-Scheme I/O translation methods for hierarchical name-directed coupling were designed. These new translation methods provide message paths that are not achieved with the standard DEVS-Scheme translation methods. To discuss these new methods we first develop formal specifications of both old and new coupling scheme.

4.1. Formalism for standard and name-directed coupling

Before showing the formalisms for standard coupling and name-directed coupling, mathematical symbols used within the formalisms are explained in the following paragraphs.

For a coupled model *DN*, let the internal coupling of a coupled model be specified in the form

$$((i, p_i), (j, p_j)) \in \textit{Internal Coupling of Digraph Model} \quad (1)$$

$$\textit{or } (p_i, p_j) \in \textit{Internal Coupling of Broadcast Model} \quad (2)$$

The format of an internal coupling is specified as in (1) or (2) depending on a type of the coupled model. The internal coupling of a digraph model (1) states that output port p_i of component i is coupled to input port p_j of component j. For

broadcast models (2) the coupling does not depend on which pair of components is involved.

As in [10, 11], δ_{in} describe an internal transition of *DN*, let i^* be the component selected from those imminent as the one to undergo its internal state transition and to send outputs to its influencees.

Let (s'_j, e'_j) be the new state of j after *DN*'s internal state transition. So the new state s' of the coupled model is represented as $\delta_{in}(s) = s' = (\cdots, (s'_j, e'_j), \cdots)$, where δ_{in} is the internal transition function of the coupled model and s is the model state before the internal transition.

Let e be the elapsed time with the $0 \le e \le ta(s)$, where $ta(s)$ is the time advance function.

First we describe how an internal transition works for the standard coupling employed in DEVS-Scheme. Then we extend this description to handle name-directed coupling. In the standard coupling (or standard DEVS-Scheme coupling) the output function of each component model, λ_i is defined as $\lambda_i : S_i \rightarrow Y_i$, where S_i is the set of sequential states and

$$Yi \subseteq Pi \times Vi \tag{3}$$

Here Y_i is the set of external event types generated as output, P_i is the set of output ports within Y_i, and V_i is the set of output values within Y_i.

The new state (s'_j, e'_j) of the component j after the internal transition of *DN* is

$$(s'_j, e'_j) = \begin{cases} (\delta_{in}(s), 0) & \text{if } j = i^* \quad \text{i} \\ (\delta_{ext,j}(s_j, e_j+ta(s), x_{i^*j}), 0) & \text{for } j \in I_{i^*} \quad \text{ii} \\ (s_j, e_j+ta(s)) & \textit{otherwise} \quad \text{iii} \end{cases} \tag{4}$$

where in ii, $((i^*, p_{i^*}), (j, p_j)) \in$ *Internal Coupling*

where,

$$x_{i^*j} = (p_j, v_{i^*}) \tag{5}$$

$$y_{i^*} = \lambda_{i^*}(s_{i^*}) = (p_{i^*}, v_{i^*}) \tag{6}$$

(4).i specifies that the internal transition of the imminent component model i^* is as its own internal transition function dictates; (4).ii specifies the effect of the signal sent by the imminent component on each of its influencees as determined by the external transition function of the influencees; (4).iii acknowledges the effect of time advance function on the elapsed time of all other components [10].

(5) indicates that the input x_{i^*j} is a port-value pair (p_j, v_{i^*}) where (6) indicates that v_{i^*} is the value placed by i^* on its port p_{i^*}. Since there is a coupling from p_{i^*} to p_j as in (4).ii the output value v_{i^*} on p_{i^*} appears as input on p_j.

The formalism for internal transition may be better understood in the form of a pseudo coded algorithm in ⟨Figure 5⟩. The algorithm starts with component i^* generating its output. Then it checks relationship of j to i^* for all components j.

Depending this relation, the new states (s'_j, e'_j) are computed. The algorithm is completed when all js are considered.

In name-directed coupling, the output function of each component model, λ_i is defined as $\lambda_i : S_i \; Y_i$, where

$$Y_i \subseteq P'_i \times V_i \tag{7}$$

where $P'_i = N \times P_i$. Here N is a set of symbols, later to be interpreted as model name specifications.

