Objective-driven DEVS Modeling Using OPI Matrix for Performance Evaluation of Discrete Event Systems

Tag Gon Kim* and Chang Ho Sung**
Dept of EECS, KAIST
Taejon, Korea
tkim@ee.kaist.ac.kr*; chsung@smslab.kaist.ac.kr**

Keywords: DEVS, OPI Matrix, Objective-driven Modeling, War Game Modeling

Abstract

DEVS (Discrete Event Systems Specification) models are widely employed for simulation-based performance evaluation of discrete event systems. Performance modeling of such a system requires a pruning of a set of components, or objects, within the system in its subset which are eventually transformed into simulation models. Such modeling is objective-driven in that simulation models may be abstract enough to measure performance indices which reflect simulation objectives. This paper proposes a practical tool, called an OPI (Object-Performance Index) matrix which is useful for such pruning in the objective-driven DEVS modeling. It also presents transformation of the pruned objects to DEVS models for performance simulation. An overall procedure for the objective-driven DEVS modeling from given objectives is presented and a simple example is illustrated.

1. INTRODUCTION

Performance evaluation is an important activity in design of a new system as well as in tuning of an existing one. Simulation may be a powerful, or sometimes the only means, for the activity. Performance of a system is usually evaluated at the discrete event system level for which a discrete event simulation model of the system is employed. DEVS (Discrete Event Systems Specification) formalism provides a framework for simulation modeling of a system at the discrete event system level in hierarchical modular manner [Zeigler, 2000]. Because of effectiveness as well as efficiency of the framework DEVS models are widely used for simulation-based performance evaluation of discrete event systems [Ahn, 1993].

Performance modeling is objective-driven in that such a model reflects a subset of a real system in its structure and/or in its behavior, which may be different depending on simulation objectives. Moreover, simulation of different performance models may require experimental frames associated with the models for simulation. Thus, multiple objectives results in multiple performance indices which in turn results in multiple models with associated experimental frames, each of which measures a designated performance index [Zeigler, 1984].

This paper first proposes a practical tool, called an OPI (Object-Performance-Index) matrix which represents relationship between performance indices and components, or objects, in a real system. The purpose of the matrix is to identify objects and associated attributes in the system which is eventually transformed in DEVS models to measure performance indices derived from simulation objectives. It then presents an overall procedure for objective-driven DEVS modeling for performance evaluation of discrete event systems.

This paper is organized as follows. Section 2 presents simulation-based performance evaluation and DEVS framework in brief. Section 3 introduces an OPI matrix and objective-driven modeling for performance evaluation. An overall procedure for DEVS modeling for performance evaluation is given in Section 4. A case study and conclusion are presented in Section 5 and Section 6, respectively.

2. PERFORMANCE MODELING AND DEVS FRAMEWORK

2.1. Simulation-based Performance Evaluation

Performance evaluation of a system may be done by simulation for which either an analytic model or a simulation model can be employed. An analytic model of a discrete event system is a set of equations in a closed form, which represent relationship between system parameters. Such equations are derived based on a set of assumptions in system's behavior which sometimes are unrealistic. Simulation of such a model can measure average performance of the system only at a steady state. The main advantage of employing the model is that its simulation time is much less than that of a simulation model. An example for such models includes Markov chain models [Bolch, 2006].

On the other hand, a simulation model of a discrete event system is represented by a set of objects each of which interacts with each other by means of events or messages. An object is a counterpart of a real system component which is represented by a set of variables and associated operations defined on them. Thus, an object can represent details of system's behavior in an algorithmic form in which all unrealistic assumptions made in an analytic model may be avoided. Although simulation of the model is expensive in time its result would be much more accurate than that of a corresponding analytic model. Moreover, such simulation can measure system performance in any operational range including in a transient period, which is very important in tuning system parameters.

Discrete event simulation models can be specified either by informal world view or by formalism. The former is based on how a modeler views behavior of a real system in his own way. It includes the event-oriented view and the process-oriented view which are basis for event-oriented and process-oriented discrete event simulation languages, respectively[Kiviat, 1973][Pritsker, 1984]. One the other hand, the latter provides a sound semantics based on mathematics by which a model is specified. A well-known such formalism is the DEVS formalism which this paper deals with. An advantage of a formal model is that the model can be not only simulated but also manipulated mathematically. For example, a DEVS model for performance evaluation can be transformed into an analytic queuing model in certain condition. Such mathematical transformation proposes a new methodology for combining simulation of DEVS models with analytic queuing models, which reduces simulation time markedly [Ahn, 1996].

