Korean J. Chem. Eng., **18**(6), 788-795 (2001)

Superstructure of yOTS-The Network-Based Chemical Process Operator Training System for Multiple Trainees

Seonyong Park, Sujin Lee and Il Moon†

Department of Chemical Engineering, Yonsei University, 134 Shinchon-dong, Seodaemun-gu, Seoul 120-749, Korea
(*Received 3 August 1999 • accepted 28 June 2001*)

Abstract—OTS (Operator Training System) is becoming popular for the safe and effective operation of chemical processes and control systems. This paper outlines the total hardware system superstructure and software modules of yOTS (Yonsei Operator Training System) which we developed. yOTS is a network based multi-training system composed of a workstation-based server module and PC-based user modules. The user module has a DCS-like user interface and sends data to OM (yOTS Manager) over the network. Reliability and stability are essential for the successful development of distributed OTS. *State-of-the-art* technologies of efficiency and stability are mainly considered in this paper. yOTS is superior to other OTS in its ease of handling discrete events, managing process models, expanding module functionality and multi-training over the network. The structure of yOTS and core algorithm for a multiple trainer over the network is also presented. A batch process example is used to illustrate the proposed advantages of yOTS.

Key words: Operator Training System, Network Based Multi-Training, Training Simulator, Virtual Reality, Chemical Processes

INTRODUCTION

The safe and efficient operation of chemical processes is becoming more important because chemical industries are more competitive and government safety regulations are becoming more strict [Dessonky, 1993; Park and Moon, 1994]. Major accidents in chemical industry such as fires, explosions and toxic chemical releases are mainly caused by unskilled operators and careless operations [Nicholas, 1995]. So systematically training plant operators is a main topic in the area of the process safety and operation. One effective way of training operators is by using a virtual reality simulator [Shier et al., 1996].

OTS includes a virtual environment similar to a real DCS system. Operators usually do not have many chances to handle emergency situations. As a result, using OTS would be the only effective solution to achieve those experiences [Kassianides, 1991]. Difficult or impossible situations in real systems can be generated to make operators acquire emergency handling experience. Using OTS is effective, cheap and sometimes the only way to develop excellent problem-solving skills.

OTS also can be used to study operating sequences, control strategies and operating conditions before and during constructing real plants [White, 1993]. Consequently, this technique makes it possible to understand process conditions more precisely, to improve the safety of the system, to acquire better operating techniques and to expand the lifetime of equipment.

In the viewpoint of management, a plant manager can realize the result of using OTS such as fewer and shorter shutdowns, improved product quality, lower maintenance costs and fewer emissions.

Implementing OTS requires the ability to handle both discrete and continuous dynamic behaviors such as pump on/off, valve open/close and discrete phase changes. It also requires fast computation, flexible interfaces among modules, efficient database for storing/retrieving emergency scenarios and the history of personal training, distributed system architecture, multiple execution, numerical solvers and psychological understanding of local operators.

Based on these techniques, we have been developing yOTS, which is better in handling discrete events, managing process models, multi-training over the network and evaluating the operator's capabilities. The network-based architecture allows multi-trainees' link and execution. These algorithms are mainly emphasized in this paper. Current yOTS can train about 10 persons or more simultaneously and allows users to access the system anywhere on the network by using a special interface.

GENERAL OPERATOR TRAINING SYSTEMS

1. General Architecture

A number of OTSs have been developed around the world and applied to numerous chemical processes [Baines, 1992; Wilmer, 1993; Baiagopal and Rafian-Naini, 1994]. The system architecture is various depending on developers, but we can classify them by three types such as DCS based, single computer based, and network based system [Stawarz and Sowerby, 1995]. DCS based type is very expensive and single computer based type requires low investment cost but produces poor effects. Educational efficiency and economies are major constraints in adopting OTS to ones plant. Any type of these solver modules and user modules is a necessity to configure OTS.

