Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. D. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

A Structured DEVS Model Representation Based on Extended Structured Modeling

Yunping Hu
Jun Xiao
Gang Rong

Institute of Cyber-Systems and Control
State Key Laboratory of Industrial Control Technology, Zhejiang University
Hangzhou, Zhejiang 310027, P.R.C

Xiaolin Hu

Department of Computer Science
Georgia State University
Atlanta, GA 30303, USA

ABSTRACT

Developing a simulation model needs lots of costs. If the model elements can be reused in newly developed models of the same physical system, the modeling costs will be reduced. Traditional DEVS model representations depend on programming languages. A modeler is difficult to identify the DEVS semantics of model elements, which limits the reuse of existing models. In this paper, the structured modeling technology is used to represent DEVS models. A DEVS model is represented as a structured model. An atomic model can be represented as a genus graph and a modular tree, and a coupled model can be represented as elemental detailed tables. Based on the visual representation, models can be stored, maintained and reused easily. Two cases for the application of structured DEVS model representation are also presented.

1 INTRODUCTION

Simulation modeling is a process of abstraction with consideration of some objective. Reuse of existing models can significantly reduce the modeling costs and improve the quality of simulation when developing new models for the same physical system.If a model can be represented as a structured format, where the model elements and the relations between elements can be represented visually, the model reusability will be improved significantly.

In this paper, we focus on the reuse of existing models in the Discrete Event System Specification (DEVS) formalism (Zeigler, Praehofer, and Kim 2000), which has been used to discrete event simulation for more than 30 years. Traditional DEVS implementations, like DEVSJava (Sarjoughian and Zeigler 1998) and CD++ (Wainer 2002), are mainly developed in the object-orientation programming (OOP) languages. An atomic DEVS model is usually represented by an OOP class, and the DEVS behaviors are modeled as methods of this class. The OOP model representations have better usability and can be executed by simulators implemented in the same OOP programming languages. However, it is difficult to identify the DEVS semantics of source codes in the programming environments, which influences the model reusability. From the structured view, these model representations are structured with elements of package, class, attribute and method in the programming aspect. They are not structured in the DEVS aspect because their elements for representation have no clear DEVS semantics.

Recently, some new DEVS modeling tools, which try to represent models in methods independent on sources codes, have been developed. In Component-based System Modeler (CoSMo), a system is modeled through three model types: template models, instance template models and instance models (Sarjoughian and Elamvazhuthi 2009). CoSMo visualizes the model structure as a tree, where each node corresponds to a port, a model or a component. AutoDEVS uses constrained natural language (NL) to define FDDEVS models and then elaborate models using source codes (Salas and Zeigler 2009). For better mode compositionality, AutoDEVS uses system entity structure (SES) to represent the model structure. However, SES cannot describe the model behavior. From the structured view, CoSMo and AutoDEVS both have a structured representation for the model structure, while they do not support a structured model behavior. Their representations of the model behavior depend on sources codes or are limited by FDDEVS.

In this paper, we use Extended Structured Modeling (ESM) to represent a DEVS model as a structured model. For the atomic DEVS formalism, a specific ESM model schema for a general atomic model is proposed. For the coupled DEVS formalism, a specific ESM model schema for a coupled model class is proposed. In these two model schemas, a set of ESM genera and modules with clear DEVS semantics are defined. This model representation not only has a structured model structure, but also has a structured model behavior. Modelers can manage the model elements visually and can modify or reuse models conveniently.

The remaining of this paper is structured as follows. In Section 2, we introduce ESM and the DEVS formalism as the background knowledge. Section 3 describes the structured DEVS model representation based on ESM in detail. Applications and benefits of this representation are shown in Section 4, and conclusions and future work are discussed in Section 5.

2 EXTENDED STRUCTURED MODELING

Structured Modeling (SM) is a kind of model representation based on discrete mathematics in the management science/operations research community (Geoffrion 1989). It uses a hierarchically organized, partitioned and attributed acyclic graph to represent a model. SM has five element types:

1. A primitive entity element is to represent any identifiable entity.
2. A compound entity element is a segmented tuple of primitive entity elements and/or other compound entity elements.
3. An attribute element, which may be constant or variable, is a segmented tuple of entity elements together with a value in some range.
4. A function element is a segmented tuple of elements together with a rule that calculate a particular value based on the attribute elements of the same tuple.
5. A test element is like a function element, except that its value is only true or false.

However, SM focuses on the representation of static models vis-a-vis dynamic models. ESM (Lenard 1992, Lenard 1993) extends SM for representing discrete event simulation (DES) models and proposes three new elements:

1. A random attribute element is to represent a random variable.
2. An action element is to represent a change in a model element.
3. A transaction element is a combination of action elements associated with a control structure.

The segmented tuple portion of an element is called its calling sequence. If one element appears in another calling sequence, the former is said to call the later. ESM has been used to DES models in the event-oriented world view. Unfortunately, ESM has not been used to represent DEVS models.