2.2. DEVS Framework

The DEVS formalism specifies discrete event models in a hierarchical and modular form. With this formalism, one can perform modeling more easily by decomposing a large system into smaller component models with coupling specification between them. There are two kinds of models: atomic model and coupled model [Zeigler, 2000].

An atomic model is the basic model and has specifications for the dynamics of the model. Formally, a 7-tuple specifies an atomic model M as follows.

$$M = < X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta >,$$

where

X: a set of input events;

Y: a set of output events;

S: a set of sequential states;

δ_{ext}: $Q \times X \rightarrow S$, an external transition function,

where $Q = \{(s,e) | s \in S, 0 \leq e \leq ta(s)\}$ is the total state set of M;

δ_{int}: $S \rightarrow S$, an internal transition function;

λ: $S \rightarrow Y$, an output function;

ta: $S \rightarrow R^{+}_{0,\infty}$ (non-negative real number), time advance function.

A coupled model provides the method of assembly of several atomic and/or coupled models to build complex systems hierarchically. Formally, a coupled model is defined as follows.

$$DN = < X, Y, M, EIC, EOC, IC, SELECT >,$$

where

X: a set of input events;

Y: a set of output events;

M: a set of all component models;

$EIC \subseteq DN.X \times \cup M.X$: external input coupling;

$EOC \subseteq \cup M.Y \times DN.Y$: external output coupling;

$IC \subseteq \cup M.Y \times \cup M.X$: internal coupling;

SELECT: $2^{M} - \emptyset \rightarrow M$: tie-breaking selector.

An overall system consists of a set of component models, either atomic or coupled, thus being in hierarchical structure. Each DEVS model, either atomic or coupled model, has correspondence to an object in a real-world system to be modeled. Within the DEVS framework, model design may be performed in a top-down fashion; model implementation in a bottom-up manner.

2.3. Simulation of DEVS Models

Simulation of a DEVS model can be done by communicating hierarchical abstract simulators the architecture of which is the same as the DEVS model architecture. The abstract simulators are a set of distributed simulation algorithms which can be implemented in a sequential as well as in a distributed computing environment [Zeigler 1984]. Since DEVS models are developed in object-oriented manner, it is natural that an environment for DEVS M&S is implemented in an object-oriented programming language such as C++ or java. In fact, the first such C++ implementation is DEVSim++ [Kim, 1992] in which modeling facilities and abstract simulators are explicitly separated. In the environment modeling facilities are opened to modelers; abstract simulators are not accessible externally. The facilities allow modelers to create DEVS models of type atomic or coupled, which are subclasses of the class atomic or coupled model defined within DEVSim++.

Within the DEVSim++ environment modelers can develops DEVS models using modelers' interface which is a set of APIs to specify DEVS models in DEVS semantics. Thus, APIs for specification of DEVS models are defined such that there is a one-to-one correspondence between APIs and functions in the formalism. For example, APIs for specification of four characteristic functions in an atomic DEVS are TimeAdvanceFn(), ExtTransFn(Message &, e), IntTransFn(), and OutputFn(Message &). APIs for specification of coupled DEVS models are similarly defined. Once DEVS models are developed by using facilities and APIs simulation of such models can be performed by abstract simulators embedded in DEVSim++.

Different implementations for DEVS M&S are available in public domains and effort for standardization of DEVS modeling/simulation environments is on-going by DEVS Standardization Organization [DEVS-STD 2005].

3. OBJECTIVE-DRIVEN MODELING AND OPI MATRIX

3.1. Objective-driven Modeling

Any simulation should start with simulation objectives without which a level of details in model representation would not be determined, thereby being impossible in modeling. Such objectives are to obtain answers for questions on system behavior and/or on its efficiency. If questions are on system's efficiency answers would be obtained by simulation of performance models. In such a case, simulation objectives should explicitly specify which performance indices are to be measured. Once a performance index to be measured is identified a simulation model including a system model and an associated experimental frame can be developed to measure the index. Experiments of multiple simulation runs of the model show the desired performance index.