Process modeling and computation methods are usually main

†To whom correspondence should be addressed.
E-mail: ilmoon@yonsei.ac.kr

hindrance to be overcome by developers [Marquardt, 1991]. How accurately is a process to be modeled and what kinds of solvers are to be included in the system wholly depend on educational objectives, economy, and hardware architecture. Most chemical processes are described by DAEs (Differential Algebraic Equations). Since general chemical processes are highly nonlinear, such equations are sometimes replaced by linear equations using linearization such as power series to speed up their computation. Selecting solving strategies requires considering the size of process and hardware performances. The well-designed superstructure of OTS increases total training performance, and numerical computing time is usually a critical constraint in real time training of large-scale processes. The selected solver is generally based on a self-developed library or common library from Netlib over the Internet or commercial dynamic simulators for chemical processes. To improve the numerical computation, we have developed yOTS with a parallel and distributed architecture.

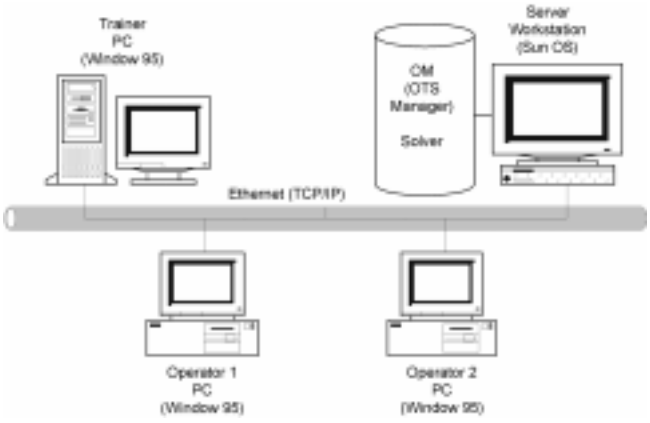

Fig. 1. Current network configuration of yOTS (Yonsei Operator Training System).

yOTS (YONSEI OPERATOR TRAINING SYSTEM)

yOTS (Yonsei Operator Training System) is a network-based system including one workstation and three PCs. Fig. 1 shows the hardware and software network configuration including two operator's PCs, one trainer's PC and one workstation which does most of the computation. One more server workstation could be added depending on the process complexity and computation time

1. The Modules of yOTS

yOTS is composed of three main modules - OTS manager module, solver module and user modules. Modules except the solver have a great impact on the performance of the total system. Numerous functions are implemented in these modules.

1-1. OM (Operator training system Manager) Module

OM supplies various functions to operators and trainers, including overall process execution, stop, operation, problem description, control of training situations, scenario execution and scenarios construction. OM also analyzes the operator's task progress and current status of operating maturity. This module also connects user and solver modules and transmits data among them. OM checks interfaces among all modules at the first execution and initializes all system variables at the trainer's connecting.

1-2. Solver Module

Basically, OTS must be able to simulate various process conditions. It must be able to handle discrete events such as valve/pump on and off, phase change, startup and shutdown of continuous and batch processes. Fast computation is very important for the process real-time simulation. In this system, we used a general purpose dynamic simulator [Ready, 1992; Barton, 1994; Oh and Pantelides, 1996], gPROMS (General PROcess Modeling and Simulation), developed by Imperial College. gPROMS has advantages of easily describing process models and dealing with process discrete events.