3 DEVS MODEL REPRESENTATION BASED ON ESM

A structured model has two parts: a model schema and a model instance. A model instance is composed of elements and is represented as elemental tables. The elements with similarity can be treated as a genus. A model schema is composed of genera and is represented as a genus graph, where the genera are connected by calling sequences. A model schema can also be represented as a modular structure tree, where the rooted node and the intermediate nodes are modules and the leaves are genera. The structured DEVS model representation is developed based on ESM and the DEVS formalism, as shown in Figure 1. In this version, we focus on the Classic DEVS formalism.

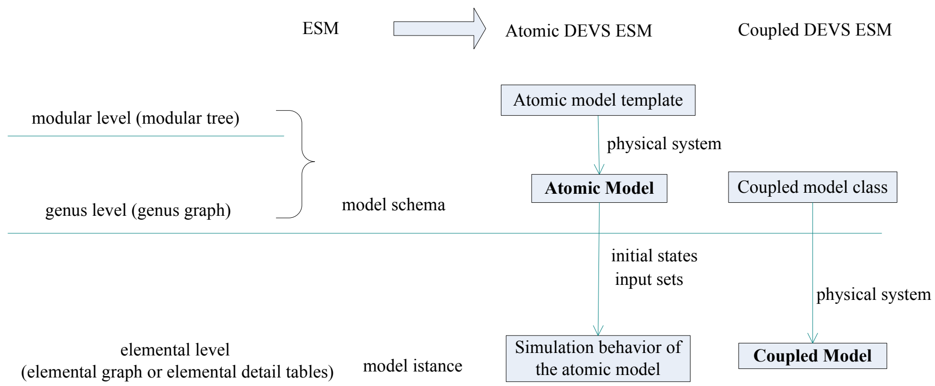

Figure 1: The framework of the DEVS representation based on ESM

An atomic model uses a serious of variables (input or output variable, state variable) and rules (state transition function, output function) to represent a physical system. The simulation process of an atomic model describes the physical system completely in a time period. This process of a model is different, if the initial states or input events are different. Therefore, in the ESM view, various simulation initial states, input sets and results of an atomic model are treated as model instances, while the atomic model is treated as a model schema.

Different atomic models in the DEVS formalism have some common features. In this paper, we develop an atomic model template which contains common genera for representing various atomic models. An atomic model of some physical system can be represented according to the template. The model instance of an atomic model is to represent the event-driven simulation behavior.

A coupled DEVS model has no model behavior. In the genus and modular level, various coupled model shares the same model schema. Using ESM, we develop a specific model schema for representing the coupled model class. A coupled model of some physical system is only an instance of this model schema.

In the following, for describing the model schemas, three formats are used: Structuring Modeling Language (SML), genus graph and modular tree. SML is a kind of textual modeling language to represent a model schema in detail(Geoffrion 1992a, Geoffrion 1992b). The model instances are represented as elemental tables.

3.1 Atomic DEVS Model representation based on ESM

For representing the atomic DEVS model template, the SML schema is show in Figure 2. It explains the modules and genera of the atomic DEVS model template in detail.

The simulation process of an atomic DEVS model is event-driven. For describing the dynamic behavior, an EVENT primitive set is defined. Each event occurs in some time point and may cause the update of state variables. So each EVENT element has attributes of TIME VALUE, CURRENT STATE, CURRENT PHASE, CURRENT TIME ADVANCE, NEXT STATE, NEXT PHASE and NEXT TIME ADVANCE. The phase variable and the time advance variable are inherent state variables in each atomic model.

