DEVS MODELING OF RUN-TIME WORKFLOW SIMULATION AND ITS APPLICATION

Byoung K. Choi, Duckwoong Lee, and Dong H. Kang
Department of Industrial Engineering
KAIST
373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, South Korea
E-mail: bkchoi@kaist.ac.kr

KEYWORDS

Run-time workflow simulation, Parallel simulation, Time synchronization, DEVS model, Workflow Management

ABSTRACT

Existing studies on workflow simulation focused mainly on simulating *process definition models* at *build time*. Recently, a novel workflow simulation method called run-time workflow simulation (RTWS) was proposed with which *process instances* at *run-time* can be simulated. This paper presents a formal DEVS model of the run-time workflow simulation module that can be embedded in an existing business process management system (BPMS). The presented DEVS model has been implemented and a *test-bed BPMS for RTWS experiment* was developed.

INTRODUCTION

Workflow management system (WfMS) is a software system that completely defines and automatically executes workflows in order to manage the actual flow of work in an organization (WfMC 1995). The software module in charge of managing the actual flow of work is called **workflow engine**, and the services it provides to execute the predefined workflows are referred to as enactment service. A portion of workflow engine responsible for enactment service is called **enactment server**. A predefined workflow model is called **process definition model** (**PDM**), while the one that is being executed is called a **process instance**. In recent years, a WfMS is often called a business process management system (BPMS) with an emphasis on orchestrating operational business processes (Delphi 2005, Smith and Fingar 2003). Along with this development comes an increased awareness of need for **workflow simulation** in evaluating and improving business processes (Smith and Fingar 2003).

As discussed in the next section, existing studies on workflow simulation are mainly focusing on simulating a PDM at its build time. Recently, the authors' group proposed a novel simulation method where the future (i.e., *enabled* and *inactive* activities) of process instances is simulated at run-time, which we call **run-time workflow simulation**. This paper represents a formal description of a run-time workflow simulation system (Hwang 2006) using the well-known **DEVS** model primitives (Hong 1997, Zeigler et al. 2000).

In the proposed run-time workflow simulation system, a local simulator is constructed for each **participant** and a separate *enactment server* is employed for the workflow simulation. As it is a parallel simulation, the **time synchronization** issues (Fujimoto 2000) have to be resolved.

BACKGROUND & RELATED WORKS

In order to make the paper self-contained, basics of enactment service mechanism are briefly described using Figure 1. For a given PDM, process instances (**PI**) are created. Shown in the figure is a PI having seven **activities** (including *Start* and *End*). Right after the completion of activity W1, the two activities W2 and W3 are *enabled*, and then the enactment server provides the following sequence of enactment services:

(1) Creates instances of the newly *enabled* activities W2 and W3
(2) Sends out W2 and W3 to respective **work-list handlers** (activity becomes a **new workitem** to be processed by the human participant).
(3) Receives the **completed workitem** W2 (In this scenario, W2 is completed first)
(4) Updates the state of the PI such that W2 becomes a *completed* activity.

Proceedings 22nd European Conference on Modelling and Simulation ©ECMS Loucas S. Louca, Yiorgos Chrysanthou, Zuzana Oplatková, Khalid Al-Begain (Editors)
ISBN: 978-0-9553018-5-8 / ISBN: 978-0-9553018-6-5 (CD)