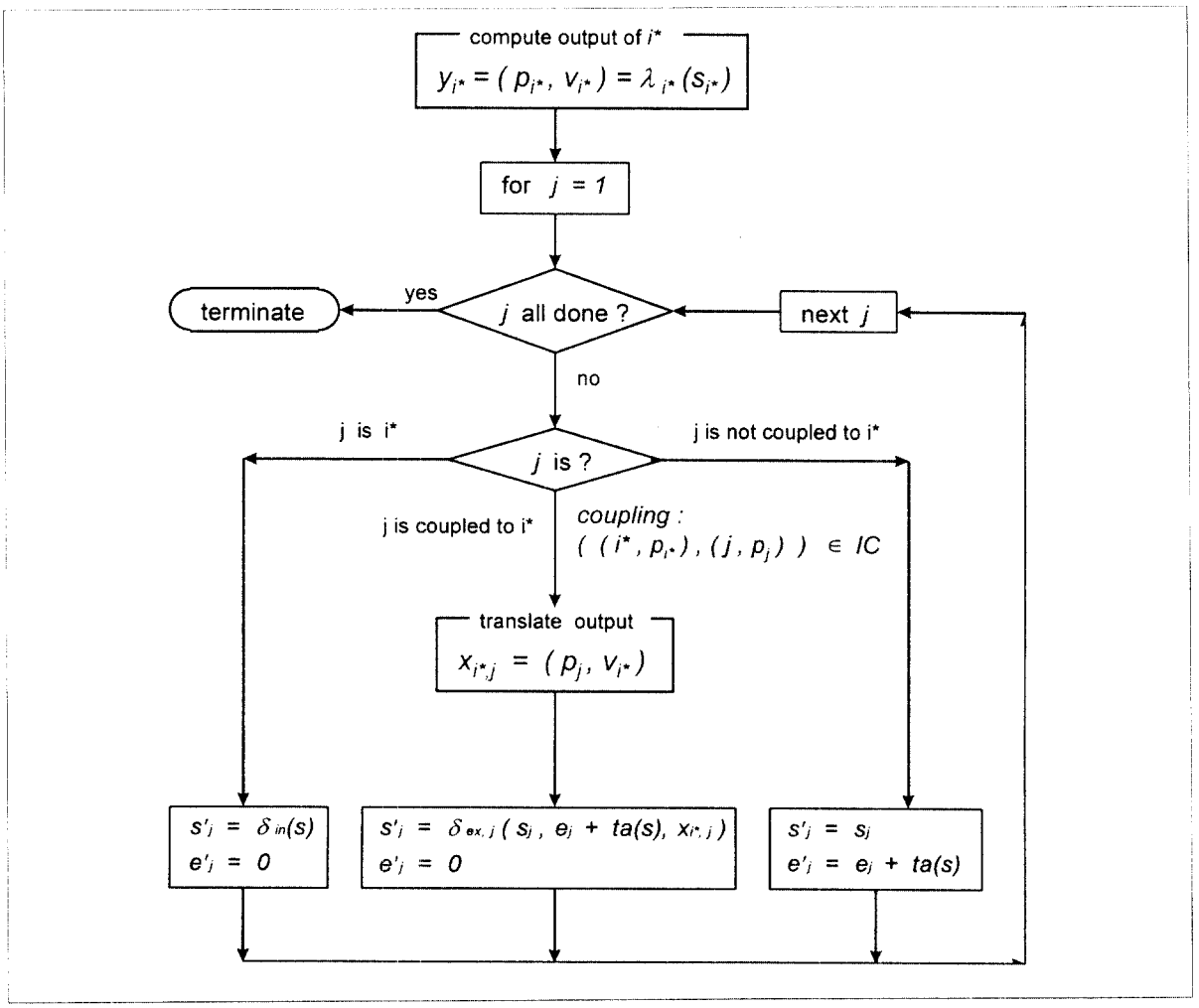
The output generated by i^* in state s_{i^*} is $y_{i^*} = \lambda_{i^*}(s_{i^*})$ and is of the form (j', p_{i^*}, v_{i^*}). This is interpreted as: component i^* places a value v_{i^*} on port p_{i^*} intending it to be received only by those components whose names match j'.

The flow chart for the internal transition of a coupled model with name-directed coupling is shown in ⟨Figure 6⟩. As shown at the top of the figure, the output computed by i^* is of the form (j', p_{i^*}, v_{i^*}) which includes a symbol j. In name-directed coupling, even if j is coupled to i^*, it may not receive an input x_{i^*j}. Only if component j matches j (shown in the middle of the figure) and there is a coupling from i^* to j does j receive x_{i^*j}. Except for these differences, the rest of the description is the same as for standard coupling.