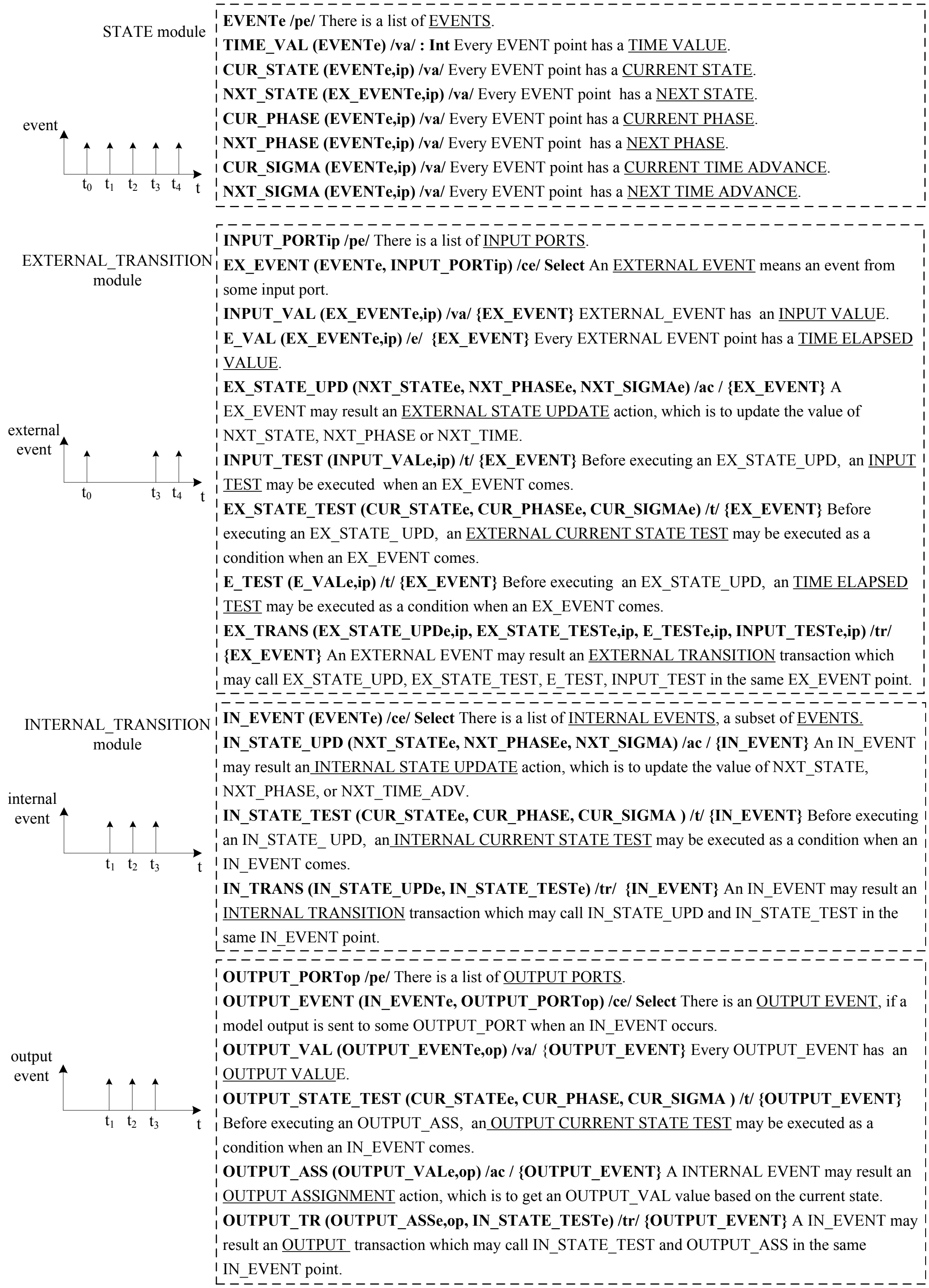

Figure 2: SML schema for the atomic DEVS model template

All events in a simulation are divided into external events and internal events. When an input comes into a model from outside, an external event occurs and the external transition function is called to update the state variables based on the present state, the input value and the time elapsed in the current state. Therefore, we define an EXTERNAL EVENT set, which is a compound set combining the EVENT set with the INPUT PORT set. Each EXTERNAL EVENT has the attributes of INPUT VALUE and TIME ELAPSED VALUE, actions of EXTERNAL STATE UPDATE and tests of INPUT TEST, EXTERNAL CURRENT STATE TEST and TIME ELAPSED TEST. The EXTERANL TRANSITION transaction calls these actions and tests when an external event occurs.

When the elapsed time equals to the time advance in the current state, an internal event occurs and the internal transition function is called to update the state variables based on the present state. We define an INTERNAL EVENT set, a subset of the EVENT set. An INTERNAL EVENT may have INTERNAL STATE UPDATE actions and the INTERNAL CURRENT STATE TESTs. The INTERANL TRANSITION transaction calls these actions and tests when an internal event occurs.

When an internal event occurs, the output function will be called to send out message before the internal transition function is called. So, each internal event is also an output event which associates with an output port. We define an OUTPUT EVENT set which is a compound set combining the INTERNAL EVENT set with the OUTPUT PORT set. Each OUTPUT EVENT has an attribute of OUPUT VALUE and an OUTPUT ASSIGNMENT action and may have an OUTPUT CURRENT STATE TEST. The OUTPUT TRANSACTION calls the action and the test when an output event occurs.

In classic ESM, each attribute has a value range in consideration of the mathematical aspect. However, each attribute here has a simple or complex data type. Because the DEVS implementations depend on programming languages, the data type property is necessary for an attribute.

For visualizing the atomic DEVS model template, the genus graph is shown in Figure 3. and the modular tree is shown in Figure 4. In this genus graph, the genera are connected as a directed graph through calling sequences. In this modular tree, the rooted node is the ATOMIC_MODEL module and the intermediate notes have four module types: STATE, EXTERNAL_TRANSITION, INTERNAL_TRANSITION and OUPUT.

