The development of a method for integration between different types of simulators

H. Hibino[1], Y. Fukuda[2], M. Nakano[3], S. Sato[3]
[1]Technical Research Institute of JSPMI(Japan Society for the Promotion of Machine Industry) ,
1-1-12, Hachiman-cho, Higashikurume,Tokyo,203,Japan
Tel: +81-424-75-1188
Fax: +81-424-75-0947
e-mail: hibino@tri.jspmi.or.jp
[2]Hosei University,
3-7-2 Kajino-cho, Koganei, Tokyo,184,Japan
Tel: +81-423-87-6358
Fax: +81-423-87-6126
e-mail: fukuda@is.hosei.ac.jp
[3]Toyota Central R&D Labs., Inc
480-11 Nagakute, Aichi, Japan
Tel: +81-561-63-4604
Fax: +81-561-63-6121
e-mail: nakano@robotics.tytlabs.co.jp,
sato@robotics.tytlabs.co.jp

Abstract

A manufacturing system simulator plays an important role in designing new systems. However, as simulators depend on particular usage, the modeling method of each simulator is different. Therefore simulation users cannot cooperate when using different models. In order to solve this problem, it is necessary to be able to integrate different models in each simulator. The purpose of this research is to develop a method of integration between the simulators which do not have the rollback function.

In this paper, we propose a concept for the integration between simulators without the rollback function using a storage model concept for manufacturing system designs. The functions and the implementation method for the proposed concept are described. Then, a case study carried out to evaluate the performance of the cooperative work, is presented.

Keywords

Manufacturing system, simulation, distributed simulation, simulation model, CORBA, TCP/IP socket, client-server application, simulator, system integration, object-oriented simulator.

1 INTRODUCTION

Manufacturing systems are being created on larger and more complicated scales than ever before. In designing such a manufacturing system, a manufacturing system simulator plays an important role. However, as simulators depend on particular usage, the modeling method of each simulator is different (Fujii, 1999, Hibino, 1999, Kryssanov, 1998). Simulation users cannot cooperate when using different models. In order to solve this problem, it is necessary to be able to integrate different models in each type of simulators.

Many distributed simulation systems have been proposed (e.g. Fujii, 1999, Fujimoto, 1990, 1995, Jones, 1998, Nicol 1997). A majority of the current methods can evaluate a variety of systems and areas but are only models to estimate the design but present problems when adapted to actual system design processes. To design systems for specific areas, methods which are adapted to specific design processes and their characteristics are needed. In the case of designing manufacturing systems, one method has been proposed using the rollback function which is to return the simulation clock to passed time to synchronize the events among the simulations leading to more accurate evaluation (Fujii, 1999). However, commercial based simulators do not include the rollback function. Therefore the purpose of this research is to develop a method of integration between the simulators which do not have the rollback function for manufacturing system designs.

Based on our past research results for analyzing manufacturing system designs, a manufacturing system is divided into a number of subsystems based on various specifications required by the system and partial optimization in each subsystem is attempted (Hibino, 1999). Each subsystem can be modeled as a unit. Relationships between a subsystem and other subsystems can be arranged and defined as input and output of material flow. Storage function units such as warehouse and buffers are usually located intermediately between the subsystems such as machining line subsystems, assembly line subsystems and so on. Therefore we focus on the storage function units as interfaces to connect different types of simulators. In simulators the storage function units are defined as storage models.

In this paper, we propose a concept for integration between simulators without the rollback function using a storage model concept for manufacturing system designs.

The functions and the implementation method for the proposed concept are described. Then, a case study was carried out to evaluate the performance of cooperative work.

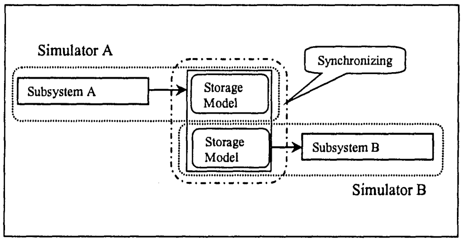
2 THE CONCEPT OF STORAGE MODEL

In the case of designing manufacturing systems using simulators, a manufacturing system is divided into a number of subsystems based on various specifications required, and then each subsystem is modeled and evaluated using a suitable simulator in response to the purposes required (Hibino, 1999). However, as simulators depend on a particular usage, the modeling method of each simulator varies. Simulation users cannot easily cooperate when using different models. This creates problems in the manufacturing system design.

In order to connect the subsystem models in particular simulators, we propose a method of integration between different simulation models using the storage model.

