DEVS FRAMEWORK INSTRUMENTED WITH DATABASE FOR WEB-BASED WORKFLOW MODELING SIMULATION

Ki Jung Hong, Jong-Kwon Lee, Dae Hyun Kim, and Tag Gon Kim
Systems Modeling Simulation Lab, Dept of EE, KAIST
373-1 Kusong-dong, Yusong-gu,
Taejon 305-701, KOREA
{kjhong,jklee,dhkim,tkim}@coregate.kaist.ac.kr

KEYWORDS: DEVS formalism, Relational algebra, ER Modeling, Workflow Modeling

ABSTRACT

This paper deals with a web-based workflow modeling simulation environment within the DEVS framework under development at SMSL (Systems Modeling and Simulation Lab) of KAIST. We propose the Relational Algebraic DEVS Interchange format (RADESIF) which can easily be exchanged to other workflow definitions. Definition of RADESIF employs a relational algebraic framework to translate DEVS semantics into relational algebra mathematically. RADESIF is converted into Java code which runs within the DEVSim-Java simulation environment. Each of the tools along with the overall architecture for the developed environment will be described. An example of web-based workflow modeling and simulation within the environment will be given.

1 INTRODUCTION

Every business office has procedures that are to be followed with the purpose of providing increased efficiency, consistency, and quality. The concepts of process engineering and workflows embody the ideas of controlling and coordinating complex activities and interactions among individuals and software components.

Workflow is concerned with automation of the procedures where documents, information or tasks are passed between participants according to a predefined set of rules to achieve, or contribute to, an overall business goal. Workflow Management Coalition (WfMC) formally defined workflow as the computerized facilitation or automation of a business process, in whole or part [WFMC95]. WfMC also defined Workflow Management System (WFMS) as a system that completely defines, manages and executes workflows through the execution of software whose order of execution is driven by a computer representation of the workflow logic.

For the workflow management system, research in the various software technologies is required. Among them, we are especially interested in modeling, simulation, and analysis of workflows [CHRW98].

Workflow process is considered as a complex discrete event system, which should represent dynamically changing business requirements. Simulation may be the only means to analyze such a complex process in realistic manner. Thus, a workflow management system should support modeling, simulation and analysis of such workflow processes.

Modeling of complex workflow process, as a discrete event system, requires a formal modeling framework, which supports modularity, ease of abstraction, and reusability. The DEVS (Discrete Event Systems Specification) formalism is one such framework, which provides a sound semantics for hierarchical, modular specification of complex discrete event models. So far, researchers have realized the formalism in different programming languages and environments. DEVSim++(C++ realization) and DEVSim-Java(Java realization) are such realizations developed by at SMSL(Systems Modeling and Simulation Lab) of KAIST.

In this paper, we propose the Relational Algebraic DEVS Intermediate Format (RADESIF) for the DEVS-based workflow execution model. Since RADESIF is a relational algebraic model, we can exploit existing relational database technologies for

DEVS modeling and simulation.

Section 2 describes a brief review of the DEVS formalism and the overall architecture for the developed environment. Section 3 explains the transformation of DEVS formalism in relational algebra. An example of web-based workflow modeling and simulation within the developed environment will be given in Section 4. Section 5 concludes this paper.

2 WORKFLOW MODELING SIMULATION AND ANALYSIS

There are many workflow examples from different application areas, e.g., software development process, mortgage request handling in an investment company, manufacturing control, lease contract management, and so on. Each such process is considered as a complex discrete event system. Simulation may be the only means to analyze such a complex process in realistic manner. This section first introduces the DEVS (Discrete Event Systems Specification) formalism [Zeigler84], which supports modeling of complex discrete event systems in a hierarchical, modular manner. It then describes a developed modeling simulation environment for a workflow applications.