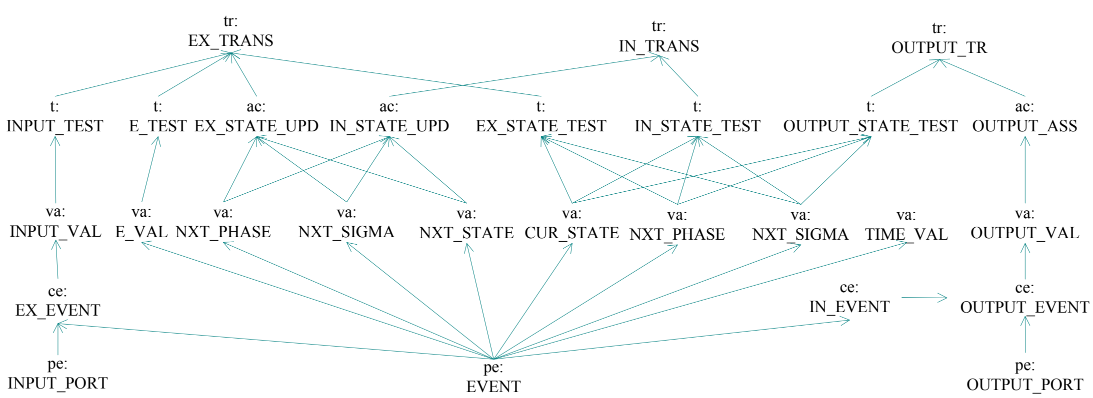

Figure 3: Directed genus graph for the atomic DEVS model template

Furthermore, the modular tree here explains how to represent an atomic model of some physical system based on the model template. A specific atomic model may have other state variable in addition to the phase variable and the time advance variable. So in the STATE module, specific CUR_STATE attributes and NXT_STATE attributes may be added. Each EXTERNAL_TRANSITION module corresponds to an input port. The reason is that an atomic model may have different rules (actions and transactions) to change the state for different input ports. If an atomic model has more than one input ports, more than one EXTERNAL_TRANSITION modules are needed. The OUTPUT module is similar with the EXTERNAL_TRANSITION module and each OUTPUT module corresponds to only one output port.

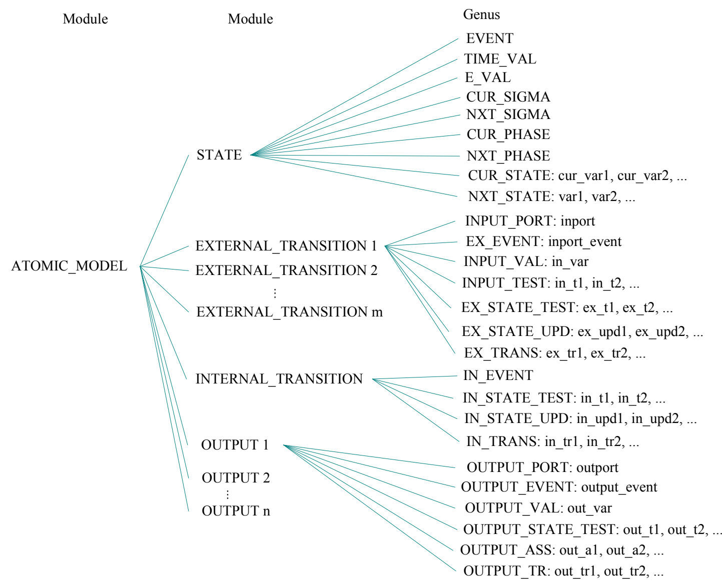

Figure 4: Modular tree for the atomic DEVS model template

A model instance of the model schema for the atomic model template can be represented as elemental detail tables, as show in Table 1. These tables only contain entities and value-based elements including primitive entities, compound entities, attributes and tests. The column names of these tables come from the model schema for the atomic model template. For an atomic model of some physical system, these elemental tables may be extended in terms of the extension of the model template.

Table 1: Elemental detail tables for the model instance of the atomic DEVS model template

Table Name	Column Name
EVENT	EVENT ‖ INTERP TIME_VAL CUR_STATE NXT_STATE CUR_PHASE NXT_PHASE CUR_SIGMA NXT_SIGMA E_VAL
INPUT_PORT	INPUT_PORT ‖ INTERP
EX_EVENT	EVENT INPUT_PORT ‖ IN_VAL E_TEST INPUT_TEST EX_STATE_TEST
IN_EVENT	EVENT ‖ IN_STATE_TEST
OUT_PORT	OUTPUT_PORT ‖ INTERP
OUTPUT_EVENT	EVENT OUTPUT_PORT ‖ OUT_VAL OUTPUT_STATE_TEST

3.2 Coupled DEVS model representation based on ESM

A coupled DEVS model only has elements of model structure. We define a series of primitive sets and compound sets in the model schema for the coupled DEVS model class. The SML representation of the coupled model class is show in Figure 5.