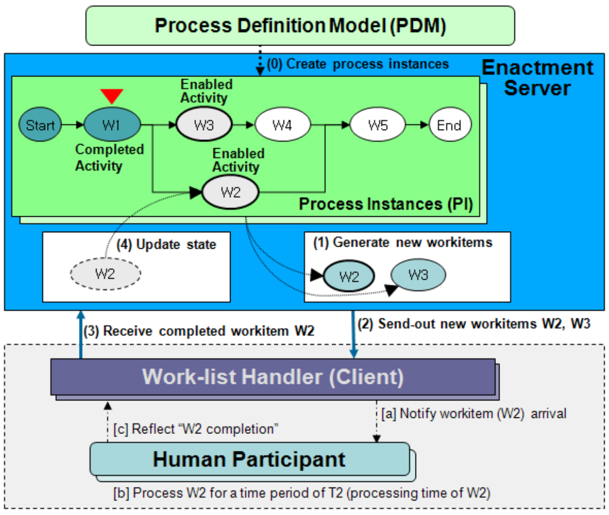

Process Definition Model (PDM)
(0) Create process instances
Enactment Server
Start
W1
Enabled Activity
W3
W4
W5
End
Completed Activity
Enabled Activity
W2
Process Instances (PI)
(4) Update state
W2
(1) Generate new workitems
W2
W3
(3) Receive completed workitem W2
(2) Send-out new workitems W2, W3
Work-list Handler (Client)
[a] Notify workitem (W2) arrival
[c] Reflect "W2 completion"
Human Participant
[b] Process W2 for a time period of T2 (processing time of W2)

Figure 1: Enactment Service Mechanism

Upon receiving the new workitem W2, the work-list handler *notifies* its participant, and the participant works on W2 for a time period of T2 and *reflects* the results at the work-list handler so that the completed workitem is returned back to the enactment server.

The communications between the enactment server (workflow engine) and the work-list handlers are made based on the **interface standards** provided by Workflow Management Coalition (WfMC 1995). For this purpose, the workflow engine has a type of internal data called workflow **relevant data** that can be manipulated by work-list handlers as well as by the workflow engine.

Most of workflow simulation studies focused on validating and/or optimizing a given PDM via simulation. The basic approach here is that the PDM is converted into a formal model, such as Petri-net (Peterson 1981) or DEVS (Zeigler et al. 2000), and then simulation is performed using this converted model. In some commercial BPMS (Bizflow 2008, FileNet 2003), PDM is converted into a proprietary simulation language. There are quite a few studies reported in the literature (Aalst and Hofstede 2000, Bae et al. 1999, Chan 2004, Greasley 2003, Hong et al. 2003, Li et al. 2002, Li at el. 2005).

RUN-TIME WORKFLOW SIMULATION

Recently, a run-time workflow simulation (**RTWS**) method was proposed in which the future of the enactment service process of a BPMS is simulated at any point in time by a RTWS module embedded in the workflow engine (Hwang 2006). It was demonstrated that a BPMS equipped with a RTWS module could be used as a *simulation-based job shop scheduling* system (Hwang and Choi 2007). The RTWS module consists of an enactment server, a **synchronization manager**, and a number of **participant simulators** as depicted in Figure 2. In order to perform a run-time workflow simulation, a copy of each process instance (PI) being executed is obtained and the work-list handler ID of each activity in the PI is changed to the corresponding participant simulator ID. Now, the enactment server of the RTWS module has a set of modified process instances (PI*).