One of the fundamental assumptions is shown in Figure 1. The manufacturing system consists of two subsystems; a machining subsystem and an assembly subsystem. A storage model is located intermediately between the subsystems. Each subsystem is modeled by different simulators (A and B). The storage is modeled as a storage model by each simulator. The simulators are then synchronized via the storage models.

Through analyzing changes in the storage model by considering a kind of products, the method to synchronize the simulators is described.



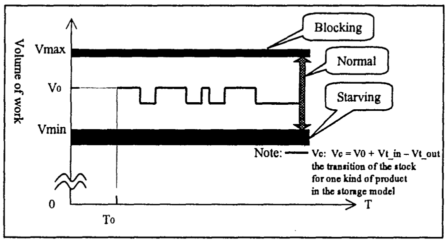


Figure 1 Concept of the proposed storage model

Figure 2 Outline of the distinction between the storage model states

First, we analyze the changes in the stock amount. The relationships of the stock amounts can be expressed using the following formula.

$V_c = V_o + V_{t\_in} - V_{t\_out}$;

Vc:	the volume of the current stock in the storage model
Vo:	the volume of the initial stock in the storage model
Vt_in:	the total storaging volume into the storage model
Vt_out:	the total shipping volume from the storage model

The states of the storage model are classified into three statuses labeled starving status, blocking status, and normal status.

$V_{min} \geq V_c$	: starving status
$V_{max} \leq V_c$	: blocking status
$V_{max} > V_c > V_{min}$	: normal status
Vmin	: the minimum volume of the stock in the storage model
Vmax	: the maximum volume of the stock in the storage model

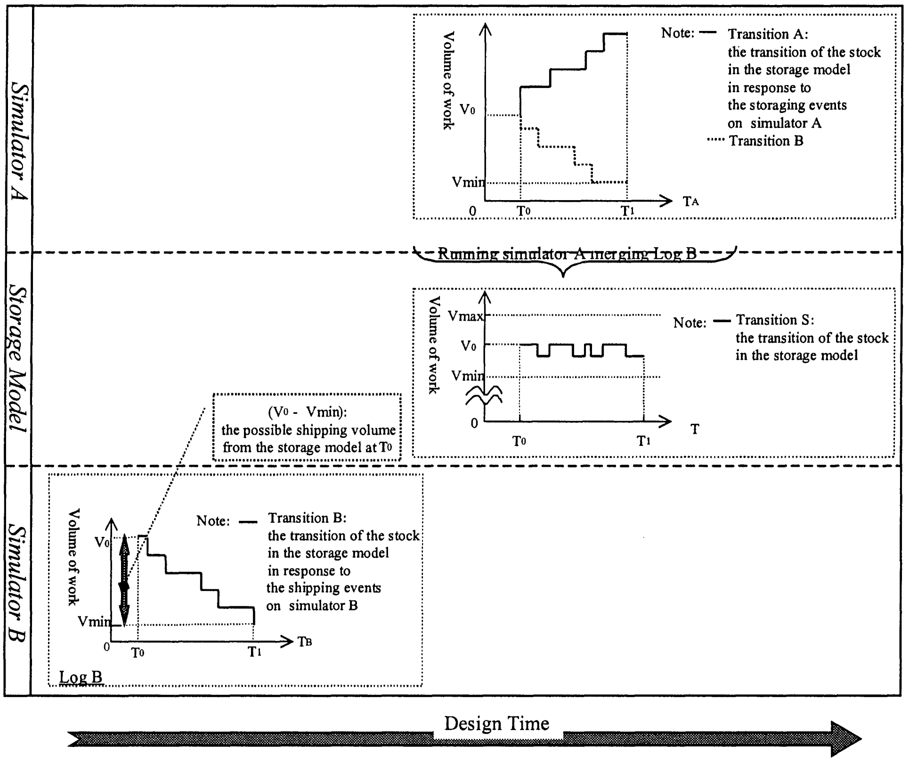
An outline of the distinction between the storage model states is shown in Figure 2. Secondly, we describe our proposed method using the results.

The first step in synchronizing the simulators is to detect how long each simulator is able to run alone. One of the promised periods which are not necessary to carry out rollback operations can be reached by running one simulator until the current state of the storage model changes into another state using the initial parameters as V_0, Vmin, and Vmax. For examples using the case of Figure 1 and Figure 2, the examined time until the storage state changes the normal status into the starving status is reached by using the following procedure.