1-3. User Module

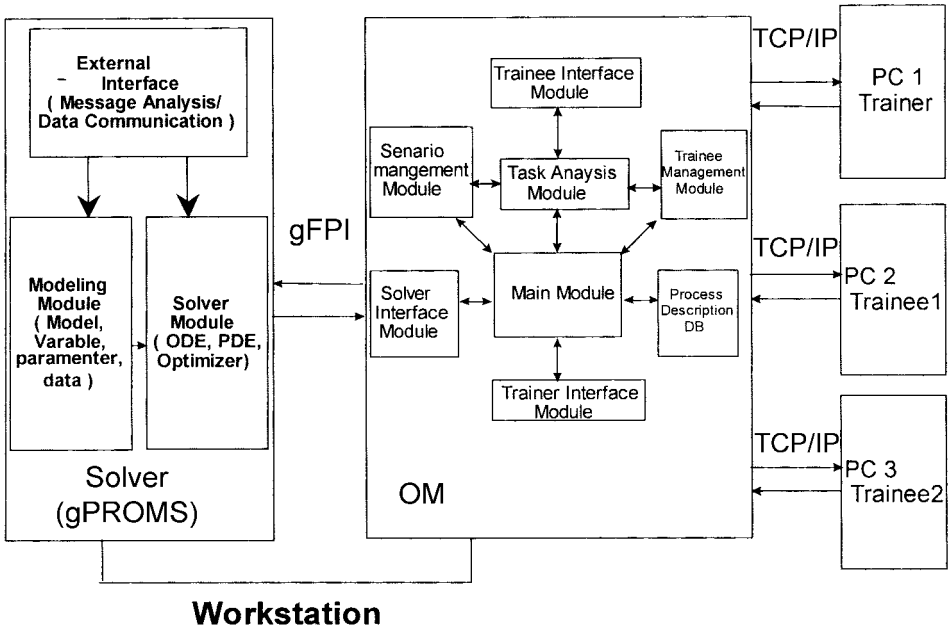

Fig. 2. Superstructure of yOTS.

For efficient education, OTS offers an operating environment similar to real DCS. yOTS is based on a GUI (Graphic User Interface) [Guillermo and Mehdi, 1992] environment which is almost the same as DCS. The trainer's module manages the graphic screen, adjusts overall processes and supervises operator's training scenarios.

The superstructure of yOTS is schematically presented in Fig. 2. A dynamic simulator and thermodynamic database compute most process dynamic behaviors. Network communication and gFPI (gPROMS Foreign Process Interface) are used to transmit calculated dynamic simulation data between OM and the solver.

OM is driven as a parent process that generates a child process and drives the solver. The data structure is the core of the interface between two modules and overall system architecture. Two methods for the data communication are used, using file and shared memory. The former takes time to open and record data so it makes a time delay. Using shared memory for process data communication takes less time, but it is hardware dependent. Our OM uses shared memory in its default.

The data transmitted from OM to a solver are simulation status, simulation time increment, unit number, task number and task value. The concrete data of simulation status are start, pause, restart, stop and continue simulation as user's request. The simulation time increment means the step size of dynamic simulation time. The unit number is a predefined task value that an operator executes. The examples of predefined tasks are valve open/close, pump on/off, controller open/close and set point reinitialization. All data of current simulating time, model status and process variable values calculated by a simulator are transferred from a solver to OM.

The solver module offers an interface method for the connection of external user modules. The FPI is designed to call functions internally. The major functions of FPI are START, PAUSE, GET, SEND and END. With these functions, OM connects the solver module. OM is a main server module in OTS system and solver carries out the client's request over OM. OM generates shared memory in which OM reads and writes the data and creates a semaphore which is used to control access to system resources. If N operators take training, the interface generates N shared memory and 2 N semaphore.

2. The Developing Environment of Current yOTS

Table 1 describes the current developing environment of the three main modules of yOTS. OM is developed in the UNIX environment and using C and C++. User module is developed in Win95 and using several visual tools and graphic tools. The Solver gPROMS is executed under the UNIX environment. Visual tools offer the developer an ease of developing functions and graphic interface of the user training environment. Generally speaking, it is the UNIX environment that is more reliable than other environments on multi-user stability and system security.

Table 2. The main header files of OM

Module	Function
GlobalDef.h	This contains Task No. Simulation Status No., Task Value, Unit No., Communication Data Structures, etc.
MessageData.h	This contains message data function for communication between OM and Solver.
Sema.h	This contains general function for semaphores of shared memories used in the communication between OM and Solver.
Shm.h	This contains shared memory managing libraries.
SockDef.h	This contains socket libraries for communication between OM and Operator/Educator.
SolverInterface.h	This contains solver interface libraries for the communication between OM and Solver.

3. Network Based Main Algorithm of OM for Multiple Trainees

OM (OTS Manager) module is the crucial part of OTS and this manages overall systems by connecting all modules. OM can accept multi-trainee's and trainer(educator)'s connecting and initialize processes. In Table 2, we describe main software headers of OM and their functions, which are developed by c++, c and FORTRAN.

GlobalDef contains various predefined numbers using Overall yOTS modules such as task number, simulation status, task value, unit number, data structure for a communication between modules. Following is the example of GlobalDef.h. Predefined values offer developers great advantages - the minimizing of communication data, robust operation handling and downsizing other modules. But we should predefine these data well lest we change this system structure later.