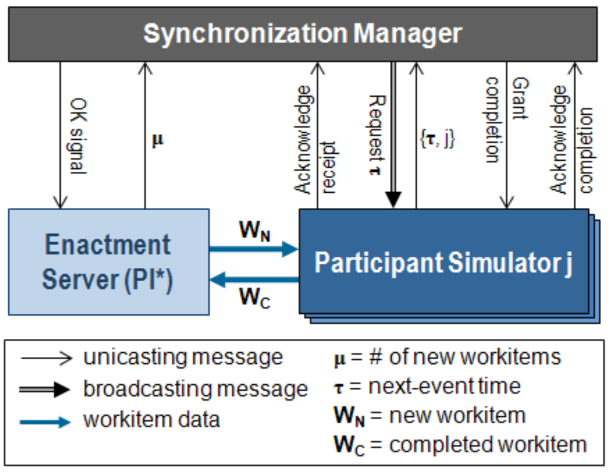


Figure 2: Run-time Workflow Simulation Module

The participant simulators in the RTWS module are **logical processors** in a parallel simulation system in which **centralized barriers** are often used for time synchronization (Fujimoto 2000). Starting from the point when an Enactment **Server** receives a completed workitem $\mathbf{W_C}$ from a Participant **Simulator**, the RTWS module of Figure 2 performs the following operations: a) Server waits until OK signal comes from Synchronization **Manager** and then updates PI* and generates new workitems, b) Server sends $\boldsymbol{\mu}$ (number of new workitems) to Manager and then sends newly generated workitems $\mathbf{W_N}$ to Simulator, c) Simulator acknowledges the receipt of workitem to Manager, d) Manager waits until the receipt count equals to $\boldsymbol{\mu}$ and then requests Simulators to send their ***next-event time*** $\boldsymbol{\tau}$, e) each Simulator **j** sends its $\boldsymbol{\tau}$ to Manager, f) Manager waits until it receives $\boldsymbol{\tau}$ from all Simulators and then grants the Simulator that has minimum $\boldsymbol{\tau}$ value to complete its workitem, g) the granted Simulator acknowledges its workitem completion to Manager and sends the completed workitem $\mathbf{W_C}$ to Server. The wait-until points in Steps a, d, and f are *centralized barriers*.

DEVS MODEL OF RUN-TIME WORKFLOW SIMULATION

The internal structure of a parallel system is not easy to describe as a sequential algorithm. In this section, we describe the RTWS module (Figure 2) as DEVS coupled model (Ziegler et al. 2000) using the well known **DEVS model primitives** (Hong 1997) as shown in Table 1. For example, '**?**' symbol is used for *external input*, '**!**' symbol for *message output*, '∼' symbol for *condition*, etc.

Shown in Figure 3 is our RTWS module with the synchronization manager and enactment server represented as DEVS atomic models. The structure of the **Enactment Server** is quite simple: Normally, it stays in the initial state 'Ready'. When it receives both a completed workitem $\mathbf{W}_C$ and 'OK signal', it 1) updates the process instances **PI***, 2) generates new workitems $\{\mathbf{W}_N\}$, 3) sends out $\boldsymbol{\mu}$ (number of newly generated workitems) to the synchronization manager, and 4) sends out new workitems $\{\mathbf{W}_N\}$ to participant simulators.

Table 1: DEVS Model Primitives

Primitives	Notation
External Transition with Condition	Condition ? External-input, Value-update, ! Message-output
Internal Transition with Condition	Condition Value-update, ! Message-output
State	Initial State