1. The possible shipping volume from the storage model at T_0 is calculated by $(V_0 - V_{min})$.
2. Simulator B runs until the shipping volume equals $(V_0 - V_{min})$.
3. The examined time (T_1) is reached.

The second step to synchronizing the simulators is to maintain consistency of the events in the storage model. After the first step, simulator A is made to run until T_1. At that time the events which occurred for the storage model on simulator B are simulated on simulator A using a log from simulator B. In the log the events are written along with the simulation clock in simulator A. Then the storage model volume at T_1 is reached. The same procedure is continued to synchronize the simulators. An outline of this procedure is shown in Figure 3.

Furthermore the procedures to synchronize the simulators are different in response to the storage model states. Excepting the normal status, there are two storage model states, starving and blocking. When the storage model state is in blocking status, the same procedure as above is followed. When the storage model state is in starving status, another type of procedure must be taken. An outline of this procedure is shown in Figure 4. When the storage model state is in normal status, either procedure can be chosen.



Simulator A
Storage Model
Simulator B
Volume of work
Vo
Vmin
0
To
T1
TA
Note: — Transition A: the transition of the stock in the storage model in response to the storaging events on simulator A
····· Transition B
Running simulator A merging Log B
Vmax
T
Note: — Transition S: the transition of the stock in the storage model
(Vo - Vmin): the possible shipping volume from the storage model at To
Note: — Transition B: the transition of the stock in the storage model in response to the shipping events on simulator B
TB
Log B
Design Time

Figure 3 Outline of the procedure of the proposed method (1)

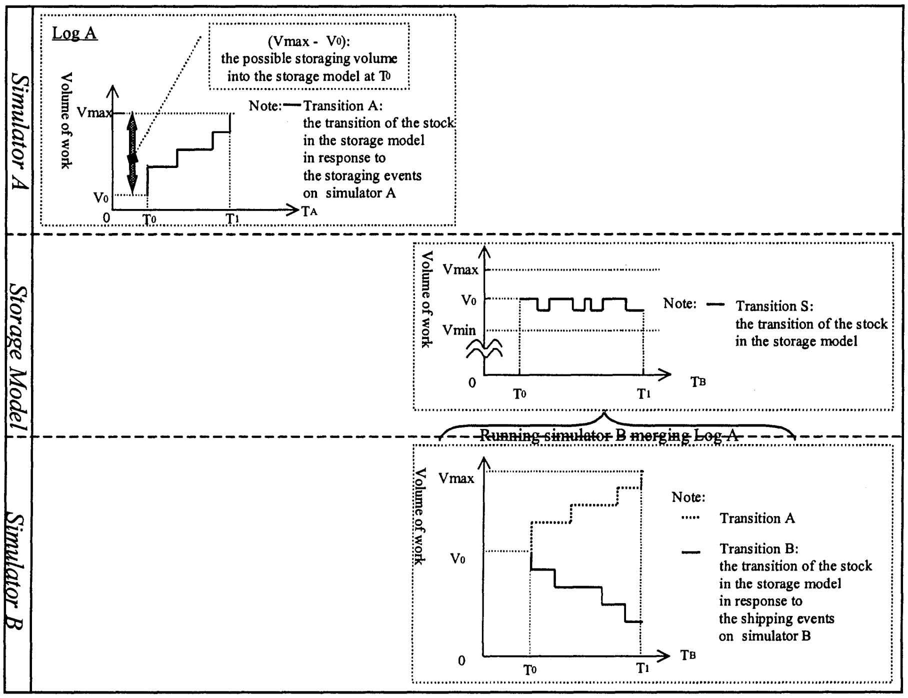

Simulator A
Storage Model
Simulator B
Log A
(Vmax - V0): the possible storaging volume into the storage model at T0
Volume of work
Vmax
V0
0
T0
T1
TA
Note: — Transition A: the transition of the stock in the storage model in response to the storaging events on simulator A
Vmin
TB
Note: — Transition S: the transition of the stock in the storage model
~~Running simulator B merging Log A~~
Note:
····· Transition A
— Transition B: the transition of the stock in the storage model in response to the shipping events on simulator B

Figure 4 Outline of the procedure of the proposed method (2)

Using these two above procedures, we propose a method of integration between simulators which do not have the rollback function for manufacturing system designs.

3 IMPLEMENTATION

In order to synchronize simulators, manager and simulator control modules are developed under the client-server environment.
The main role of the manager as the server is to command operations such as starting, and resuming for each simulator through a simulator control module in response to the simulation clocks and the storage model state. The manager behavior depends on a production management of a modeled manufacturing system such as push types and pull types. In the case of Figure 1, the manager behavior using the pull type is shown in Figure 5.