In the initial state, OM identifies overall system, then receives user's input. If computation is required, OM connects a solver, checks inputs as time varies and executes recalculations. OM is located at the server computer, initially started by the trainer and it can execute and quit simulations.

Table 1. Current development environment of yOTS

	OM module	User module	Solver module
OS	UNIX	Win95	UNIX
Developing	C/C++	Visual C++	gPROMS
Tool	UNIX tools	Visual Basic	
		Graphic S/W	
Function	Module connection	Graphic based user interface	Discrete event handling
	User Input analysis and execution	Connection to OM	DAEs PDAEs solving
	Automatic database initialization	Changing user input to predefined value	Optimization
	Managing total	Managing Trainer's state (**Trainee module only**)	
	Process (solver, background processing)	Evaluating training result (**Trainee module only**)	

Table 3. An example of the definition in GlobalDef

# TASK definition			
const NO_TASK	1000		
const NOT_UNIT_TASK	1100	# Task Value definition	
const TASK_OPEN	1110	**const** Conct_Index1	3010
const TASK_CLOSE	1120	**const** Conct_Index2	3020
const TASK_ON	1130	**const** Conct_Index3	3030
const TASK_OFF	1140	...	
const TASK_CONTINUE	1150	# unit definition	
const TASK_SWITCH	1160	**const** NO_UNIT	10100
...		# FeedTank	
# Process Status definition		**const** T10101	10101
const SIMULATION_START	2011	**const** T10102	10102
const SIMULATION_PAUSE	2012	...	
const SIMULATION_RESTART	2013	**const** P10102	10202
const SIMULATION_STOP	2014	....	
const SIMULATION_BACK	2015		

More detailed explanation of the OM algorithm is the following:

3-1. Initializing Total Systems

OM checks whether each module exists and interfaces among modules are set up well. Generally speaking, processing has a tendency to create a zombie process so we should prevent this from occurring by the initial system checking.

3-2. Waiting User's Connecting

OM is a server process working in a workstation. So it can be worked by background processing and has to wait till a user connects. OM always checks the connecting sate of a trainer and a trainee because OM is designed for multiple operators.

3-3. Selecting a Training Session and its Contents

If step 2 is fully performed, OM waits for operator's input and makes him/her choose the training session and its contents. Then it reads required process models and data from database to initialize all the system variables. Usually the system resources - process models, variables, shared memories, semaphores and database storage space - are very huge. So OM must create those resources only when it is required.

3-4. Waiting for Messages

If a process requires simulations, OM waits for messages from a trainer or a trainee. After receiving messages, OM analyzes them whether they are process operations or something else global defined variables.

3-5. Handling Process Operation

OM analyzes whether the messages are operation-related tasks such as valve open and pump off or simulation-related tasks such as time reset, execution, temporarily quitting and restarting of the simulation. If it is an operation-related task, OM sends data to the solver.

3-6. Waiting Calculated Data

OM waits the calculated data. OM receives these data, then sends the results of the computation to trainer and trainee's modules.

Fig. 4 shows one of yOTS process screens that is a distillation column and combined subsystem.

4. The Connectivity and Data Flow of yOTS

If a user module program is in a PC-connected network, the user can connect the OM and can train the course with or without a trainee. The OM server always checks the user requests and creates new trainer resources, selecting/changing training session, changing process status, loading/exchanging object process, etc. The connectivity and data flow in executing of yOTS are illustrated in Fig. 5.

```
Start
  Initialize all the system variables and resources

  WHILE user input
    Check for the trainer's login
    Check for the operator's login
    IF n(Trainer login)
      create process related resources for n(The number of Trainer)
      (such as shred memory, process data, initializing solver...)
      Loading n(selected process and training session resources)
    Else IF Trainee login
      create resources for Trainee
      Loading selected process and training session resources
    EndIF

    IF Trainee/Trainer Data
      IF related to process then
        Changing process vars
        Write changes to the shared memory for sovler
      else IF only state vars
          IF not(end of process)
              Changing yOTS state
          else IF end of process
              Save educational, systematical information to database
              Terminate the solver
          EndIF
      EndIF
    EndIF

    IF n(caculated data from sovler)
      Data transfer to Trainer/Trainee module
    EndIF

  EndWHILE

  Remove all resources
  End
```

Fig. 3. Pseudo code of the main algorithm.