Figure 1 shows the concept of objective-driven modeling in performance evaluation in which a system may have m models and k experiments may be performed for $1 \le m \le k$. Note that in spite of a simulation model per a performance index one system model would measure more than one performance indices in some cases. Such cases include one in which two variables in a sub-model are independent functions of two performance indices. Note also that a sub-model itself can be a common component of two models. Similarly, an experimental frame would measure more than one performance indices. Generally, an overall simulation model is developed to measure more than one performance indices which reflect multiple objectives for simulation. The overall model is an assembly of components each of which measures only one performance index.

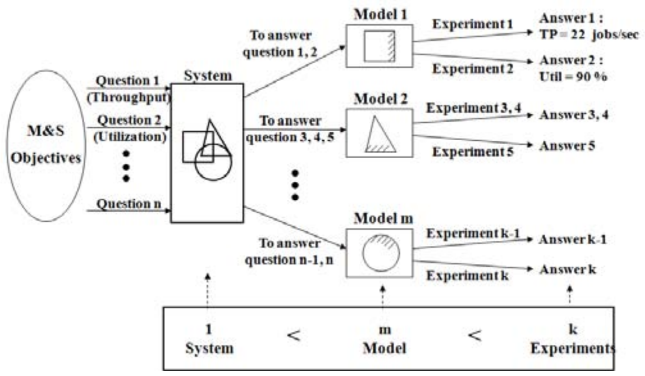
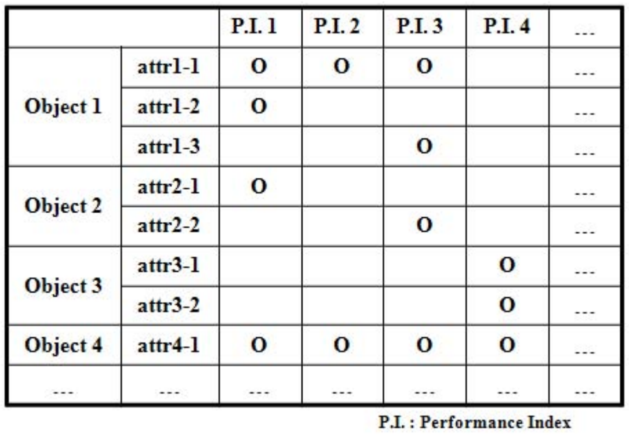


Figure 1 Objective-driven Modeling Simulation

3.2. OPI Matrix

The DEVS framework is known to be compatible with the object-oriented world view in which a system is viewed as a collection of components, or objects [Kim, 1995]. Such objects would be mapped into DEVS models of either atomic or coupled class. However, as explained in the last sub-section, not all objects are to be mapped. Instead, mapped are objects that are related to performance indices to be measured derived from simulation objectives. Moreover, only the minimum information that contributes to calculate the indices are to be associated with a given object to be modeled.

The OPI matrix is a practical tool to select such objects and associated attributes. Simply speaking, the matrix has rows of objects and columns of performance indices to be measured. Each row has sub-rows each of which lists an attribute of an object in the row. Figure 2 shows a prototype of the OPI matrix. Let a position ((i, j), k) of an OPI matrix be an intersection between the j-th attribute of the i-th object and the performance index k. Then, OPI((i, j), k), the value at ((i, j), k) of OPI, would be true, or 1, if the j-th attribute of the i-th object contributes to calculate the k-th index. Otherwise, it would be false. Thus, the matrix clearly identifies which objects are to be mapped into DEVS models with given performance indices. Moreover, it gives information on what attributes in an object are to be attached to such models.