The main role of the simulator control module as the client, is to control a simulator directly in response to the manager commands. The simulator control module is located intermediately between the manager and each simulator. The proposed system under the client-server environment is shown in Figure 6.

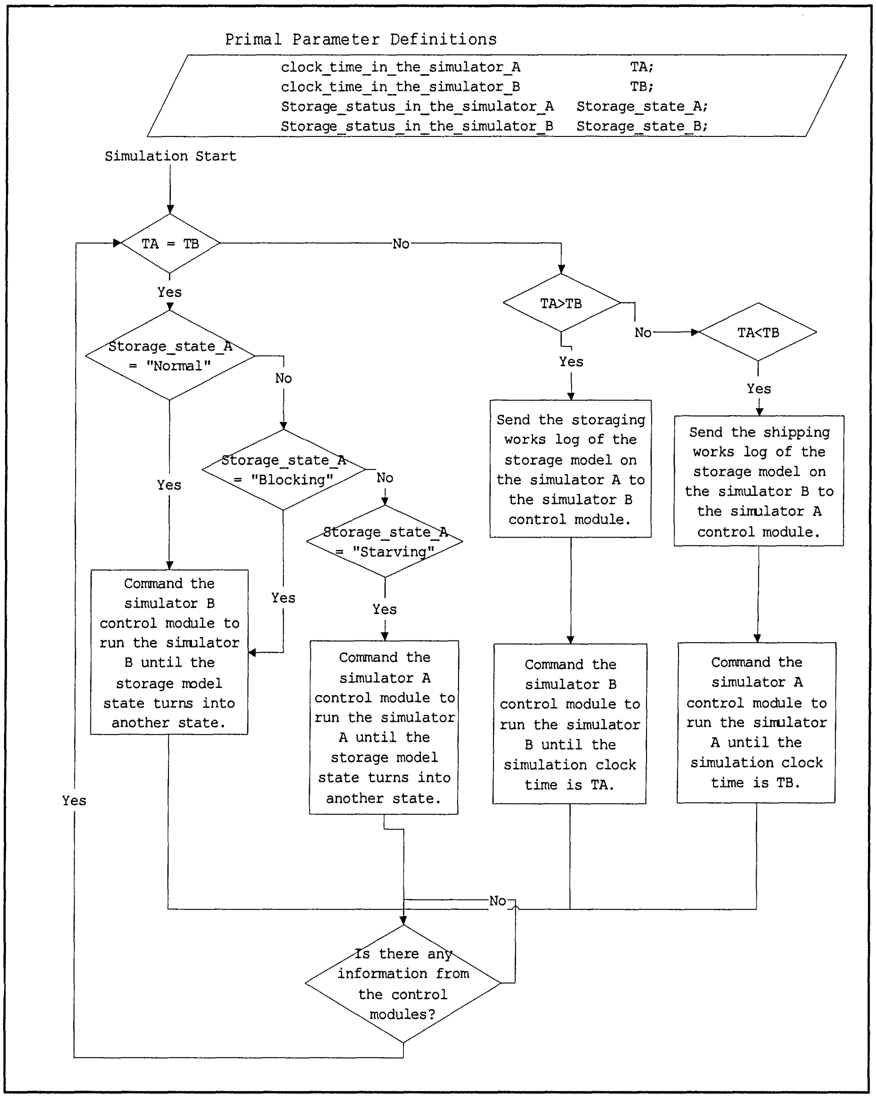

Primal Parameter Definitions
clock_time_in_the_simulator_A TA;
clock_time_in_the_simulator_B TB;
Storage_status_in_the_simulator_A Storage_state_A;
Storage_status_in_the_simulator_B Storage_state_B;
Simulation Start
TA = TB
No
Yes
TA>TB
No
TA<TB
Yes
Storage_state_A = "Normal"
No
Yes
Storage_state_A = "Blocking"
No
Storage_state_A = "Starving"
Yes
Send the storaging works log of the storage model on the simulator A to the simulator B control module.
Send the shipping works log of the storage model on the simulator B to the simulator A control module.
Command the simulator B control module to run the simulator B until the storage model state turns into another state.
Command the simulator A control module to run the simulator A until the storage model state turns into another state.
Command the simulator B control module to run the simulator B until the simulation clock time is TA.
Command the simulator A control module to run the simulator A until the simulation clock time is TB.
Is there any information from the control modules?