A coupled model is composed of components, each of which associates with an atomic or coupled model. So, we define two primitive sets of MODEL and COMPONENT and a compound MODEL COMPONENT set. The ATOMIC MODEL set and the COUPLED MODEL set are both compound sets, subsets of the MODEL set. The components in a coupled model are connected by couplings: internal couplings (ICs), external input couplings (EICs) and external output couplings (EOCs). Therefore, we define three compound sets of IC, EIC and EOC. The select function can be represented as a set of ordered pairs, each element in which has one prioritized component and one or more than one concurrent components. So, we define a SELECT compound set which has two member COMPONET sets.

MODELm /pe/ There is a list of MODELs.
C_MODEL (MODELm) /ce/ Select There is a list of COUPLED MODELs, a subset of MODELS.
A_MODEL (MODELm) /ce / Select There is a list of ATOMIC MODELs, a subset of MODELS.
MODEL_INPUT_PORT (MODELm, INPUT_PORTip) /ce/ Select {MODEL} × {INPUT_PORT} A MODEL INPUT PORT means that a MODEL has zero or more INPUT_PORTs.
MODEL_OUTPUT_PORT (MODELm, OUTPUT_PORTop) /ce/ Select {MODEL} × {OUTPUT_PORT} A MODEL OUTPUT PORT means that a MODEL has zero or more OUTPUT_PORTs.
COMPONENT /pe/ There is a list of COMPONENTs.
MODEL_COMPONENT /ce/ Select {MODEL} × {COMPONENT} A MODEL COMPONENT means that a COMPONENT associates with a MODEL.
EOC (C_MODELm, OUTPUT_PORTop, COMPONENTc, OUTPUT_PORTop) /ce/ Select {C_MODEL} × {OUTPUT_PORT} × {COMPONENT} × {OUTPUT_PORT} A EXTERNAL OUTPUT COUPLING represents a output coupling between a COMPONENT and a COUPLED_MODEL.
EIC (C_MODELm, INPUT_PORTip, COMPONENTc, INPUT_PORTip) /ce / Select {C_MODEL} × {INPUT_PORT} × {COMPONENT} × {INPUT_PORT} A EXTTERNAL INPUT COUPLING represents a input coupling between a COMPONENT and a COUPLED_MODEL.
IC (COMPONENTc, OUTPUT_PORTip, COMPONENTc, INPUT_PORTop) /ce/ Select {COMPONENT} × {OUTPUT_PORT} × {COMPONENT} × {INPUT_PORT} A INTTERNAL COUPLING represents a internal coupling between a COMPONENT and another COMPONENT.
SEL (COMPONENTc, COMPONENTc) /ce/ Select A SELECT FUNCTION has a set of ordered pairs where the first elements are prioritized components and the second elements are concurrent components.

Figure 5: SML schema for the coupled DEVS model class

Based on the SML schema for the coupled DEVS model class, the genus graph is shown in Figure 6 and the modular tree is shown in Figure 7. The rooted node of the modular tree is COUPLED_MODEL, which has three intermediate nodes: MODEL, COMPONENT and COUPLING. The elemental detail tables for a coupled model are shown in Table 2. Modelers can write the coupled model elements for some physical system into these tables.

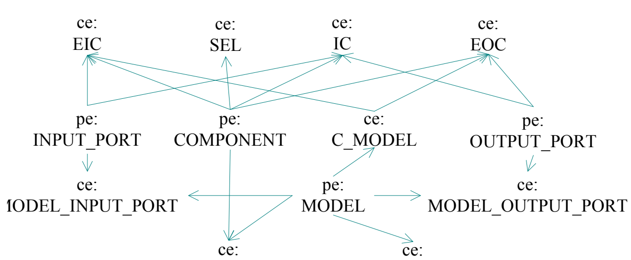

Figure 6: Directed genus graph for the coupled DEVS model class

4 CASES

Based on the structured DEVS model representation, we can represent specific DEVS models. In this section, we use two cases to test the model representation and discuss the benefits using this method.

4.1 The Structured Representation of an Atomic Model

We have defined the atomic model template based on ESM. In this section, we use the template to represent the processor model. The formal specification of the processor model is shown in Figure 8

The modular tree and the genus graph for the processor model is shown in Figure 9. There is a state variable named job in the processor model. Therefore, in the STATE module, specific CURRENT

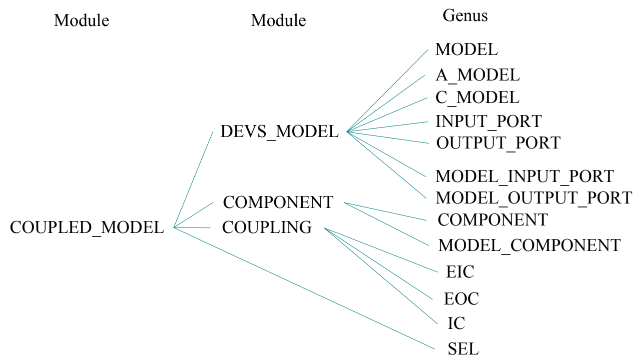

Figure 7: Modular tree for the coupled DEVS model class

Table 2: Elemental detail tables for the model instance of the atomic DEVS model template