2.1 DEVS Formalism: A Brief Introduction

The DEVS formalism specifies a model in a hierarchical, modular form. A discrete event system consists of entities whose dynamics are described as a set of procedure rules. Such rules control the interactions among the communicating entities. The communicating entities and the procedure rules can be decomposed into the smaller ones with the modeling semantics. These decomposed components are directly mapped to the atomic models, from which larger ones are built. A basic model, called an atomic model, is not further decomposed specification of the dynamic behavior of a component. Formally, an atomic model AM is specified as [Zeigler84]:

$$\begin{array}{rcl}
AM & = & < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta > \\
X & : & \text{input events set;} \\
S & : & \text{sequential states set;} \\
Y & : & \text{output events set;} \\
\delta_{int} & : & S \rightarrow S : \text{internal transition function;} \\
\delta_{ext} & : & Q \times X \rightarrow S : \text{external transition function;} \\
\lambda & : & S \rightarrow Y : \text{output function;} \\
ta & : & S \rightarrow Real : \text{time advance function ;} \\
Q & = & \{(s,e) | s \in S, 0 \leq e \leq ta(s)\} \\
 & : & \text{total state of } AM \ (e : \text{elapsed time}).
\end{array}$$

The second form of the DEVS model, called a coupled model(or coupled DEVS), is a specification of the hierarchical model structure. It describes how to couple component models together to form a new model. This new model can be employed as another component in a larger coupled model, thereby giving rise to the construction of complex models in a hierarchical fashion. Formally, a coupled model CM is defined as [Zeigler84]:

$$\begin{array}{rcl}
CM & = & < X, Y, \{M_i\}, EIC, EOC, IC, SELECT > \\
X & : & \text{input events set;} \\
Y & : & \text{output events set;} \\
\{M_i\} & : & \text{DEVS components set;} \\
EIC & \subseteq & X \times \cup_i X_i \\
 & : & \text{external input coupling relation;} \\
EOC & \subseteq & \cup_i Y_i \times Y \\
 & : & \text{external output coupling relation;} \\
IC & \subseteq & \cup_i Y_i \times \cup_i X_i \\
 & : & \text{internal coupling relation;} \\
SELECT & : & 2^{\{M_i\}} - \phi \rightarrow \{M_i\} : \text{tie breaking selector.}
\end{array}$$

A detailed discussion about the DEVS formalism and modeling is found in [Zeigler84].

2.2 Workflow Modeling Simulation Environment

The model of a process should represent what activities, dependencies among activities, and roles are necessary to the business process. Simulation-based analysis of a process model requires the construction of an executable model that can be used for the purpose of experiments carried out upon this model. A process definition describes only the structural aspects of organizational functions. Thus, the detailed operation behavior should be described in the workflow execution model for simulation and execution.

Figure 1 shows an overall workflow system architecture for the developed environment, including a modeling simulation environment and a real-time execution engine. We are focusing on the modeling simulation environment in the figure. The workflow system starts from a process definition and then

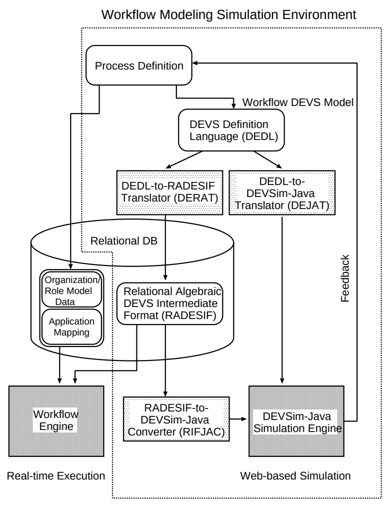

Workflow Modeling Simulation Environment
Process Definition
Workflow DEVS Model
DEVS Definition Language (DEDL)
DEDL-to-RADESIF Translator (DERAT)
DEDL-to-DEVSim-Java Translator (DEJAT)
Relational DB
Organization Role Model Data
Application Mapping
Relational Algebraic DEVS Intermediate Format (RADESIF)
Feedback
Workflow Engine
RADESIF-to-DEVSim-Java Converter (RIFJAC)
DEVSim-Java Simulation Engine
Real-time Execution
Web-based Simulation

Figure 1: Workflow Simulation Environment

constructs a Workflow DEVS Model as a workflow execution model. The Workflow DEVS Model is described by DEDL (DEVS Definition Language). DEDL is executed for the simulation engine as well as for the workflow engine. The workflow engine provides a real-time execution of the Workflow DEVS Model.