		P.I. 1	P.I. 2	P.I. 3	P.I. 4	---
Object 1	attr1-1	O	O	O		---
	attr1-2	O				---
	attr1-3			O		---
Object 2	attr2-1	O				---
	attr2-2			O		---
Object 3	attr3-1				O	---
	attr3-2				O	---
Object 4	attr4-1	O	O	O	O	---
---	---	---	---	---	---	---

P.I. : Performance Index

Figure 2 OPI Matrix

Let explain how to identify objects and associated attributes which are to be modeled to measure a given performance index using the OPI matrix in Figure 2. 1s and 0s in the OPI matrix self-explains such identification. Consider an identification of objects and associated attributes for performance index PI 1. Then, the identification is to find a set of (i, j)s such that OPI((i, j), 1) = 1, where i and j are indices of an object and an associated attribute, respectively. Clearly, (i, j)s which satisfy the constraint are: (1,1), (1,2), (2,1) and (4,1). Thus, PI 1 is function of attributes 1 and 2 of object 1, attribute 1 of object 2, and attribute 1 of object 4. Objects and associated attributes for calculation of other performance indices can be identified in a similar manner. Note that OPI((1,1), 1) =

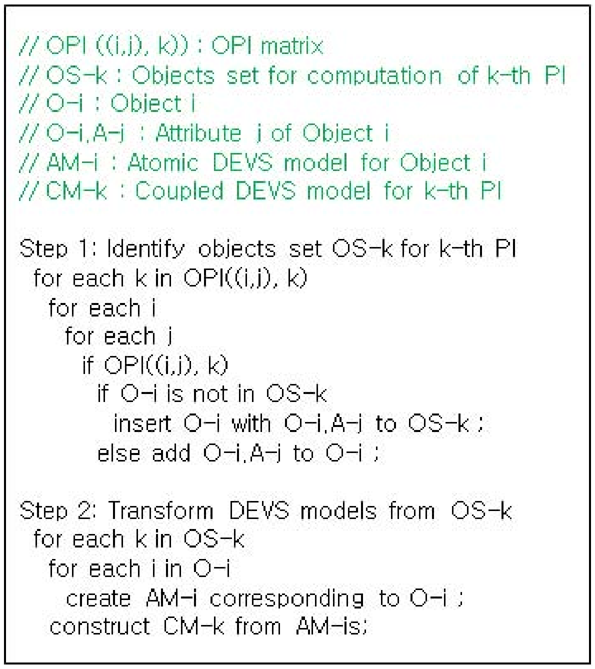
OPI((1,1), 2) = OPI((1,1), 3) = 1, meaning that object 1 with associated attribute 1 is contributed to calculate three performance indices PI 1, PI 2 and PI 3. Thus, a model corresponding to object 1 is a common sub-model among three models to calculate the three PIs.

4. PROCEDURE FOR DEVS MODELING FOR PERFORMANCE EVALUATION

4.1. Transformation of Objects to DEVS Model

An OPI matrix OPI((i, j), k) contains all performance indices which reflect modeling objectives as well as all objects which contribute to measure the indices. Note that each PI may be measured by an independent DEVS model. Thus, theoretically an overall performance model consists of k DEVS models with no interaction between them. However, there may be a component DEVS model which belongs to more than one DEVS models of k PIs. In such a case the component can be shared among the DEVS models, thus reducing the size of the overall performance model. However, a state set of the component in one model may be different from that of the other model. Thus, the sharing may require a union of state sets of the same component.

Transformation of k independent DEVS models is strait-forward; the reduced overall performance model may require some extra work. We first present the transformation procedure. Then we discuss the reduction. Figure 3 shows a procedure for transformation of an OPI((i,j), k) matrix with k PIs into a set of k DEVS models, each for measurement of one PI.