5. Training Hierarchy

The training procedure using yOTS includes a process description session, a normal operating session and fault diagnosis and treatment session. A beginner starts the process description session to understand the overall processes, equipment. A trainee can learn the normal operating procedure in the second session. At the last session, an operator recovers the process status from the emergency situation. Through these three sessions an operator can practice repeatedly until familiar with the process and can develop excellent problem-solving skills.

5-1. Process Description Session

This session explains the overall process description at each stage, shows pictures of equipment and illustrates theoretical physico-chemical background and simple actions. This helps operators to understand causes and effects of subsystems by illustrating examples. Con-

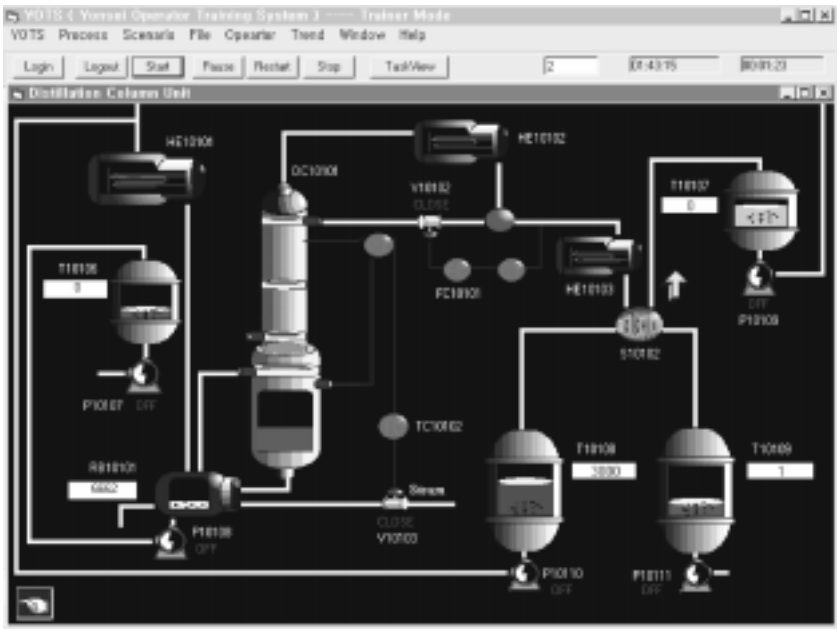

Fig. 4. An example of a screenshot of yOTS.