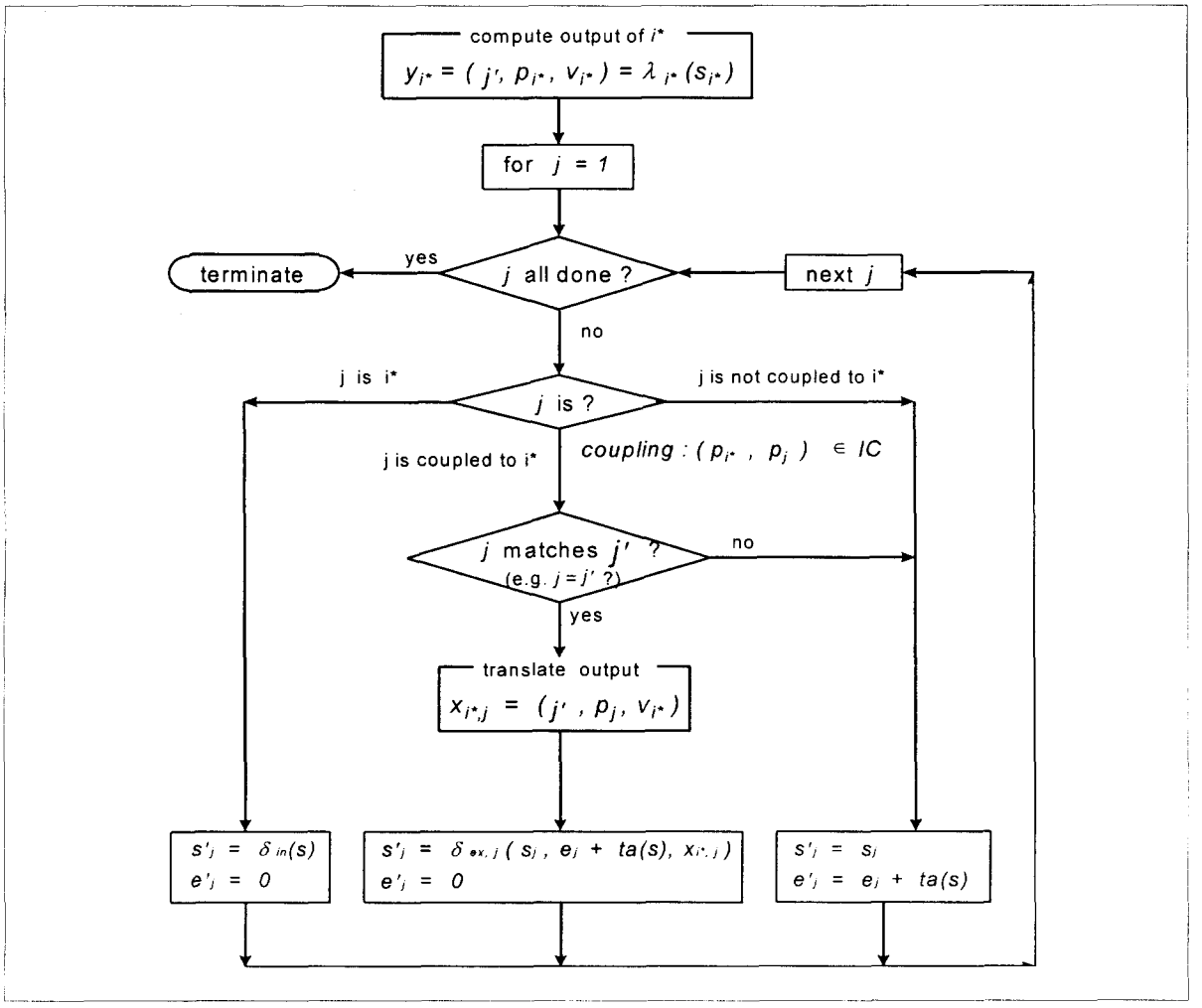
Note that j' is just a symbol which does not have any meaning until it is later interpreted as a specification of receiver names. So in (7) is equivalent to P_i in (3) as far as modularity is concerned since M_i (component model i) may be defined independently (of any other models). This equivalence indicates that name-directed coupling does not violate modularity.