DEDL can be directly translated into DEVSim-Java code for the purpose of simulation. However, facilitation of an identical model in both simulation and real-time execution requires a more versatile specification format. Moreover, it is recommended to satisfy requirements for the interoperability between different definition tools or simulation engines. For these objectives, we have defined the Relational Algebraic DEVS Intermediate Format (RADESIF) by transformation of the DEVS semantics to relational algebra. The resulting RADESIF can easily be exchanged to other workflow definitions. Thus, a RADESIF is a DEVS model in relational algebraic semantics. Since RADESIF is a relational algebraic model, we can exploit existing relational database technologies for DEVS modeling and simulation. Moreover, since DEVSim-Java is based on Java, database accesses using the standard JDBC within DEVSim-Java requires no additional interface for any computer platform.

The developed web-based workflow modeling and simulation environment consists of DEVSim-Java along with supporting tools for automatic semantics translation and format conversion. The tools include DEDL (DEVS Definition Language), DERAT (DEDL-to-RADESIF Translator), DEJAT (DEDL-to-DEVSim-Java Translator), and RIFJAC (RADESIF-to-DEVSim-Java Converter).

Section 4 describes an example modeling simulation procedures of a workflow process within the environment of Figure 1.

3 DEVS FORMALISM IN RELATIONAL ALGEBRA

This section describes a transformation of the DEVS formalism to relational algebra. It first summarizes relational algebra briefly, and then presents the transformation procedure.

3.1 Relational Algebra

There are six fundamental opertaions that serve to define relational algebra [SKS97]. Let R, S be two relations over sets of attributes R', S', respectively. Let $X \subseteq R'$
(1) Rename ρ : Given a relational-algebra expression E, the expression ρ_x(E) returns the result of expression E under the name x.
(2) Selection σ : $\sigma_F(R) = \{t | F(t) \vee t \in R\}$
(3) Projection Π : $\Pi_X(R) = \{t[X] | t \in R\}$
(4) Union $\cup$: $R \cup S = \{t | t \in S \vee t \in S\}$
(5) Difference $-$: $R - S = \{t | t \in R \wedge t \notin S\}$
(6) Cartesian product $\times$: $R \times S = \{< r, s > | r \in R \wedge s \in S\}$
(7) Natural join $\bowtie$: $R \bowtie S = \Pi_{R' \cup S'}(\sigma_{R.A_1 = S.A_1 \wedge R.A_2 = S.A_2 \wedge \cdots \wedge R.A_n = S.A_n} R \times S)$ where $R' \cap S' = \{A_1, A_2, ..., A_n\}$

3.2 Transformation of DEVS model to RA model

3.2.1 RA model definition

A hierarchical, modular DEVS model can be expressed in a tree structure. Since such tree structure requires a root node. *Model* table is devised to express the root node and to share the structure information. The X_{DEVS} and Y_{DEVS} are used in both atomic and coupled models in the following definitions.

$$\begin{aligned} Model' &= \{id, parent\_id, model\_name\} \\ X' &= \{id, ev\_name, type\_id, val\} \\ Y' &= \{id, ev\_name, type\_id, val\} \end{aligned}$$

$$
\begin{aligned}
X_{DEVS} &= \Pi_{(ev\_name, type\_id)}(Model \bowtie X) \\
Y_{DEVS} &= \Pi_{(ev\_name, type\_id)}(Model \bowtie Y) \\
& \text{where ' means the attributes set}
\end{aligned}
$$

- Atomic model definition.
  An atomic model has 7-tuple ,except X and Y, which is included by Model definition.