Table Name	Column Name
MODEL	MODEL ∥ INTERP
A_MODEL	MODEL
C_MODEL	MODEL
INPUT_PORT	INPUT_PORT ∥ INTERP
MODEL_INPUT_PORT	MODEL INPUT_PORT
OUTPUT_PORT	OUTPUT_PORT ∥ INTERP
MODEL_OUTPUT_PORT	MODEL OUTPUT_PORT
COMPONENT	COMPONENT ∥ INTERP
EIC	MODEL INPUT_PORT COMPONENT INPUT_PORT
EOC	COMPONENT OUTPUT_PORT MODEL OUTPUT_PORT
IC	COMPONENT OUTPUT_PORT COMPONENT INPUT_PORT
SEL	COMPONENT COMPONENT

Processor = $<X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta>$
Where
InPorts = {“in” }, where $X_{in} = R_0^+$
X = $\{(p,v)|p \in InPorts, v \in X_p\}$ is the set of input ports and values
OutPorts = {“out” }, where $Y_{out} = R_0^+$
Y = $\{(p,v)|p \in OutPorts, v \in Y_p\}$ is the set of Output ports and values
S = {phase, sigma, job} = {“passive”, “active”}$\times R_0^+ \times R_0^+$
δ **ext(msg: type ExternalMessage)** {
cur_job = job; cur_phase = phase; cur_sigma = sigma;
if(msg.port== in) { inJob = msg.value; job = inJob;
if(cur_phase == passive) { phase = active; sigma = 5;}
else sigma = cur_sigma – e; }
δ **int(msg: type InternalMessage)** {
cur_phase = phase; cur_sigma = sigma;
phase = passive; sigma = Inf; }
λ **(msg: type InternalMessage)** {
cur_job = job;
outJob = cur_job; send outJob to out port; }
ta(“passive”) = Inf **ta(“active”) = 5**

Figure 8: The formal specification of the processor model

STATE named cur_job and specific NEXT STATE named job are defined. Because there is only one input port and one output port, there is one EXTERNAL_TRANSITION module and one OUTPUT module in the modular tree. In the EXTERNAL_TRANSITION module, the tr1 transaction genus calls the phase_t test, the job_ac1 action, the sigma_ac1 action and the sigma_ac3 action with the if-else control structure. There is no transaction in the OUTPUT module because there is only one action named outJob_ass. In the INTERNAL_TRANSITION module, the tr2 transaction calls the sigma_ac2 action and the phase_ac2 action with the sequential control structure. After simulation, we can get the model instance for the processor