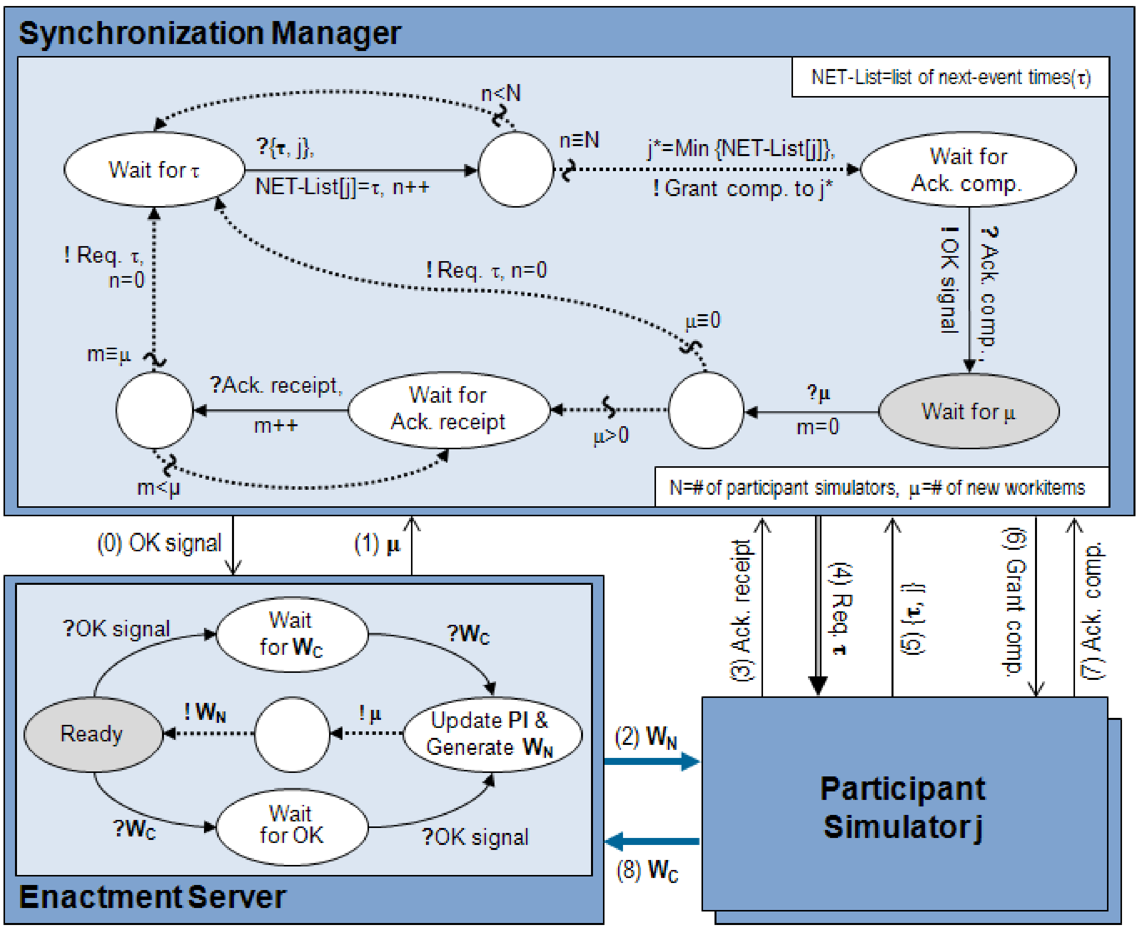


Figure 3: DEVS Model of Synchronization Manager

The DEVS model of Synchronized Manager (**SM**) has four 'Wait for' states: 1) Wait for μ, 2) Wait for acknowledge receipt, 3) Wait for τ and 4) Wait for acknowledge completion. Once initialized, SM stays in the 'Wait for μ' state. Upon receiving $\boldsymbol{\mu}$, SM sits in the 'Wait for Ack. receipt' state until the 'receipt' count becomes μ and then moves to the 'Wait for τ' state while sending out 'Request τ' message to all participant simulators (if μ≡0, it directly moves to the 'Wait for τ' state). And SM waits there while storing the next-event time (NET) $\boldsymbol{\tau}$ of simulator j in NET-List[j]. If it receives $\{\boldsymbol{\tau}, \mathrm{j}\}$ from all simulators, 1) the simulator **j*** that has the smallest **τ** is selected (i.e., j*=Min NET-List[j]), 2) 'Grant comp.' message is sent to the selected simulator, and 3) its state is changed to 'Wait for Ack. comp.' Finally, when SM receives the 'Ack. comp.' message from the selected simulator, it sends out 'OK signal' to the enactment server and moves to the initial state 'Wait for μ' to initiate the next cycle.

Shown in Figure 4 is a *coupled DEVS model* of a 'single machine' participant simulator. A typical *human participant* of a BPMS may be regarded as a **single machine system** in which arriving jobs (i.e., new workitems) are put into a queue and a job is selected for processing based on a dispatching rule. The coupled model consists of Coordinator, Queue and Processor each of which is an atomic DEVS model. It is a typical single machine system model.