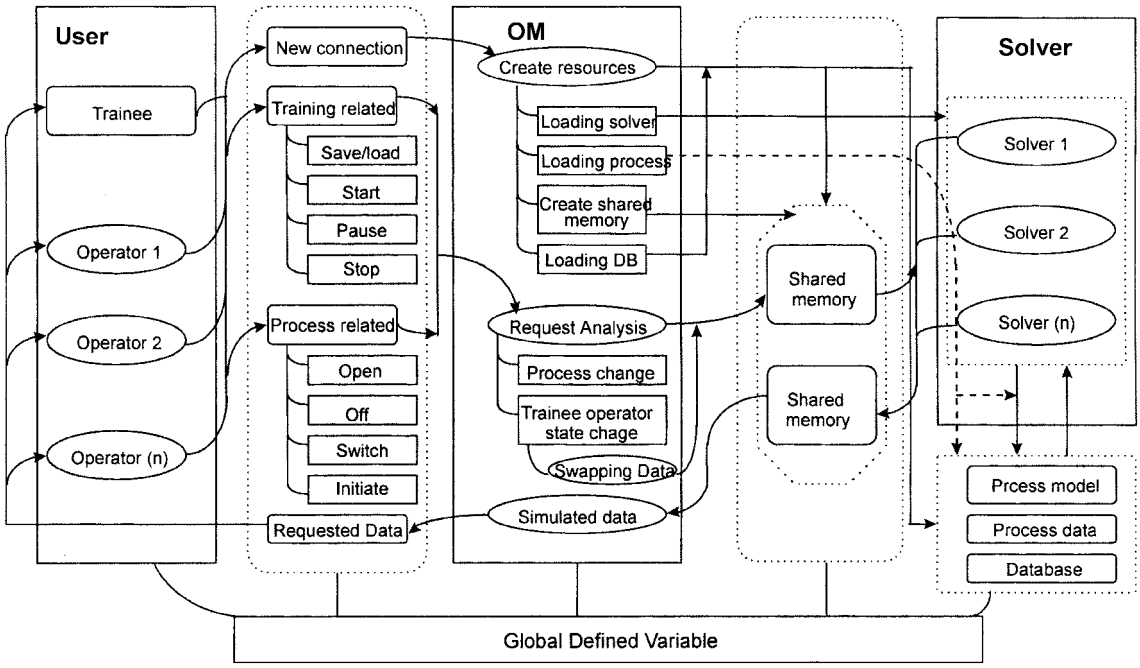

Fig. 5. The connectivity and data flow of yOTS.

sequently, OTS makes it easy for beginners to understand the process by showing dynamic pictures and easy explanations.

5-2. Normal Operating Session

Operators learn normal operating procedures and their work to do in this session. They operate DCS by using a real-time simulator. As a result, the operator acquires system-handling techniques of startup and shutdown procedures. By practicing normal operations using OTS, a beginner can overcome the fear of handling real DCS systems. Normal operation training reduces a great deal of training time.

5-3. Fault Diagnosis and Treatment Session

In this stage, an operator practices the sequences of handling the abnormal state by detecting, analyzing, correcting and recovering errors. To supply emergency scenarios, the following two methods are used: trainer's direct interruption and indirect interruption by prepared scenarios. If a process is selected as a current state among initial startup, final shutdown and steady state stage, the trainer interrupts the operator's training and executes malfunctions, which might happen in real plant. So an operator can practice the sequence of coping with this unexpected dangerous situation. This stage is

Table 4. Education using yOTS

	Process description session	Normal operating session	Fault diagnosis and treatment session
Function	• Parts of processes • Overall tutoring of the total process	• Normal process operating • Start-up • Shutdown	• Process upset detecting • Treatment session
Method	• 3D graphic • Multimedia • Operating procedure	• Normal operating • Process dynamic results	• Process malfunction initialization • React the operation of operators
Target processes	• Equipment • Subsystem • Total processes	• Subsystem • Total process	• Total process
Education objective	• Mastering fundamental process knowledge • Process physico-chemical properties	• Normal operation • The cause and effect of process • Dynamic process characteristic	• Detection • Diagnosis • Recovery • Coping with an emergency

the most important for the efficient education using OTS.

THE FUTURAMA OF THE DEVELOPMENT OF yOTS

Safety standards and laws are becoming stricter; as a consequence, all companies will need to improve safety methodologies. The transition from simple simulation of modeled processes to vast and greatly realistic virtual environments will take place in the next 10 years. We inquire into the future of OTS with our developing yOTS as the central figure.

1. ITS and IES

ITS (Integrated Training System) which is the integrated system type having the other subsystem based on simulation with trainer/trainee module, can offer engineers and operators the control strategy study, safety control and operation, researching of operation sequence, control tuning, process scheduling. A batch manager including dynamic scheduler and safety analysis module is included in ITS. This system can be used in the area of process operation, scheduling, and operating optimization. Dynamic scheduler, object-oriented database, scenario manager and safety analyzer are essential parts of ITS. Also, we propose this architecture as Fig. 5. This is composed of three main parts: yOTS, batch management system, and safety analysis system. This enables us to decide the process operation strategy in an abnormal situation with the cooperation of batch management module and OTS module. Safety analyzer can evaluate the process operation sequences whether it is going to the dangerous situation over the symbolic operation checking.

IES (Integrated Environment System) is the future system with a real world environment of a plant. In a virtual plant, an operator or engineer can see the total process, operate the process and can react to process upsets such as fires, explosions, and environmental polluting. This type requires virtual tools, case by case situation, cyber environment and holographic environment.

The final form of this system is a virtual plant including process and plant environment. Fire, explosion, toxic material release and vast disaster could be executed and users could respond to this situation. Ideas represented by us are a type that uses the cyber space and hologram - *The virtual environment*.