Figure 5 Manager behavior using the pull type

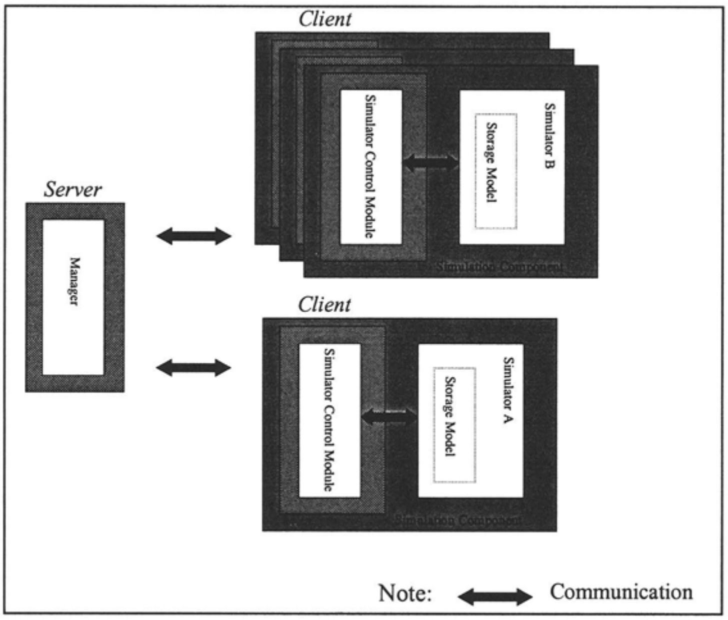


Figure 6 Proposed system under the client-server environment

The control module has principally eight functions which are classified into two groups.

Group1 : The functions to be used between the control module and the manager.

1. The function to get the commands such as starting, resuming and so on from the manager.
2. The function to inform about completion of the simulation by the indicated time to the manager.
3. The function to inform of changes in the storage model state to the manager.
4. The function to send and get the logs of storaging jobs or shipping jobs in the storage model.

Group2 : The functions to be used between the control module and a simulator.

5. The function to command the operations such as starting, resuming and so on for the simulator.
6. The function to stop the simulation when a change in the storage model states occurs.
7. The function to record events which occur in the storage model along with the simulation clock on a log file.
8. The function to create events in the storage model in response to the passed events using the log of the companion simulator on which the storage model is modeled. In the case of storage in the storage model, transactions are created and stored in the storage model. In the case of shipping the works from storage, transactions are shipped from the storage model and terminated.

As the manager and the simulation module are developed using our proposed methods, simulators can be synchronized without the rollback function.

4 A CASE STUDY

A case study in an automobile parts supplier was carried out to verify the proposed method.
The manufacturing system has the following features;

1. Flow shop type
2. Pull type production management
3. A machining line subsystem , two assembly line subsystems, a transfer subsystem using AGVs, and the storage unit