Name-directed coupling becomes hierarchical name-directed coupling if the receiver name is specified in hierarchical name format (to implement level access methodology). When hierarchical name format is used, a receiver name, j' can be a subcomponent of j for the match in 〈Figure 6〉 to be satisfied. In this case the match indicates all components at the appropriate hierarchical level receive input from i^*

In sequential simulation, hierarchical name-directed coupling is much more efficient than standard coupling. In the latter all the submodels within a broadcast model must receive an external input (and thus have to go through their *external transitions*) once the external input arrives to the broadcast model. This is true even if only one model actually



〈Figure 5〉 Flow Chart for sequence of DEVS representation of standard coupling



〈Figure 6〉 Flow chart for sequence of DEVS representation of name-directed couping

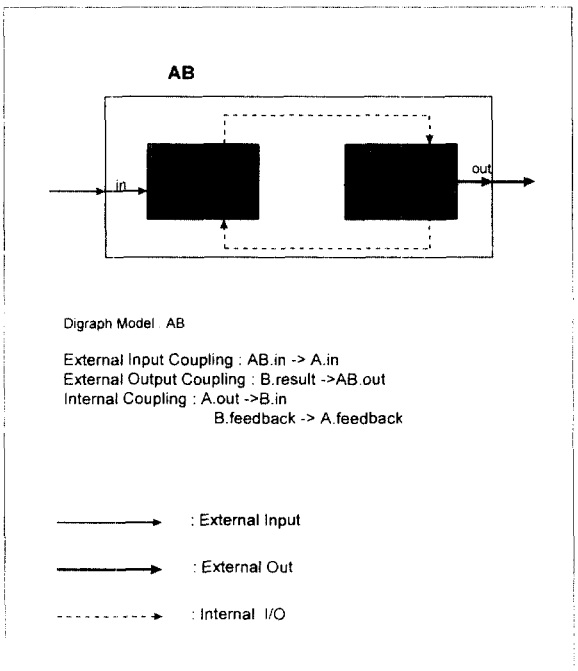
needs this input. In hierarchical name-directed coupling, we can specify a submodel that is to receive this input. In this case only one submodel goes through external transition, thus saving the execution time of each of the other components. In distributed or parallel simulation, this time is not critical since all external transition may be performed concurrently. However, even here the reduced messaging is significant.

4.2. Implementation of hierarchical name-directed coupling

To implement the new coupling formalism, new I/O port translation methods were implemented in DEVS-Scheme. Before showing how hierarchical name-directed coupling is realized, the standard DEVS coupling is discussed first.

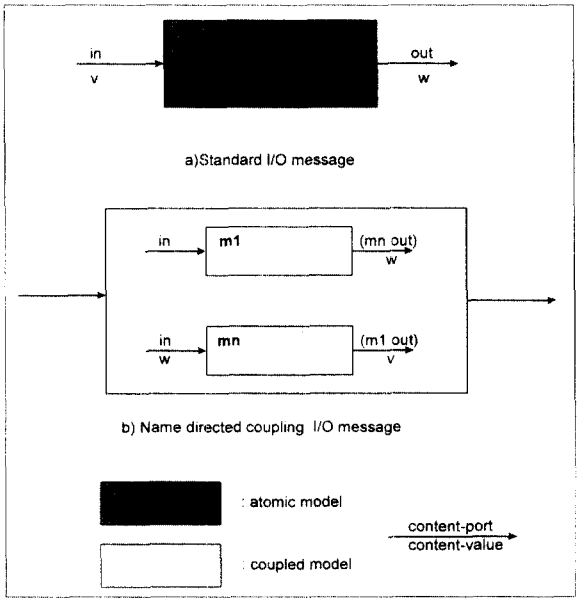
The coupling of DEVS components into a coupled model is based on its coupling table. There are three types of coupling information in such a coupling table : external input coupling, external output coupling and internal coupling. When an external input arrives at a coupled model, the input is sent to the components based on the external input

coupling in the coupling table, i.e., the external input coupling. Likewise, external output and internal I/O translation. 〈Figure 7〉 illustrates the coupling information and types of I/O of a digraph model AB.



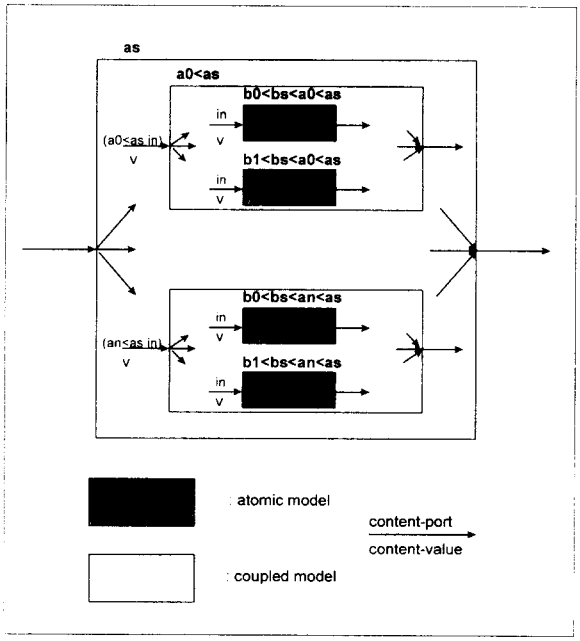
〈Figure 7〉 I/O port coupling of a digraph model AB