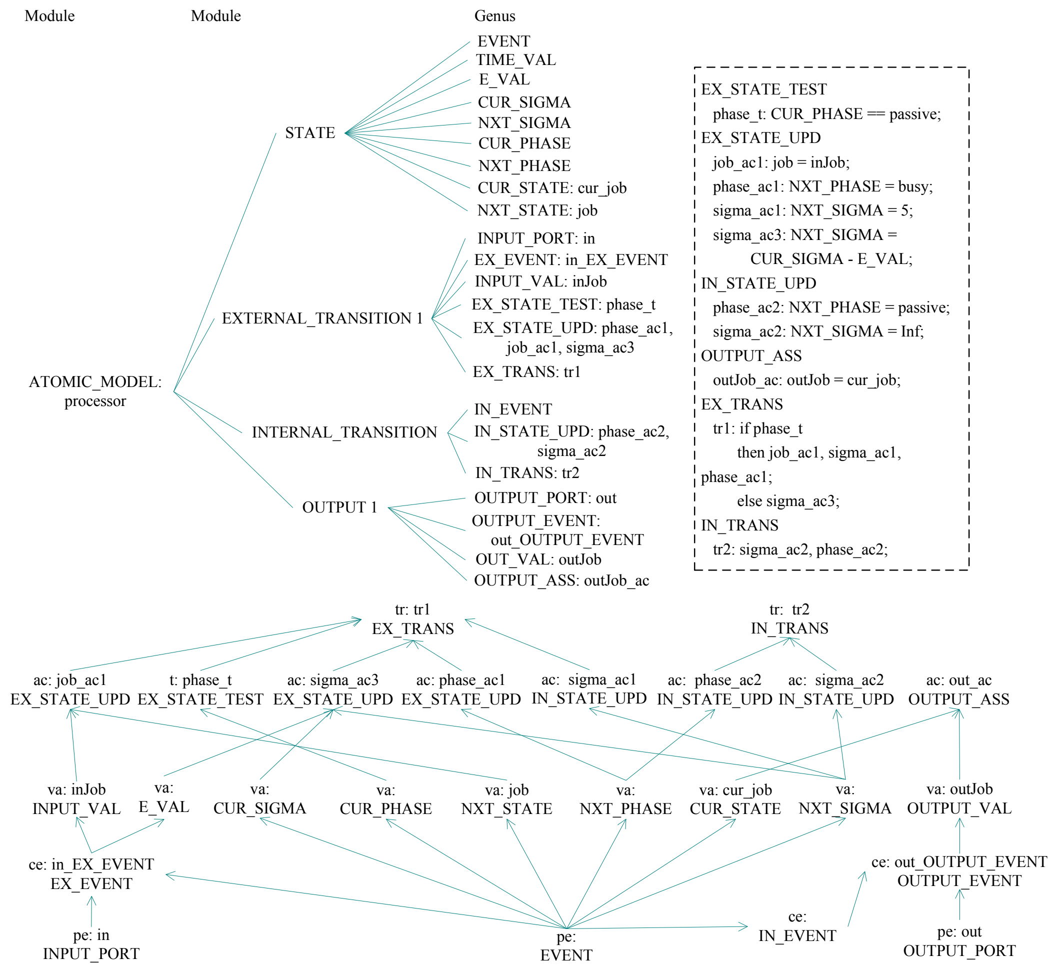

Figure 9: Modular tree for the coupled DEVS model class

model. The initial states and input sets have been set before simulation. The initial phase is passive, the initial time advance is infinite and the initial value of the job variable is 0. There are three input events in the time points of 1 and 4. The elemental detail tables of this model instance are shown in Figure 10.

There are several benefits for the structured representation of DEVS models. The modular tree and genus graph can represent an atomic model in a visual format. If the modeler does not use some variable in a newly developed model, the variable can be deleted and the actions and transaction calling the variable will be deleted automatically in the genus graph. The model instance shows the simulation results in detail. The modeler can choose corresponding tables to look at the state updates, input events or output events.

Table Name: EVENT

EVENT	||	INTERP	TIME_VAL	CUR_PHASE	NXT_PHASE	CUR_SIGMA	E_VAL	NXT_SIGMA	cur_job	job
e1	||	in job 1	1	passive	active	Inf	1	5	0	12
e2	||	in job 2	4	active	active	5	3	2	12	12
e3	||	out job 1	6	active	passive	2	0	Inf	12	12

Table Name: INPUT_PORT

INPUT_PORT	||	INTERP
in	||	in job port

Table Name: OUTPUT_PORT

OUTPUT_PORT	||	INTERP
out	||	out job port

Table Name: IN_EVENT

EVENT
e3

Table Name: in_EX_EVENT

EVENT	INPUT_PORT	||	inJob	phase_t
e1	in	||	12	true
e2	in	||	17	false

Table Name: out_OUTPUT_EVENT

EVENT	OUTPUT_PORT	||	outJob
e3	out	||	12

Figure 10: Elemental detail tables for the processor model

4.2 The Structured Representation of an coupled model

We have defined the coupled model class based on ESM. In this section, we use the model class to represent the EFP (Experiment Frame and Processor) model, as shown in Figure 11. The EFP model is a coupled DEVS model and has three components: Generator, Processor and Transducer. Each component associates with an atomic DEVS model. For example, Generator is a component of the genr model.

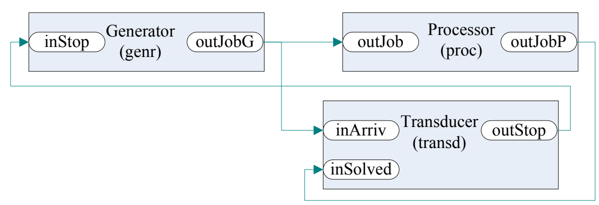

Figure 11: EFP coupled model

The EFP coupled model is represented as a model instance of the coupled model class. The elemental detail tables for this model instance are shown in Figure 12. The model elements of the EFP model are all listed in these tables.

5 CONCLUSION

In this paper, we use the ESM to represent a DEVS model as a structured model. For the atomic DEVS formalism, we propose an atomic DEVS model template represented as an ESM model schema. This template contains basic modules and genera for the representation of various specific atomic models. The model instance for an atomic model is to represent the simulation behavior in some initial states and input sets. For the coupled DEVS formalism, we propose a coupled model class represented as an ESM model schema. A specific coupled model is a model instance of this model schema.

The structured representation of DEVS models provides a visual format to build and maintain models. An atomic model can be represented as a modular tree or a genus graph. The modeler can add or delete model elements visually. Furthermore, the simulation behavior of an atomic model is represented as elemental detail tables in detail. The user can observe the simulation results conveniently. The model elements of a coupled model are stored in a series of tables, where they can be edited clearly.

We are planning to develop a DEVS modeling platform to support the ESM model representation. A DEVS model is firstly represented as an ESM model, and then is transformed into a model executed by CD++ or DEVSJava. The DEVS models are created and maintained in a structured format in this platform. The Java language and the data base technology are used in developing this platform.