The **Coordinator** model has four 'Wait for' states: 1) Wait for new workitems $\mathbf{W}_N$, 2) Wait for τ 3) Wait for Grant completion and 4) Wait for completed workitem $\mathbf{W}_C$. Normally, it stays in 'Wait for $\mathbf{W}_N$' while passing received $\mathbf{W}_N$ to Queue and sending 'Ack. receipt' message back to Enactment Server. If 'Req. τ' message is received, it moves to 'Wait for τ' after sending out 'ask τ' message to Processor. Then upon receiving **return τ**, it moves to 'Wait for Grant comp.' after passing received **τ** to Synchronization Manager. At this point, its state may be changed back to 'Wait for τ' (if 'Req. τ' is received) or to the next state 'Wait for $\mathbf{W}_C$' (if 'Grant comp.' message is received). Finally, it moves to the initial state if it receives $\mathbf{W}_C$ from Processor.

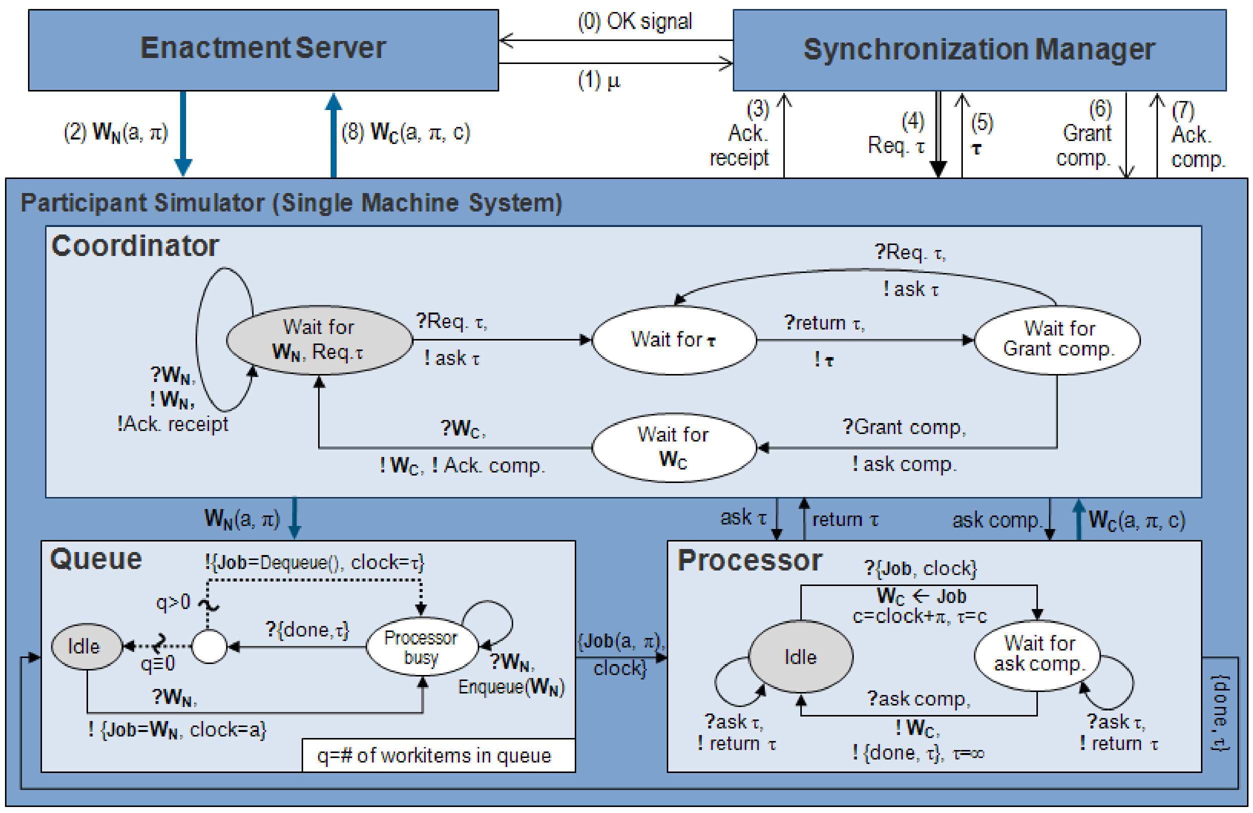


Figure 4: DEVS Model of a Participant Simulator

The **Queue** model receives a new workitem $\mathbf{W}_N$ (with *arrival time-stamp* 'a' and *processing time* 'π') from the Coordinator. The received $\mathbf{W}_N$ is either sent to idle Processor or stored in a queue (i.e., Enqueue) when Processor is busy. Upon receiving a 'done' signal, it retrieves a job ($\mathbf{W}_N$) from the queue (Dequeue) and sends the job (together with the job-start time '**clock**') to the Processor.