In standard coupling, atomic models only need to specify on which port the model wants to place its output message. The variables for the port name and output message are *content-port* and *content-value*, respectively, Thus the output message of an atomic model consists of a content-port and content-value corresponding to equation(3). In name-directed coupling the name specification must also be included in the content-port of an output message. 〈Figure 8〉 shows I/O message formats for standard coupling and name-directed coupling. The content-port is shown above the arrows and the content-value is shown below the arrows. In 〈Figure 8〉. a) the content-port of the output message is *out* and content-value is *w*. Whereas, as in 〈Figure 8〉.b), the content-port of output messages for models m1 and *mn* are *(mn out)* and *(m1 out)*, with content-value *w* and *v* respectively.



〈Figure 8〉 Standard coupling and name-directed couping I/O message format

As mentioned before level access methodology is achieved by applying the name-directed coupling to the hierarchical environment and this applied coupling is called hierarchical name-directed coupling. In hierarchical name-directed coupling a receiver models name is written in hierarchical name format which includes path and level information. This path and level information is needed for accessing a receiver model at a particular level within a hierarchically constructed model. 〈Figure 9〉 shows specification of a receiver name in hierarchical name format and broadcasting of a message according to the receiver name. If the receiver name is same as the name of the coupled model that received the message, the message is broadcast to all components. For digraph models as long as there are external input coupling between the coupled model and its components, the message is broadcast. The model *as* is coupled model constructed in hierarchical and modular fashion. All the atomic models within *as* receive the input message since the content-port of the input message for *as* is *(as in)*. If the input message to model *as* is *(a0⟨as in)* then only the models within *a0*



〈Figure 9〉 Hierarchical name-directed coupling message and its message format

〈*as*, i.e., *b0*〈*bs*〈*a0*〈*as* and *b1*〈*bs*〈*a0*〈*as*, receive the input message.

In hierarchical name-directed coupling a receiver in content-port is changed during port translation process by the translation methods in order to broadcast the message to submodels. 〈Figure 9〉 show change of the receiver name in content-port by translation methods as the hierarchical level is lowered during translation process. Initially the receiver name is *as*, then changed to *a0*〈*as* and *an*〈*as* at first translation. Finally *a0*〈*as* and *an*〈*as* are changed to *b0* 〈*bs*〈*a0*〈*as* and *b1*〈*bs*〈*a0*〈*as*, and *b0*〈*bs*〈*an*〈*as* and *b1*〈*bs* 〈*an*〈*as* respectively. This last change is not shown in content-port of the four models but the effect of this change are shown as the four models receiving the content-value *v*. The Figure assumes all the coupled models are broadcast models.