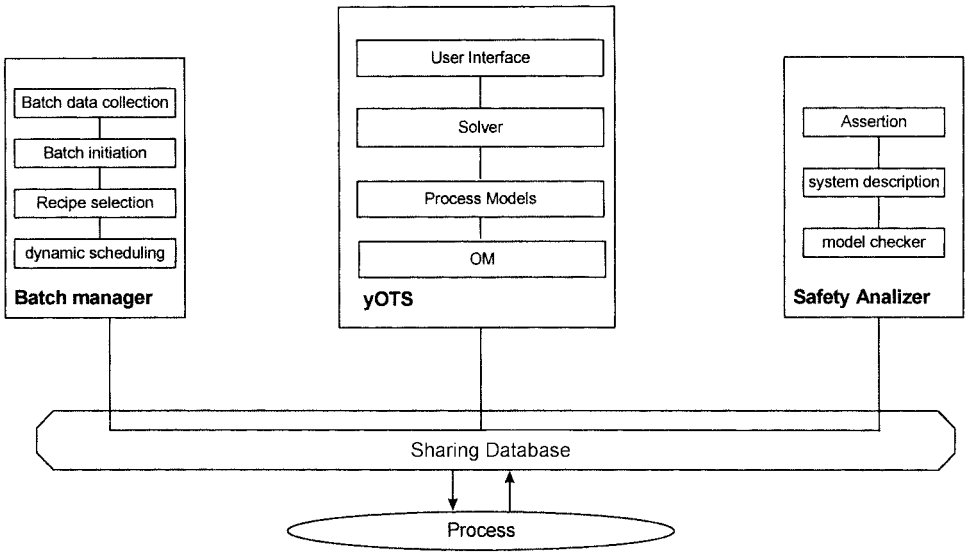

Fig. 6. The ITS architecture we propose.

2. Distributed and Parallel Computation

A huge computing capability is demanded in the construction of virtual plants. Image processing and 3D virtual image formation as well as numerical computation are necessary to the development of IES. A single server like one workstation cannot support these fully. Distributed and parallel computing is one of the answers. We are expanding our server system to the distributed and parallel server - flexible calculation, tearing process, independent calculating of physical property, image processing. A well-designed server will cut down cost as a result of the replacement from workstation-based server to PC-based server. For the purpose of applying to yOTS, the technologies of local area multi computing, message-passing programming, distributed shared memory system on the network of workstations and parallel programming are being studied in our laboratory.

3. Free from Hardware Capability

Keeping pace with the advances of computer science, training packages will no more be hardware dependent. Operating systems will be replaced by the network. Solver and other modules will exist somewhere on the network. Some of the training packages designed with these hardware-free languages are independent from trainer computers. This can help whole world users to connect to the state-of-the-art systems easily and to come in contact with various types of developed systems. This will be an advantage to the system manager in developing as well as improving a developed system. Distributed and parallel computing will be no more necessary in the future network support cyber space. The only thing needed is network and system.

CONCLUSION

OTS enables operators to have confidence in operating plants by removing fears in handling brand new equipment and by supplying similar circumstances with DCS-based real plants. It also gives operators chances of handling emergency situations that are difficult or impossible to practice in real plants. As a result, this improves an operator's ability to manage emergency situations and find effective operating solution strategies. Additional advantages of using OTS include finding an optimal operating sequence, optimal control path and safe operating procedures. To acquire these advantages, we have developed yOTS, and its structure and main algorithms are illustrated in this paper. yOTS is superior to other OTS in its ease of handling discrete events, managing process models, expanding module functionality and multi-training over the network. Proposed superstructure and modules are well defined and can be easily expanded to the next version of OTS such as ITS and IES. Other modules like a dynamic scheduler, scenario manager and database are being developed. Main algorithms of OM for more effective module communication are also presented. This algorithm is designed for multi-trainees and is very robust for the training simulator. The future of OTS - development-ITS, IES and parallel-distributed computing is also delineated. yOTS will be improved continuously in its efficiency and correctness.

ACKNOWLEDGEMENT

This work was supported in part by the Korea Science and Engineering Foundation (KOSEF) through the Automation Research Center at Pohang University of Science and Technology.

REFERENCES

Anderson, V., "Customize Process Simulation Training," *Computers Chem. Engng.* **Feb.**, 141 (1990).

Baiagopal, S. and Rafian-Naini, M., "Workstation Based and Advanced Operator Training Simulator for Consolidated Edison," *IEEE Transactions on Power Systems*, **9**(4), 1980 (1994).