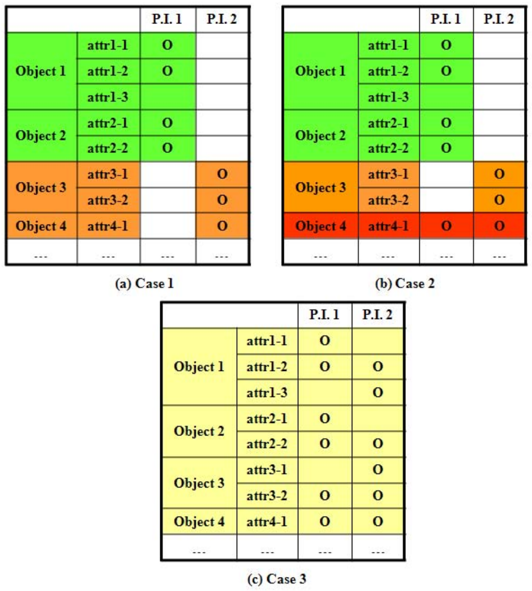
```
// OPI ((i,j), k)) : OPI matrix
// OS-k : Objects set for computation  of k-th  PI
// O-i : Object i
// O-i,A-j  : Attribute  j of Object  i
// AM-i : Atomic DEVS model for Object  i
// CM-k : Coupled DEVS model for k-th PI

Step 1: Identify  objects  set OS-k for k-th PI
  for each k in OPI((i,j), k)
    for each i
      for each j
        if OPI((i,j), k)
          if O-i is not in OS-k
            insert O-i with O-i,A-j to OS-k ;
          else add O-i,A-j to O-i ;

Step 2: Transform  DEVS models from  OS-k
  for each k in OS-k
    for each i in O-i
      create AM-i corresponding  to O-i ;
    construct CM-k from  AM-is;
```

Figure 3 Transformation of OPI to DEVS models

Note that the transformation of Figure 3 results in k independent DEVS models in which no interaction is occurred between them. Accordingly, simulation experiments are performance in a way that one experiment can measure only one performance index by a DEVS model, which is very inefficient. As shown in Figure 31 an experimental frame can be designed such that one simulation experiment can measure more than one performance indices. To do so requires an assembly of two models with a common component model between the two.

4.2. Assembly of Individual Performance Models

Figure 4 shows three OPI matrices in which different objects are involved to measure two performance indices. Figure 4 (a) shows that there are no common objects to measure performance PI 1 and PI 2. Figure 4 (b) shows that there is a common object, Object 4, to measure performance PI 1 and PI 2. Finally, Figure 4 (c) shows that all objects are contributed to measure performance PI 1 and PI 2. Figure 5 shows performance models which are transformed from the OPI matrices in Figure 4. Specifically, Figure 4 (a), (b), and (c) are transformed in Figure 5 (a), (b) and (c), respectively. Figure 5 (a) has two coupled models which has no common atomic model. In this case the two coupled models are used independently to measure different PIs. Figure 5 (b) has a model, M2, which is common to two coupled models, CM1 and CM2. In this case, an assembly of CM1 and CM2 can be made, which will be explained in the following. Figure 5 (c) is a case where every model contributes to all performance indices. This case may be considered as a special case of the case (a), that is, all atomic models are common to all coupled models. In this case, design of an experimental frame is very complex.



		P.I. 1	P.I. 2
Object 1	attr1-1	O	
	attr1-2	O	
	attr1-3		
Object 2	attr2-1	O	
	attr2-2	O	
Object 3	attr3-1		O
	attr3-2		O
Object 4	attr4-1		O
...	...	...	...

(a) Case 1

		P.I. 1	P.I. 2
Object 1	attr1-1	O	
	attr1-2	O	
	attr1-3		
Object 2	attr2-1	O	
	attr2-2	O	
Object 3	attr3-1		O
	attr3-2		O
Object 4	attr4-1	O	O
...	...	...	...

(b) Case 2

		P.I. 1	P.I. 2
Object 1	attr1-1	O	
	attr1-2	O	O
	attr1-3		O
Object 2	attr2-1	O	
	attr2-2	O	O
Object 3	attr3-1		O
	attr3-2	O	O
Object 4	attr4-1	O	O
...	...	...	...

(c) Case 3

Figure 4 Three Cases of OPI Matrix

Let us discuss the sharing of a component between two models and measure two PIs by one simulation experiment. Assume that CM-j and CM-k are coupled DEVS models which measure j and k performance indices, respectively. Assume also that an atomic DEVS AM-i is a common component in CM-j and CM-k. Recall that although AM-i is common to CM-j and CM-k a state set of AM-i in the two may be different. Thus, the first step is to merge the two AM-i, which is done by a union of two state sets and composition of associated transition functions. Then, a coupled model with two component models, CM-j and CM-k, can be constructed which can measure two PIs, PI-i nd PI-j. Note that the coupled model is not just an assembly of two coupled models, CM-j and CM-k; it is a reduced coupled model by means of sharing a common component AM-i between the two. Of course, careful design of an experimental frame is required to measure two PIs at the same time. Assembly of more than two coupled models may be much complex, which may be done by a series of pair-wise reductions explained here until no such common component is found.