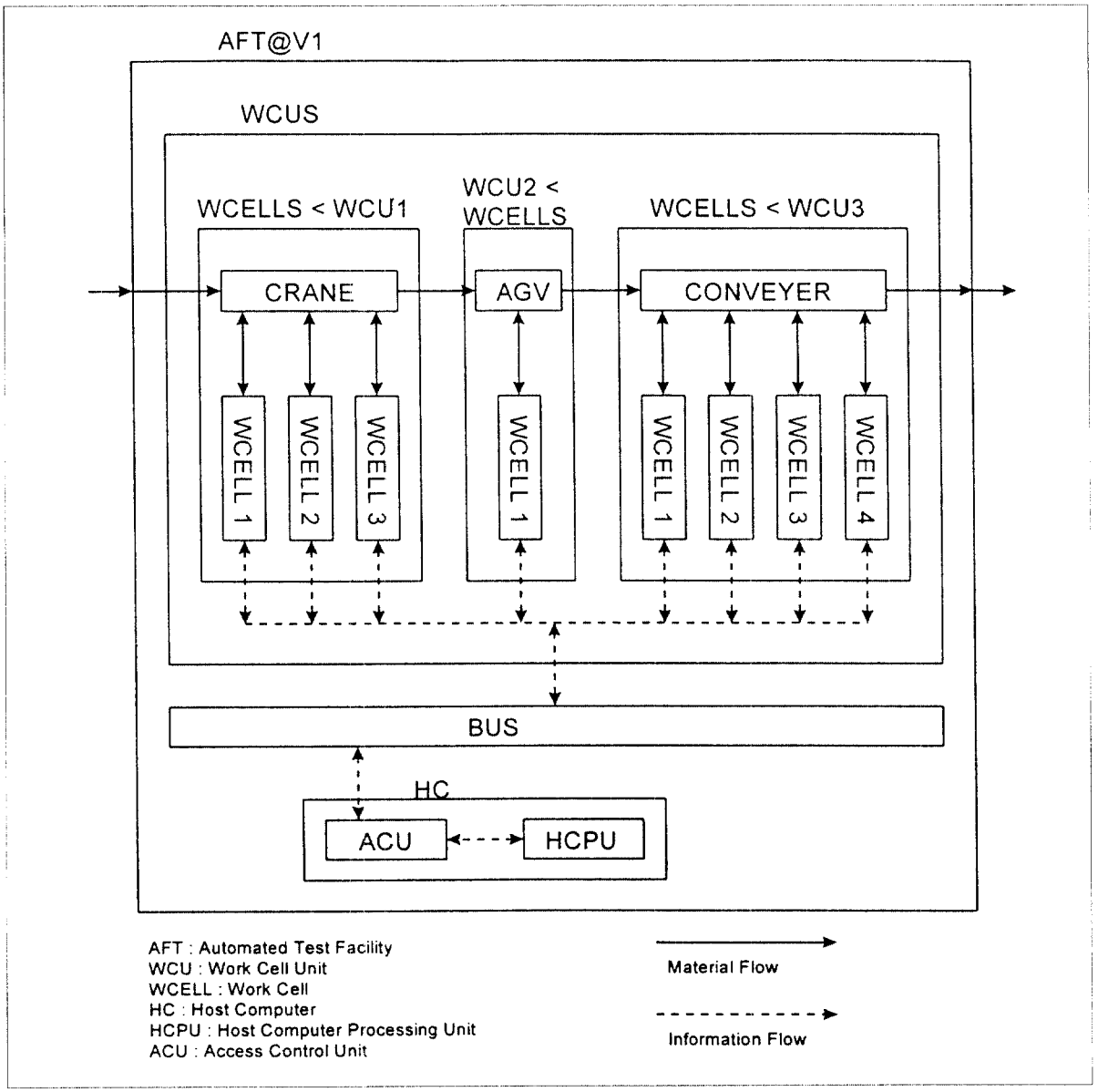
4.3 Level access methodology example(contd)

This section further explains how the hierarchical name specifies an access level and path within modular, hierarchical models of the analogical system shown in 〈Figure 3〉.

The multiple entities in the SES shown in 〈Figure 4〉 are pruned to broadcast models in the pruning operation. The hierarchical name of each model is generated after the pruning operation. An example, using the notation introduced earlier, is *cell0*〈*cells*〈*AM-Module0*〈*AM-Modules*〈*AM-Cluster0*. For the Host Computer to access all the models (cells) within *AM-Module0* of *AM-Cluster0*, it places its message on port *(AM-Module0*〈*AM-Modules*〈*AM-Cluster0 Host-Computer out)*. A receiver model which has the requested information can reply back to the sender model by using the senders name. In this way a cell can reply to Host Computer. Since Host Computer is not subcomponent model of *AM-Clusters*, it is hierarchically higher than the Cellmodel(e.g., Cell0). So the output message produced by *Cell0* is sent all the way up through the *AM-Clusters* and the down to Host Computer.

5. Application of level access to an Automated Test Facility Model

The level access methodology can also be applied to the Automated Test Facility(ATF)model shown in Figure10[3]. ATF is a facility for testing PCBs (printed circuit boards) with a centralized information processing system where all the processing data of PCBs at WCELLs(workcells)are sent to HC(host computer)for the monitoring the controlling of ATF system. As PCBs are routed within the system by the routing decision made at HC, a processing information of each PCB is sent to HC after completion of the processing at a certain WCELL. Two simulations are executed on ATF model , one exploiting the level access methodology and the other without it. 〈Table 1〉 compares these two simulation results. The message delivered by level access methodology is sent from HC to WCELLs by translation methods of hierarchical name-directed coupling. The content of the message for this simulation is to lower the test level in WCELLs in WCU1. The simulation run made without the message maintains the initial test level of WCELLs in WCU1. Whereas, in the other simulation the message changes the