Table Name: MODEL

MODEL	INTERP
genr	a generator atomic model
proc	a processor atomic model
transd	a transducer atomic model

Table Name: ATOMIC_MODEL

MODEL
genr
proc
transd

Table Name: INPUT_PORT

INPUT_PORT	INTERP
inStop	stop job port
inJob	in job port
inArriv	arrive job port
inSolved	solved message port

Table Name: OUTPUT_PORT

OUTPUT_PORT	INTERP
outJobG	out job port
outJobP	out job port
outStop	stop message port

Table Name: COMPONENT

COMPONENT	INTERP
Generator	a genr component
Processor	a proc component
Transducer	a transd component

Table Name: MODEL_INPUT_PORT

MODEL	INPUT_PORT
genr	inStop
proc	inJob
transd	inArriv
transd	inSolved

Table Name: MODEL_OUTPUT_PORT

MODEL	OUTPUT_PORT
genr	outJobG
proc	outJobP
transd	outStop

Table Name: MODEL_COMPONENT

MODEL	COMPONENT
genr	Generator
proc	Processor
transd	Transducer

Table Name: IC

COMPONENT	OUTPUT_PORT	COMPONENT	INPUT_PORT
Generator	outJobG	Processor	inJob
Generator	outJobG	Transducer	inArriv
Processor	outJobP	Transducer	inSolved
Transducer	outStop	Generator	inJob

Table Name: SEL

COMPONENT	COMPONENT
Generator	Processor
Generator	Transducer
Processor	Transducer
Transducer	Generator

Figure 12: Elemental detail tables for the EFP model

ACKNOWLEDGEMENTS

The financial supports from the National High Technology R&D programs of China (2012BAE05B03 & 2013AA040701) are gratefully acknowledged.

REFERENCES

Geoffrion, A. 1989. "The formal aspects of structured modeling". *Operations Research* 37 (1): 30–51.

Geoffrion, A. 1992a. "The SML language for structured modeling: Levels 1 and 2". *Operations Research* 40 (1): 38–57.

Geoffrion, A. 1992b. "The SML language for structured modeling: Levels 3 and 4". *Operations Research* 40 (1): 58–75.

Lenard, M. L. 1992. "Extending the structured modeling framework for discrete-event simulation". In *Proceedings of the Twenty-Fifth Hawaii International Conference*, 494–503. IEEE.

Lenard, M. L. 1993. "A prototype implementation of a model management system for discrete-event simulation models". In *Proceedings of the 25th conference on Winter simulation*, 560–568. ACM.

Salas, M. C., and B. P. Zeigler. 2009. "AutoDEVS: A Methodology for Automating Modeling and Simulation Software Development and Testing of Interoperable Systems". *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology* 6 (1): 33–52.

Sarjoughian, H. S., and V. Elamvazhuthi. 2009. "CoSMoS: a visual environment for component-based modeling, experimental design, and simulation". In *Proceedings of the 2nd international conference on simulation tools and techniques*. ICST.

Sarjoughian, H. S., and B. P. Zeigler. 1998. "DEVSJAVA: Basis for a DEVS-based collaborative M&S environment". In *Proceedings of the International Conference on Web-based Modeling & Simulation*, 29–36.

Wainer, G. 2002. "CD++: a toolkit to develop DEVS models". *Software - Practice and Experience* 32:1261–1306.

Zeigler, B. P., H. Praehofer, and T. G. Kim. 2000. *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*. Academic press.

AUTHOR BIOGRAPHIES

Yunping Hu is a doctor candidate of Control Science and Engineering at Zhejiang University in Hangzhou, China. He holds a M.S. in Control Science and Engineering from Nanjing University of Technology in Nanjing, China. His research interests include modeling & simulation, model management and decision support system. His email address is yphu@iipc.zju.edu.cn.

Jun Xiao is a master candidate of Control Science and Engineering at Zhejiang University in Hangzhou, China. He earned his Bachelor's degree in Automation from Wuhan University in Wuhan, China. His research interests include DEVS modeling & simulation and mathematical knowledge management. His email address is jxiao@iipc.zju.edu.cn.

Xiaolin Hu is an Associate Professor in the Computer Science Department at Georgia State University, Atlanta, Georgia. He received his PhD degree from the University of Arizona, MS degree from the Chinese Academy of Sciences, and BS degree from the Beijing Institute of Technology in 2004, 1999, and 1996, respectively. His research interests include modeling and simulation theory and application, agent and multi-agent systems, and complex systems science. He has served as program chairs for several international conferences/symposiums in the field of modeling and simulation, and is an associate editor for Simulation: Transaction of The Society for Modeling and Simulation International. Dr Hu is a National Science Foundation (NSF) CAREER Award recipient. His email address is xhu@cs.gsu.edu.

Gang Rong is Professor of Control Science and Engineering at Zhejiang University in Hangzhou, China. His research interests cover modeling, simulation & optimization, data-ming, data visualization and enterprise-control system integration in process industries. He holds a Ph.D. in Control Science and Engineering from Zhejiang University in Hangzhou, China. His email address is grong@iipc.zju.edu.cn and his web page is http://mypage.zju.edu.cn/en/rglab.