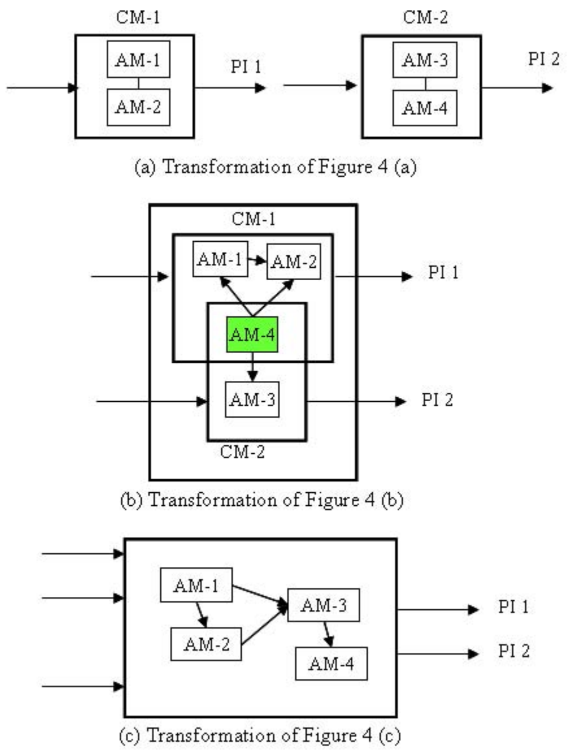

CM-1
AM-1
AM-2
PI 1
CM-2
AM-3
AM-4
PI 2
(a) Transformation of Figure 4 (a)
CM-1
AM-1
AM-2
AM-4
AM-3
CM-2
PI 1
PI 2
(b) Transformation of Figure 4 (b)
AM-1
AM-2
AM-3
AM-4
PI 1
PI 2
(c) Transformation of Figure 4 (c)

Figure 5 Transformation of OPI in Figure 4 into Models

4.3. Development of Overall Performance Model

Figure 6 shows an overall procedure for objective-driven performance simulation experiment using the proposed OPI matrix. The procedure starts with set up modeling and simulation objectives based on which performance indices are defined. To measure such indices model design along with experimental frame design is followed. Model design first develops an OPI matrix. Then the matrix is transformed into performance simulation models by means of the procedure shown in Figure 3, each of which measures associated performance indices. Once all models for all PIs are obtained assembly of such models would be performed. An assembled model may have one or more models which are common to two different models. On the other hand, experimental frame design includes development of models for data generation to performance models and for data collection from the models.

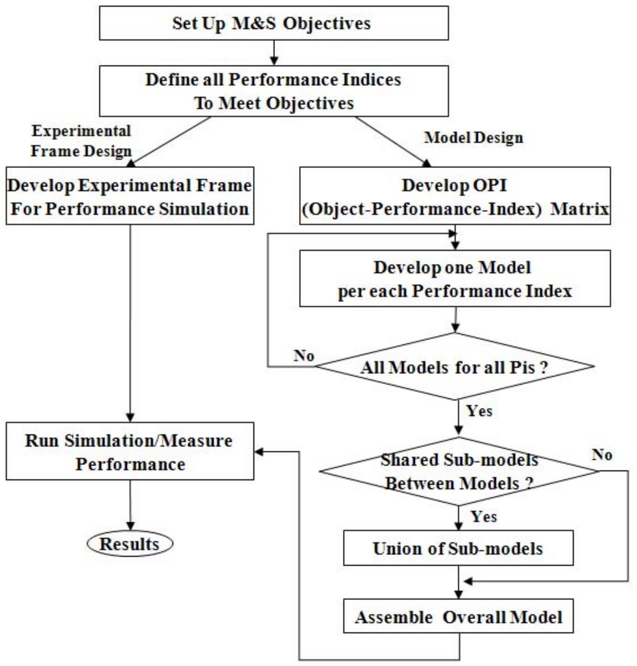
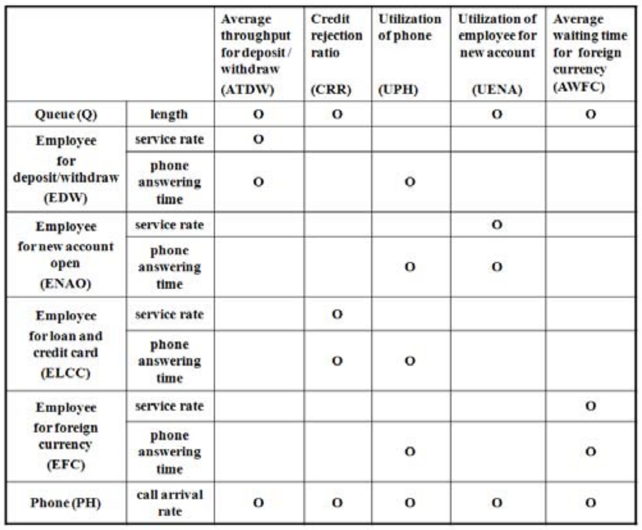