〈Figure 10〉 Automated Testing Facility (ATF) model[3]

test level of WCELLs in WCU1 in the middle of simulation run. The change of testing level is made under the following assumption. The current test level is too high for the types of PCBs being tested. This causes unnecessary extra processings. The HC computer issued the message, between 150 and 200 time units, to the WCELLs in WCU1 to lower the test level. As a result of this message the testing times of tester type workcells in WCU1 are reduced and consequently, the production rate increases as shown in 〈Table 1〉. The simulation runs made above are purely for

〈Table 1〉 Simulation result of ATF.

	w/o lowering	w/ lowering
Avg.arrival rate [job/time unit]	0.055	0.055
Avg. flow time [time units]	156.46	134.11
Avg. WIP [jobs]	8.50	7.31
Production Rate [jobs/time unit]	0.014	0.023
% utilization of WCELL1 of WCU1	4.98	5.45
WCELL2	54.66	45.32
WCELL3	50.63	37.58
WCELL1 of WCU2	42.58	61.96
WCELL1 of WCU3	43.15	42.46
WCELL2	13.81	15.49
WCELL3	1.44	0
WCELL4	1.15	2.29

Simulation Period : 0-350[time units]

demonstration purposes but are indicative of the utility of message broadcasting using level access methodology. The following paragraphs explain how the hierarchical name is specified and discuss the advantages of using level access methodology. APPENDIX explains the I/O port interconnections of hierarchical name-directed coupling of this example that are not found in standard coupling.

Level access methodology is used for accessing from BUS to WCELL models within WCU1. The content-port of the output message generated by BUS is *(WCU1〈WCUS〈ATF@V1 out)*. Notice that the model names are not written in hierarchical name format due to the restricted space within the figure. The actual name of WCELL1, for example, is WCELL1〈WCELLS〈WCU1〈ATF@V1. Since the message using level access methodology is broadcasted to all the models within WCU1 it is up to each submodel whether to respond to this message or not, that is, models within WCU1 can be made to responded to the message by adding a port and a corresponding external input function for receiving the message. Models without the proper port ignore the message. So HC only needs to specify the destination fo the message as WCU1 even though the WCELL1 of WCU1 is a production store type workcell which does not perform testing. In other words, HC does not have to know that only WCELL2(incircuit tester) and WCELL3(function tester) of WCU1 are the tester type workcells to have its message properly understood. WCELL1 and CRANE ignored the message. Let us note the extra work needed in lowering the test level when level access methodology is not used.

In this case HC model must have the information on the type of each workcell in WCU1, e.g., WCELL1 is production store type, WCELL2 is incircuit tester type and WCELL3 is functional tester type. Based on these types messages are sent to all the tester type models. Note that the placement information for the routing is not needed in HC when level access methodology is used. This points to another advantage using level access methodology of HC design, invariance to changes in components within the designated level (within WCU1). For example, when the structure of WCU1 is changed (e.g. number of tester type workcells, order of workcesIls, etc) in subsequent simulation, HC model has to be given the new placement information of WCU1. Whereas, with the level access methodology, no modification is needed. This invariance of level access methodology greatly enhances modeling flexibility and convenience.

There is another way of delivering the messages to tester type models in WCU1. As in level access methodology we broadcast messages to all the models within WCU1 but let each WCELL model decide whether to respond to this message or not. However, this vroadcasting to all the model in WCU1 is not done automatically. All the models connedting from HC to WCELL models must bhve capavility of routing the messages, which means additional modeling effort within whete models(BUS and WCU1〈WCELLS). In this case extra simulation run time is also required as well as extra modeling activity. Since WCUS is vroadcast model, the message arrived at the model is vroadcasted to all the submodels of WCUS(e.g, WCU!, WCU2, and WCU3). This vroadcasting causes extra processing on WCU2 and WCU3 since the message is intended only to WCU1. As noted vefore, with level access methodology, the message can be delivered only to WCU1 without causing extra precessings at WCU2 and WCU3.

In either case there is extra modeling effort needed compared to the case when level access methodology is used.

Summary

A New coupling approach called name-directed coupling is defined by extending DEVS formalism. Name-directed coupling exploits the capabilities of object-oriented programming to provide a flexible communication paradigm that combines port-to port coupling with name-directed messaging. Level access methodology, within the DEVS modeling and simulation environment, is achieved by designing and implementing new translation methods which realize the design concepts of hierarchical name-directed coupling. The two example applications illustrate how effectively the proposed methodology can be used in modeling distributed intelligent processing systems and computerized flexible manufacturing systems.