  It is very complex that the four characteristic functions are translated into tabular form. The internal transition function, $\delta_{int} : S \rightarrow S$, means that a set $\{(s, s')|s, s' \in S\}$ is a transition tuple set. If the transition tuple set has a small cardinality, the transition set can be directly translated into tabular form. But, as the cardinality increases, the size of the table increases, Such translation leads to table space explosion. To solve such problem, a function, func_I, is introduced for the transition function of DEVS formalism. The func_I requires the operation information which is named as *func* or *act_func*. An atomic model definition in RA is as follows.

$$
\begin{aligned}
S' &= \{id, name, type\_id, val\} \\
Init\_cond' &= \{id, func\} \\
\delta'_{int} &= \{id, func, act\_func\} \\
\delta'_{ext} &= \{id, func, iport\_id, act\_func\} \\
\lambda' &= \{id, func, oport\_id, act\_func\} \\
ta' &= \{id, func, type, taval, ta\_id\} \\
ta'.type &\in \{INFINITY, SIGMA, RAND\} \\
func' &= \{id, func, op, e1, e2\} \\
func'.op &\in \{STATEID, PLUS, \cdots\}
\end{aligned}
$$

- Coupled model definition.
  A coupled model has just structure information such as child(ren) information and coupling scheme(s). It is easily translated into the tabular form as below.

$$
\begin{aligned}
M' &= \{id, Cid, Mid, Cname\} \\
EIC' &= \{id, Mid, Miportid, Cid, Ciportid\} \\
EOC' &= \{id, Cid, Coportid, Mid, Moportid\} \\
IC' &= \{id, src\_Cid, src\_Coportid, dst\_Cid, \\
& \quad dst\_Ciportid\} \\
SELECT' &= \{id, Cid, priority\}
\end{aligned}
$$

3.2.2 Function Interpretation

To express more general function, the characteristic function interpreter, func_I, supports arithmatic, logic expression and state transition. It is executed in the characteristic functions of the atomic model by function call.
In brief, func_I is as follows.

```
proc func_I ($cstate, cfunc : integer$)
begin
    $val$ : integer;
    $(op, e1, e2) := \Pi_{(T.op, T.e1, T.e2)}($
        $\sigma_{T.func=cfunc}\ (\rho_T(\text{func})))$
    if ($op$ = PLUS) then
        $val := \text{func\_I}(cstate, e1)$
            $+ \text{func\_I}(cstate, e2);$
    else if ($op$ = STATEID) then
        $val := \Pi_{val}(\sigma_{id=cstate \wedge name=e1}(S'))$
    else
    ...
    endif
    return val;
end
```

The four characteristic function, $\delta_{int}, \delta_{ext}, \lambda, ta$ is defined by using func_I in RA as follows.

```
proc $\delta_{int}(id, cstate)$
begin
    $action\_id := \Pi_{T.action\_id}($
        $\sigma_{\text{func\_I}(cstate, T.func) \wedge T.id=id}(\rho_T(\delta'_{int})));$
    $\text{func\_I}(cstate, action\_id);$
end

proc $\delta_{ext}(id, cstate, e, portid)$
begin
    $action\_id := \Pi_{T.action\_id}$
$(\sigma_{T.iport\_id=portid \wedge \text{func\_I}(T.func) \wedge T.id=id}(\rho_T(\delta'_{ext})));$
    $\text{func\_I}(cstate, action\_id);$
end

proc $\lambda(id, cstate)$
begin
foreach $(port\_id, action\_id) \in \Pi_{(T.port\_id, T.action\_id)}($
    $\sigma_{T.id=id \wedge \text{func\_I}(T.func)}(\rho_T(\lambda')))$
$Y' \leftarrow \sigma_{T.id=id \wedge T.ev\_name=port\_id}$
    $(\sigma_{T.val \leftarrow \text{func\_I}(cstate, action\_id)}(\rho_T(Y')));$
endforeach
end

proc $ta(id, cstate)$
begin
    $(tatype, taval, ta\_id) := \Pi_{(T.tatype, T.taval, T.ta_id)}$
        $(\sigma_{T.id=id \wedge \text{func\_I}(T.func)}(\rho_T(ta')));$
    if (tatype = INFINITY) then
        $val$ := INFINITY;
    else if ($tatype$ = SIGMA) then
    $val := \Pi_{T.val}(\sigma_{(T.id=id \wedge T.name=cstate)}(\rho_T(S')));$
    else if ($tatype$ = RAND) then
```

```
        $val$ := random();
    endif
    return $val$;
end
```