Set Up M&S Objectives
Define all Performance Indices To Meet Objectives
Experimental Frame Design
Model Design
Develop Experimental Frame For Performance Simulation
Develop OPI (Object-Performance-Index) Matrix
Develop one Model per each Performance Index
No
All Models for all Pis ?
Yes
Run Simulation/Measure Performance
Shared Sub-models Between Models ?
No
Yes
Results
Union of Sub-models
Assemble Overall Model

Figure 6 Procedure for Objective-driven M&S

5. EXAMPLE: BANK QUEUING SYSTEM

Consider a simplified bank queuing system in which customers are arrived and employees serve them. There are a queue and multiple employees each of which has his/her own special job. The jobs are: deposit/withdraw, open new account, load and credit card, and foreign currency. Assume that modeling objectives are set up and the corresponding OPI matrix is obtained as shown in Figure 7. As shown in the figure the dimension of the matrix is 6 x 10 x 5, meaning that 5 performance indices, 6 objects and 10 attributes are considered. From the figure it is clear that what objects are needed to compute a given performance index. For example, measurement of average throughput for deposit/withdraw needs three objects: Queue, Employee for deposit/withdraw and phone. Similarly, measurement of utilization of phone needs five objects: four employees and phone.



		Average throughput for deposit / withdraw (ATDW)	Credit rejection ratio (CRR)	Utilization of phone (UPH)	Utilization of employee for new account (UENA)	Average waiting time for foreign currency (AWFC)
Queue (Q)	length	O	O		O	O
Employee for deposit/withdraw (EDW)	service rate	O				
	phone answering time	O		O		
Employee for new account open (ENAO)	service rate				O	
	phone answering time			O	O	
Employee for loan and credit card (ELCC)	service rate		O			
	phone answering time		O	O		
Employee for foreign currency (EFC)	service rate					O
	phone answering time			O		O
Phone (PH)	call arrival rate	O	O	O	O	O

Figure 7 OPI Matrix for bank queuing system

Let us first design models and then an experimental frame design. For the model design apply the procedure explained in the previous section and construct an overall performance model in two steps. The first step is to identify a set of objects for each performance index. This can be done intuitively. Sets of all objects for each performance measure are as follows.

OS-ATDW = {Q, EDW, PH };
OS-CRR = { Q, ELCC, PH };
OS-UPH = { EDW, ENAO, ELCC, EFC, PH };
OS-UENA = {Q, ENAO, PH };
OS-AWFC = {Q, EFC, PH }.

The next step is to transform each OS set into DEVS models. For example, OS-ATDW is transformed into CM-ATDW whose components are AM-Q, AM-EDW and AM-PH, where CM- and AM- are a coupled DEVS and an atomic DEVS models, respectively. Other DEVS models are similarly obtained as below.

CM-ATDW = {AM-Q, AM-EDW, AM-PH};
CM-CRR = {AM-Q, AM-ELCC, AM-PH};
CM-UPH = {AM-EDW, AM-ENAO, AM-ELCC, AM-EFC, AM-PH};
CM-UENA = {AM-Q, AM-ENAO, AM-PH};
CM-AWFC = {AM-Q, AM-EFC, AM-PH}.