References

[1] CACI Products Company, MODSIM II; The Language for Object-Oriented Programming, Reference Manual, 1990

[2] Conception, A.I. And Zeigler, B.P., DEVS Formalism: A Framework for Hierarchical Model Development, it IEEE Trans. On Software Engineering, vol. 14, no. 2, pp. 228-241, Feb.1988

[3] Cho, T.H., Rozenvlit, J.W. and Zeigler, B.P., Knowledge-Based Simulation Environment Techniques; A Manufacturing System Example, Control and Dynamic Systems, Academic Press, vol. 49, pp.191-239,1991

[4] Hillis, W.D., The connection machine, MIT press, Cambrige, Mass., 1985

[5] Klahr,P., Expressibility in ROSS, an Object-Oriented Simulation System, Proceedings of AI Applied to Simulation Conference, pp.136-139

[6] Kim,T.G. And Zeigler,B.P.,Knowledge-based environment for investigating multicomputer architectures, Information and Software Technology, vol 31,no 10,dev. 1989

[7] Lee,C., Kim,T.G., Zeigler, B.P., A Hierarchical Meghodology for Multilevel AI Computer Architectures, Proc. Of the Society of Computer Simulation Eastern Multiconference, Nashvile, TN, April 1990

[8] Sage, A.P., Analogical Reasoning, Concise Encyclopedia of Information Processing in Systems and Organizations, pp. 1-9, Pergamon, 1990

[9] Uhr,L., Multi-Computer Architectures for Artificial Intelligence: Toward Fast, Rovust, Parallel Systems, NewYork, NY: JohnWiley and Sons, 1987

[10] Zeigler, B.P., Multifacetted Modeling and Discrete Event Simulation, Academic Press, Orlando, FL, USA, 198A, 1984

[11] Zeigler, B.P., Object-Oriented Simulation with Hierarchical, Modular Models, Academic Press, San Diego, CA, USA, 1990

[12] Zeigler, B.P., DEVS-Scheme: Alisp-Vased Environment for Hierarchical Modular Discrete Events Models, Tech. Rep. AIS-2, CERL Lab., Dept. of ECE, Univ. of Arizona, Tucson, 1986

[13] Zeigler, B.P., hierarchical, Modular Discrete Event Modeling in an Object Oriented Environment, Simulation J., vol. 49:5, pp. 219-230, 1987

APPENDIX

Here are couplings used within ATF model that are not found in standard coupling in applying level access methodology to ATF model. 1) External I/O of Controlled Models: The communication links established between BUS and WCELLs in WCU1 are not through the controller (CRANE) of WCU1, but by direct links as shown in 〈Figure 12〉.b). 2) External I/O of Broadcast models: The message is delivered only to WCU1 despite that WCU2 and WCU3 are also sub-models of broadcast model SCUS as shown in 〈Figure 11〉.b). 3)Broadcasting to submodels: This application is shown in broadcasting the message to all the models (WCELLS and CRANE) in CCU1. The message is broadcast to any level and any module(s) regardless of model types. i.e.,.even though WCU1 is a controlled models type, the

message is broadcast to all the submodels. 〈Figure 9〉 shows this capability. 1),2) and 3) are capabilities provided by hierarchical name-directed coupling broken down into parts to show new I/O port interconnections used in this example.

● 저자소개 ●

조대호
1983년 성균관대학교 전자공학과 졸업(공학사)
1987년 University of Alabama 전자공학과 졸업(공학석사)
1993년 University of Arizona 전기 및 컴퓨터공학과 졸업(공학박사)
1993-1995년 경남대학교 전자계산학과 전임강사
1995-현재 성균관대학교 정보공학과 조교수
관심분야 : 컴퓨터시뮬레이션, 지능형시스템, 공장자동화

이칠기
1980 성균관대학교 전자공학과 졸업
1979-1983 한국방송공사 근무
1985 Arizona State University 전기 및 컴퓨터 공학과 석사
1990 University of Arizona 전기 및 컴퓨터 공학과 박사
1990-1995 삼성전자 수석연구원
1995-현재 성균관대학교 전기전자 및 컴퓨터 공학부 조교수
관심분야 : 컴퓨터 시뮬레이션, 객체지향 모델링, 공장자동화, 전문가 시스템