The reader is advised to examine the **Processor** model shown in Figure 4 which should be self-explanatory (Note: c = completion time-stamp).

IMPLEMENTATION AND APPLICATION

The DEVS model of a run-time workflow simulation (RTWS) module has been implemented and embedded in a test-bed BPMS named Harmony® (Choi and Hwang 2005). As it is not convenient to perform RTWS experiments under a real life environment, we have developed a test-bed BPMS for run-time workflow simulation as shown in Figure 5.

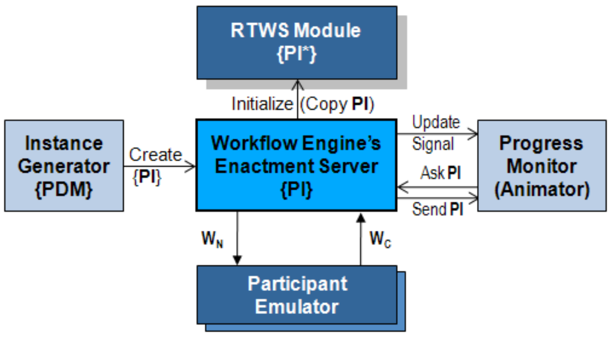
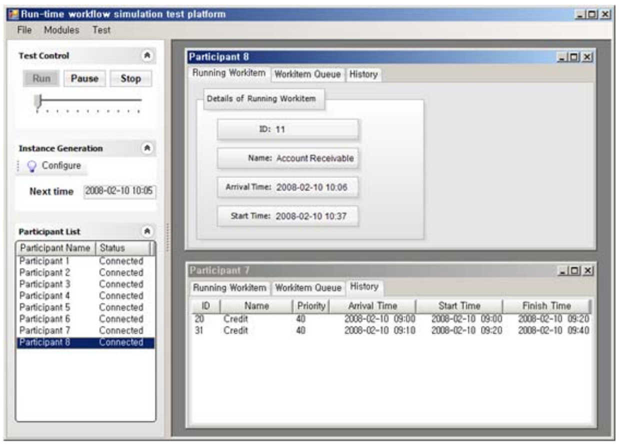


Figure 5: A Test-bed BPMS for RTWS Experiments

The test-bed BPMS for the RTWS experiment consists of the original enactment server, the RTWS module, an instance generator, a number of participant emulators, and a progress monitor. Each of the work-list handler is replaced by a **participant emulator** which is a *scaled real-time simulator* representing a human participant. The **instance generator** creates a sequence of process instances {PI} in scaled real time according to a predefined schedule. With the two components added to the original enactment server, we can create a virtual enactment service scenario. Figure 6 shows a GUI for *process instance generation* (left-middle) and *participant emulation* (right).