4 A COMPLETE EXAMPLE: PURCHASING PROCESS

This section provides an example of a Workflow DEVS modeling and simulation for a business process. Although a company is composed of many divisions, we focus on the purchasing division. Its primary tasks are request processing, purchasing, delivery, and accounting.

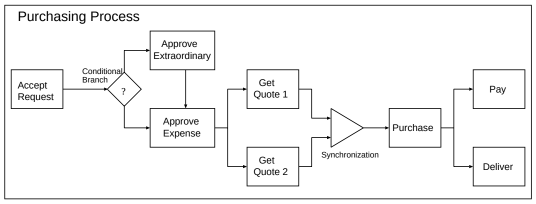
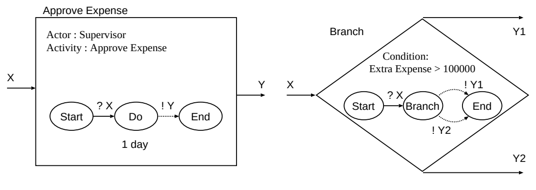


Figure 2: Workflow DEVS Model for Purchasing Process

Figure 2 shows a Workflow DEVS model for purchasing process. Purchasing requests from an outer environment are processed by the **Accept Request** task. Then, a flow is divided into two paths depending on the amount of expense. If the amount is above the specified expense two approval tasks, **Approve Extraordinary** and **Approve Expense**, are required. Otherwise, only one approval task, **Approve Expense**, is needed. The approval tasks are followed by the **Get Quote**. To purchase items at the lowest price, price information must be collected by the **Get Quote** tasks. The example has two **Get Quote** tasks to be processed simultaneously. After both tasks end, the **Purchase** task is processed. Based on the information from the previous tasks purchasing decision is made by which requested items are purchased. Finally payment for purchasing and delivery of purchased items are processed. The two tasks, **Pay** and **Deliver**, can be processed simultaneously.

In Workflow DEVS modeling, each task is represented by a DEVS model. A task can be mapped into either a coupled model or an atomic model. In Figure 2, **Branch** and **Synchronization** models are the flow-control models and the remains are subtask models. As shown in Figure 2, arrows, which connect between submodels, means a business task sequence. For example, in Figure 2 **Purchase** and **Pay** models are connected by an arrow, meaning that the **Pay** task starts only after the **Purchase** task is completed.



(a) Atomic DEVS model for an activity (b) Atomic DEVS model for a conditional branch

Figure 3: Atomic DEVS Model for Activity and Flow-control

Figure 3 shows how to express business tasks into atomic models in DEVS formalism. (a) is an atomic model for the **Approve Expense** subtask and (b) is an atomic model for the **Branch** flow-control model. A subtask starts by 'X' input, processes the tasks and ends by sending out 'Y' output. It includes the following task informations. Actor - who will take the task, activity - how the task will be accomplished, and duration - how long the task takes, and so on. A workflow engine exploits this information to accomplish the task.

To simulate the **Purchasing Process** model in Figure 2, the model is first described in DEDL 4. Then the model in DEDL is translated into RADESIF 5 and is executed by RIFJAC with DEVSim-Java. Figure 6 shows a simulation process in DEVSim-Java.