Baines, G. H., "Benefit for using a High-fidelity Process Simulation for Operator Training and Control Checkout," *Tappi Journal*, **Jan.**, 133 (1992).

Barton, P. I., "The Modeling and Simulation of Combined Discrete/Continuous Process," Ph.D. Thesis, Imperial College (1994).

Choi, Y. J., Kwon, T. I. and Yeo, Y. K., "Optimization of the Sulfolane Extraction Plant Based on Modeling and Simulation," *Korean J. Chem. Eng.*, **17**, 712 (2000).

Dessonky, Y. M., "An Object Oriented Architecture for Combined Simulation Modeling with Real-time Capabilities for Operator Training," Ph.D. Thesis, Arizona state Univ. (1993).

Gottmann, O. and Holl, P., *Auto. Tech. Prax* (*Germany*), **38**(12), 11 (1996).

Grossmann, I. E., Caballero, J. A. and Yeomans, H., "Mathematical Programming Approaches to the Synthesis of Chemical Process Systems," *Korean J. Chem. Eng.*, **16**, 407 (1999).

Guillermo, D. I. and Mehdi, R., "Heuristic Scenario Builder for Power System Operator Training," *Proceedings of the IEEE*, **80**(5), 698 (1992).

Han, J. R., Manousiouthakis, V. and Choi, S. H., "Global Optimization of Chemical Processes Using the Interval Analysis," *Korean J. Chem. Eng.*, **14**, 270 (1997).

Himmelblau, D. M., "Applications of Artificial Neural Networks in Chemical Engineering," *Korean J. Chem. Eng.*, **17**, 373 (2000).

Kassianides, S. C., "An Integrated System for Computer Based Training of Process Operators," Ph.D. Thesis, Imperial College (1991).

Laganer, F., "Dynamic Process Simulation Trends and Perspective in an Industrial Context," *Computers Chem. Engng*, **20**, suppl., 1595 (1996).

Marquardt, W., "Dynamic Process Simulation Recent Progress and Future Challenges," Proceedings CPC IV, Texas, 45 (1991).

McKenty, F., Gravel, L. and Camarero, R., "Numerical Simulation of Industrial Boilers," *Korean J. Chem. Eng.*, **16**, 482 (1999).

Morgan, S. W., "Improve Process Training with Dynamic Simulation," *Hydrocarbon Processing*, **Apr.**, 51 (1994).

Nicholas, B., "A Virtual World for Operator Training," *Chem. Eng. Prog.*, **102**(5), 135 (1995).

Oh, M. and Moon I., "Framework of Dynamic Simulation for Complex Chemical Processes," *Korean J. Chem. Eng.*, **15**, 231 (1998).

Oh, M. and Pantelides, C. C., "A Modelling and Simulation Language for Combined Lumped and Distributed Parameter Systems," *Computers chem. Engng*, **20**(6/7), 611 (1996).

Park, S. Y. and Moon, I., "Development of an Operator Training System for Batch Chemical Processes," Prepared for presentation at the AIChE Annual Meeting/Chicago, Annual meeting, Nov. 10-15 (1996).

Park, S. Y. and Moon, I., "Characteristic and Development of Operator

Training System," *Chemical Industry and Technology*, **12**(6), 22 (1994).
Ready, L. A., "Programming Operator Interfaces with Visual Basic," *Control Eng*., **Feb**., 23 (1993).
Sargent, R. W. H., "Systems of Differential-Algebraic Equations," Technical report, Imperial College (1992).
Shier, W., Kennett, R. and Vclav, E., Proceedings of ESS 96. 8th European Simulation Symposium, **1**, 24 (1996).
Stawarz, A. C. and Sowerby B., "Cost Effective Operator Training," *Computers chem. Engng*, **19**, 459 (1995).
Watzdorf, R. V., "Dynamic Modeling and Simulation of Batch Processes," Technical report, Imperial College (1992).
White, J. R., "The Operator is the Process Manager," *Control Eng*., **Jan**., 67 (1993).
Wilmer, T. J., "Quality Management Certification for the Nuclear Industry," *Nucl. Eng*. (UK), **34**(5), 153 (1993).