Next time 2008-02-10 10:05

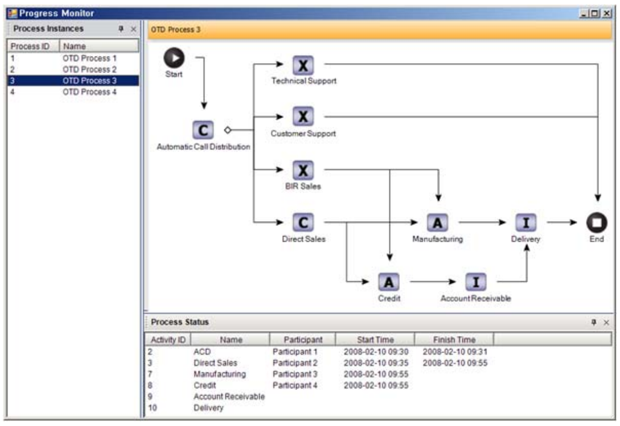
Participant Name	Status
Participant 1	Connected
Participant 2	Connected
Participant 3	Connected
Participant 4	Connected
Participant 5	Connected
Participant 6	Connected
Participant 7	Connected
Participant 8	Connected

ID: 11
Name: Account Receivable
Arrival Time: 2008-02-10 10:06
Start Time: 2008-02-10 10:37

ID	Name	Priority	Arrival Time	Start Time	Finish Time
20	Credit	40	2008-02-10 09:00	2008-02-10 09:00	2008-02-10 09:20
31	Credit	40	2008-02-10 09:10	2008-02-10 09:20	2008-02-10 09:40

Figure 6: GUI for Process Instance Generation and Participant Emulation

The **progress monitor** component displays the state changes of the selected process instances (i.e., it animates the progresses of process instances). Also, it is responsible for visualizing the overall progress of the business process. Shown in Figure 7 is a display screen for the progress monitor (C = completed activity, A = active activity, I = inactive activity, X = excluded activity).



Process ID	Name
1	OTD Process 1
2	OTD Process 2
3	OTD Process 3
4	OTD Process 4

Activity ID	Name	Participant	Start Time	Finish Time
2	ACD	Participant 1	2008-02-10 09:30	2008-02-10 09:31
3	Direct Sales	Participant 2	2008-02-10 09:35	2008-02-10 09:55
7	Manufacturing	Participant 3	2008-02-10 09:55	
8	Credit	Participant 4	2008-02-10 09:55	
9	Account Receivable			
10	Delivery			

Figure 7: Progress Monitor's GUI

At any point in time, the user may trigger the **RTWS module** to start a workflow simulation. As discussed earlier, the RTWS module (See Figure 2) gets a copy (PI*) of each process instance PI that is stored in a DB and performs simulation in an as-fast-as-possible manner. In the current implementation, every participant is modeled as a 'single machine system' consisting of a queue and a processor (See Figure 4).

CONCLUSION

Presented in this paper is a formal DEVS model of a run-time workflow simulation (RTWS) module that can be embedded in an existing BPMS (business process management system). The proposed RTWS module is a parallel simulation system having a number of *logical processors* (participant simulators and enactment server) and a *synchronization manager*, and centralized barriers are used for time synchronization. Also presented is a test-bed BPMS for RTWS experiment consisting of the original enactment server, a RTWS module, a process instance generator, the participant emulators, and a progress monitor. The main purpose of the research is to demonstrate the validity of the presented DEVS model using a test-bed BPMS. Further research efforts might be need in order to apply it to commercial BPMSs.

ACKNOWLEGEMENT

This research was supported by the Korea Science and Engineering Foundation (KOSEF): No.R01-2006-000-11118-0.

REFERENCES

Aalst, W.M.P. and Hofstede, A.H.M., 2000, Verification of Workflow Task Structures: A Petri-Net-Based Approach, *Information Systems*, v.25, no.1, pp.43-69.

Bae, J.S., Jeong, S.C., Seo, Y.H., Kim, Y.H. and Kang, S.H., 1999, Integration of Workflow Management and Simulation, *Computers & Industrial Engineering*, v.37, pp.203-206.