5 CONCLUSIONS

This paper describes a workflow modeling simulation environment within the DEVS framework. The framework includes DEVSim-Java which is a Java realization of DEVS formalism. With the DEVSim-Java simulation and workflow engine, we can simulate and execute DEVS models in the web browser. The developed web-based workflow modeling and simulation environment consists of DEVSim-Java along with supporting tools for automatic semantics translation and format conversion. The tools include DEDL (DEVS Definition Language), DERAT (DEDL-to-RADESIF Translator), DEJAT (DEDL-to-DEVSim-Java Translator), and RIFJAC (RADESIF-to-DEVSim-Java Converter).

```
    ...
interface ApproveExpense
    input : {X}
    outputs : {Y}
end ApproveExpense;

atomic model ApproveExpense
    state variables :
        Status in {Start, Do, End};
    initial condition:
        Status := Start;
    internal transition:
        (Status=Do) => {
            Status:=End;
        }
    external transition:
        (Status=Start)*X => {
            Status:=Do;
        }
    output function:
        (Status=Do) => Y;
    time advance :
        (Status=Start) => ta:=infinity;
        (Status=Do) => ta:=24;
        (Status=End) => ta:=infinity;
end ApproveExpense;
    ...
```

Figure 4: Part of the DEDL Description of Purchasing Process Model

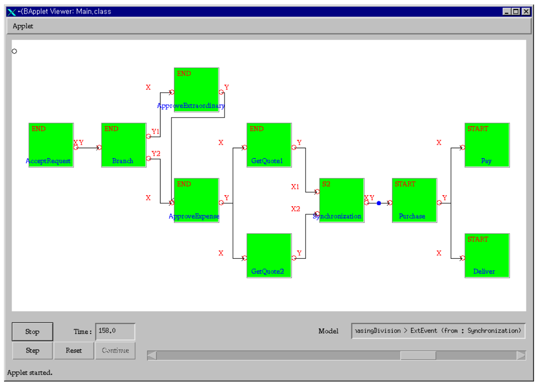
(x) Func table

id	func	operation	e1	e2
...	...	...	...	...
2	48	ASSIGN	49	50
2	49	STATEID	Status	<NULL>
2	50	INTEGER	Start	<NULL>
2	51	EQ	52	53
2	52	STATEID	Status	<NULL>
2	53	INTEGER	Do	<NULL>
...	...	...	...	...

Figure 5: RADESIF for Purchasing Process

We have shown that a workflow modeling simulatoin environment instrumented with the relational algebraic DEVS framework can satisfy the interoperability requirements of workflow management systems. From a business process definition, we construct a Workflow DEVS Model as a workflow exeuction model. The Workflow DEVS Model which is translated into the RADESIF format can be easily stored into relational databases and synthesized for the purpose of simulation or real-time execution. RADESIF is also used to exchange workflow definitions among different definition tools and/or simulation engines.

A complete example has been shown for the modeling simulation procedures from DEDL to the DEVSim-Java simulation engine.



158.0

Model nasingDivision > ExtEvent (from : Synchronization)

Applet started.

Figure 6: Simulation Process Screen Shot

References

[CHRW98] Andrzej Cichocki, Abdelsalam (Sumi) Helal, Marek Rusinkiewicz, and Darrell Woelk, *Workflow and Process Automation: Concepts and Technology*, Norwell, MA, Kluwer Academic Publishers, 1998.

[HK96] G.P Hong , T.G Kim, "A Framework for Verifying Discrete Event Models Within a DEVS-Based System Development Methodology," *TRANSACTIONS of The Society for Computer Simulation*, vol. 13, no.1, pp.19-34, 1996.

[Kim94] Tag Gon Kim, *DEVSim++ User's Manual: C++ Based Simulation with Hierarchical, Modular DEVS Models*, ftp://sim.kaist.ac.kr/pub/DEVSim++-1.0/, Systems Modeling Simulation Lab., KAIST, Taejon, Korea, 1994.

[SKS97] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan, *Database System Concepts*, McGraw-Hill Companies, Inc., 1997.

[WFMC95] David Hollingsworth, *Workflow Management Coalition The Workflow Reference Model*, The Workflow Management Coalition, Brussels, Belgium, 1995.

[Zeigler84] B. P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Orlando, FL, Academic Press, 1984.