The final step is to assemble all coupled models above by union of each atomic model which is common in coupled models. The finally assembled coupled model in this example is the one in which all atomic models are included.

We now design an experimental frame to perform simulation experiments with models developed in the previous step. The frame includes models for generation of customers as well as for collection of simulation data. A generator model, GEN, should generator customers each of which has an attribute of a type of jobs to be performed in the bank. On the other hand, data collection can be done by multiple data collection models each of which measures performance index of its own. More specifically, DC-ATDW is a data collection model to measure the performance index ATDW and DC-CRR is for CRR, and so on. Figure 8 shows an overall model to measure five performance indices, which includes a generator and five data collectors.

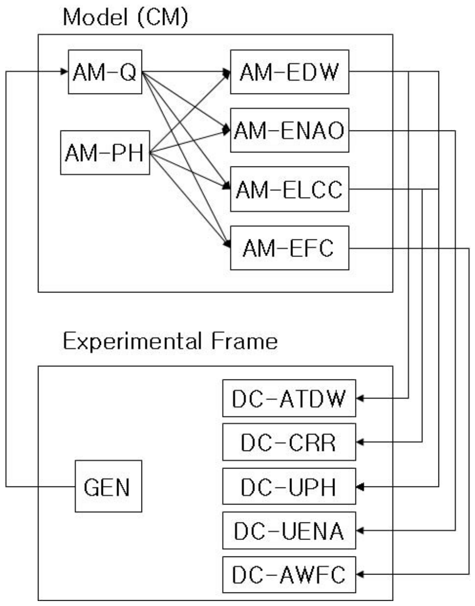


Figure 8 Performance Model for OPI in Figure 7

6. CONCLUSIONS

This paper proposed an engineering tool called an OPI (Object-Performance Index) matrix for realization of a methodology for objective-driven discrete event systems modeling. The matrix supports modeler to identify a set of objects which contribute to a given performance index. It also presented a procedure to transform the objects to a DEVS model which measure the performance index. Use of the OPI matrix for multiple objectives for M&S was also considered in which a set of performance models would be obtained and a possible assembly would be made. The proposed tool and the associated modeling methodology were successfully applied to a small size M&S project. Further work would develop an automated tool which generates DEVS models from a given OPI matrix.

References

[Ahn, 1993] M.S. Ahn and Tag G. Kim, "DEVS Methodology for Evaluating Timeconstrained Message Routing Policies", *Discrete Event Dynamic Systems*, Vol. 3, P. 173 - 192, 1993.

[Ahn, 1996] M.S.Ahn and Tag G. Kim, "Hybrid Modeling and Simulation of Discrete Event Systems", *Proc. Of Eurpean Simulation Symposium*", Serial. 8, Genoa, Italy, P. 271 - 276, 1996.

[Bolch, 2006] Gunter Bolch et al, *Queuing Networks and Morkov Chains: Modeling and Performance Evaluation with Computer Science Applications*, John Wiley & Sons, 2006.

[DEVS-STD 2005] http://www.devs-world.org.

[Kim, 1992] Tag G. Kim and Sung B. Park, "The DEVS Formalism:Hierarchical Modular Systems Specification in C++", *1992 European Simulation Multiconference*, York, United Kingdom, P. 152 - 156, 1992.

[Kim, 1995] Tag G. Kim, "The DEVS Formalism: Reusable Model Specification in an Object-oriented Framework", *International Journal in Computer Simulation*, Vol. 5, No. 4, P. 397 - 415, 1995

[Kiviat, 1973] P.J. Kiviat et. al, *SIMSCRIPT: A Simulation Programming Language*, CACI, 1973.

[Pritsker, 1984] A.A.B. Pritsker, *Introduction to Simulation and SLAM II.* (2nd Edition), Halsted Press, 1984.

[Zeigler, 1984] B.P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Academic Press, 1984.

[Zegiler, 2000] B.P. Zeigler, H. Praehofer and Tag G. Kim, *Theory of Modelling and Simulation* (2nd Edition), Academic Press, 2000.