BizFlow®, 2008, http://www.handysoft.com

Chan, D.Y.K., 2004, Design An Effective Workflow in Simulation, *The 8th International Conference on Computer Supported Cooperative Work in Design Proceedings*, v.2, pp.324-328.

Choi, B.K. and Hwang, H.C., 2005, Architecture of A 3-layered Closed-loop BPMS Harmony®, *KAIST VMS Lab Technical Report*, http://vms.kaist.ac.kr/publications/TechnicalReport/3layer.pdf (in Korean)

Delphi, 2005, *BPM 2005 Market Milestone Report*, http://www.delphigroup.com/research/whitepaper_request_download.htm

FileNet, Bruce Silver Associates, 2003, Event-driven Business Process Management. *Industry Trend Reports*, http://whitepapers.silicon.com

Fujimoto, R.M., 2000, *Parallel and Distributed Simulation Systems*, John Wiley & Sons.

Greasley, A., 2003, A Simulation of A Workflow Management System, *Work study*, v.52, no.5, pp.256-261.

Hong, G.P., 1997, DEVS-Based Framework for Logical and Performance Analysis of Discrete Event Systems, *Doctoral Thesis*, *Department of Electronic Engineering, KAIST*.

Hong, K.J., Lee, J.K., Kim, D.H. and Kim, T.G., 2003, DEVS Framework Instrumented with Database for Web-Based Workflow Modeling Simulation, *Society for Computer Simulation (Simulation Councils, Inc.)*, v.31, no.3, pp.113-118.

Hwang, H.C. and Choi, B.K., 2007, Workflow-based Dynamic Scheduling of Job Shop Operations, *International Journal of Computer Integrated Manufacturing*, v.20, no.6, pp.557-566.

Hwang, H.C., 2006, Development of a Workflow Engine based Simulation System, *Doctoral Thesis, Department of Industrial Engineering, KAIST.*

Li, J.J., Casati, F. and Shan, M.C., 2002, Business Process Simulation with HP Process Manager, *Simulation series*, v.34, no.1, pp.291-295.

Li, R., Zhu, Z., Wang, X., Liu, L. and Jiang, X., 2005, Workflow Simulation and Its System Development, *Proceedings of SPIE-the International Society for Optical Engineering*, v.6041, pp.604104.

Peterson, J.L., 1981, *Petri Net Theory and the Modeling of Systems*, Prentice Hall.

Smith, H. and Fingar, P., 2003, *Business Process Management: The Third Wave*, Meghan-Kiffer Press, 2003.

Workflow Management Coalition (WfMC), WfMC-TC-00-1003, 1995, *The Workflow Reference Model*, http://www.wfmc.org

Zeigler, B., Praehofer, H. and Kim, T., 2000, *Theory of Modeling and Simulation*, Academic Press: Boston.

AUTHOR BIOGRAPHICS

Byoung K. Choi is a professor of the Department of Industrial Engineering at KAIST since 1983. He received a BS from Seoul National University in 1973, a MS from KAIST in 1975, and a Ph.D. from Purdue University in 1982, all in Industrial Engineering. His current research interests are system modeling and simulation, BPMS, simulation-based scheduling, and virtual manufacturing.

Duckwoong Lee is a PhD candidate student in the Department of Industrial Engineering at KAIST. He received as BS from Ajou University in 2002, a MS from KAIST in 2004, all in Industrial Engineering. His research interests are in the area of BPMS, system modeling and simulation, and parallel and distributed simulation. He can be reached at ldw721@vmslab.kaist.ac.kr

Dong H. Kang is a PhD candidate student in the Department of Industrial Engineering at KAIST. He received as BS from KAIST in 2003 in Computer Science, a MS from KAIST in 2005 in Industrial Engineering. His research interests are in the area of BPMS, system modeling and simulation and RFID application. He can be reached at donghun.kang@vmslab.kaist.ac.kr