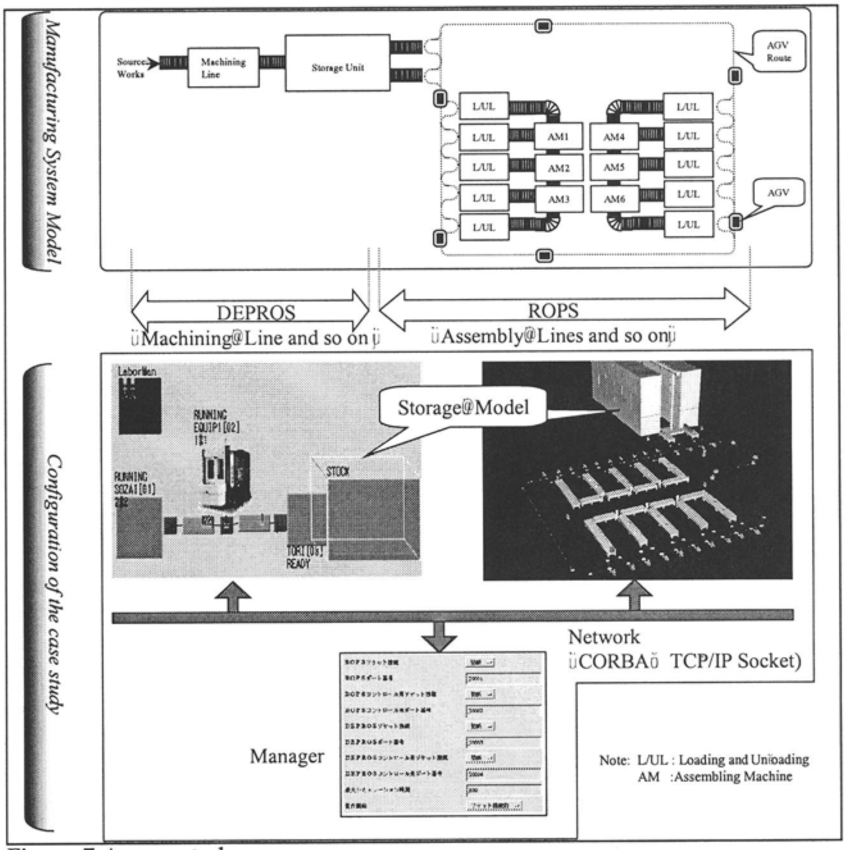


Figure 7 A case study

Two kinds of simulators were used for modeling the manufacturing system. The machining line subsystem was modeled on the DEPROS (Fukuda, 1994, Hibino,

1999, Kryssanov, 1998). The assembly line subsystems and the transfer subsystem were modeled on the ROPS (Nakano, 1994, Noritake, 1997). The storage unit was modeled as the storage model on both simulators. DEPROS is an object oriented simulator. ROPS is an agent type simulator. DEPROS and ROPS were implemented in the client systems. These client systems had a developed simulation control module. The client systems were connected with the server system which was implemented as the manager. The network systems to implement the client-server system used the CORBA protocol and the TCP/IP socket protocol .The outline of the case study model is shown in Figure 7.
The potential applicability for the developed model and client-server system was confirmed through this case study.

5 SUMMARY

We addressed a method of integration between simulators which do not have the rollback function using a storage model concept.
Our research showed:
1. A storage model concept for integration between the simulators which do not have the rollback function.
2. The development of a system configuration and its function as the storage model.
3. The implementation of the storage model using two different types of simulators.
4. Confirmation of the potential applicability for the storage model through a case study.

6 ACKNOWLEDGMENTS

This research is a part of the "IMS9810: MISSION; Modeling and Simulation Environments for Design, Planning, and Operation of Globally Distributed Enterprises" research project of the IMS program.
We thank Prof. Ronald S. Zavislak for his helpful comments.

7 REFERENCES

Fukuda Y, Hibino H, Mitsuyuki K, Kojima F, Ikeda Y, Aratake T, Tukada M, Suzuki K (1994) Integrated Environment for Production System Design Simulations: *Proceedings of Simulation for Manufacturing and Communications*, p434-438, Japan.

Fujii S, Kidani Y, Ogita A, Kaihara T (1999) Synchronization Mechanisms for Integration of Distributed Manufacturing Simulation Systems: *International Journal of Simulation*, **Vol. 71, No.3**, p187-197.

Fujimoto R (1990) Parallel Discrete Event Simulation: *Communication of ACM*, **Vol. 33, No.10**, p30-53.

Fujimoto R (1995) Parallel and Distribured Simulation: *Proceedings of the 1995 Winter Simulation Conference*, p118-125, U.S.

Hibino H, Fukuda Y, Fujii S, Kojima F, Mitsuyuki K, Yura Y (1999) The Development of an Object-Oriented Simulation System Based on the Thought Process of the Manufacturing System Design: *International Journal of Production Economics*, **Vol. 60-61** p343-351.

Jones K, Das S (1998) Combing Optimism Limiting Schemes in Time Warp Based Parallel simulations: *Proceedings of the 1998 Winter Simulation Conference*, p499-505, U.S.

Kryssanov V, Abramov V, Hibino H, Fukuda Y (1998) A Framework for the Development of Manufacturing Simulators: Towards New Generation of Simulation Systems: *Proceedings of 1998 Japan-U.S.A. Symposium on Flexible Automation*, p1307-1314, Japan.

Nakano M, Sugiura N, Tanaka M, Kuno T (1994) ROPS II: Agent-Oriented Manufacturing Simulator on the Basis of Robot Simulator: *Proceedings of 1994 Japan-U.S.A. Symposium on Flexible Automation*, p201-208, Japan.

Nicol D, Johnson M, Yoshimura A (1997) The IDES Framework: A Case Study in Development of a Parallel Discrete-event Simulation System: *Proceedings of the 1997 Winter Simulation Conference*, p93-99, U.S.

Noritake S, Tanaka M, Nakano M (1997) Manufacturing Simulation Language "ROPS" to Build Optimization Agents: *Proceedings of 14th International Conference on Production Research*□p270-273, Japan.