Proceedings of the 14th
KACC, Octorber 1999

Operation of a Networked Virtual Manufacturing System using Quasi-Procedural Method

°Noh, Sangdo*, Sheen, Dongmok*, Hahn, Hyung Sang*, Lee, Kyoil**

* Institute for Advanced Engineering(Tel:+81-0335-330-7604; Fax:+81-0335-330-7116; E-mail:nsd@iae.re.kr)
** Seoul National University(Tel:+81-02-880-7135; Fax:81-02-888-4182; E-mail:lki@plaza.snu.ac.kr)

Abstract : Nowadays, one of the major technical issues in manufacturing is to create an environment to promote collaboration among diverse engineering activities. Collaborative engineering is an innovative approach integrating widely distributed engineering activities through promoting information sharing and actual collaboration. It requires close interactions among developers, suppliers and customers, and consideration of entire product life cycle from concept to disposal. A carefully-designed operating system is crucial for successful collaboration of many different activities in a Networked Virtual Manufacturing System(NVMS). High extensibility, flexibility and efficiency are the key characteristics requested of an operating system to handle the complexity of the NVMSs. In this paper, we propose a model of the operating system for collaborative engineering using concurrent quasi-procedural method(QPM). QPM is a goal-driven data management technique for distributed and parallel computing environments. It is to be applied to the evaluation of activities to be executed, validities of input data, execution path of activities for a needed output, and expected to greatly improve the productivity of operations by preventing redundant evaluations. Collaboration among many different engineering activities in NVMSs is to be performed by the network of agents that encapsulate the capabilities of both users and their tools.

Keywords : Collaboration, Engineering, Network, Virtual, Manufacturing

1. Introduction

Nowadays, one of the major technical issues in manufacturing is to create an environment to promote collaboration among many different engineering activities. Collaborative engineering is an innovative approach integrating widely distributed activities through promoting information sharing and actual cooperation. It requires close interactions among developers, suppliers and customers, and considerations of entire product life cycle from concept to disposal.

Networked Virtual Manufacturing System(NVMS) is a new architecture to integrate diverse engineering and manufacturing activities distributed geographically. It is a precise computer model distributed on many computers in a network, which represents physical and logical behavior of a real manufacturing system. A carefully-designed operating system is crucial to promote collaboration among many different activities in NVMSs. High extensibility, flexibility and efficiency are the key characteristics needed of an operating system to successfully handle the complexity of the NVMSs.

1.1 Collaborative Engineering

Today, the market environment is characterized by frequent change of customers demands, rapid technological progress and shorter product life cycle, and thus requires a new engineering strategy incorporating consideration of the total product life cycle. Concurrent engineering has been recognized as the most powerful strategy with such property. One of the key-words in current research in concurrent engineering is *Co-operation*[8][9]. Co-operation here means *Collaboration,* and is very important in today's engineering, as systems become larger and more complex. *Collaborative Engineering* means *Co-operative works* sharing information and knowledge of engineers of many companies globally distributed. It must cope with all the collaborative activities among engineers concerned in various phases of the entire product life cycle. In Collaborative Engineering, participation of all the related parties from system developers to customers, and considerations on the entire product life-cycle, including user requirements, quality, cost, and schedule are needed[8].

Collaborative engineering has become more important as manufacturing activities need higher expertise and involvement of many people on networks, including design engineers, production managers, process planners, production engineers, delivery managers, customers and expert advisors.

1.2 Networked Virtual Manufacturing System

The integration of all the engineering and production activities becomes one of the most important technical issues in today's manufacturing. A common united model which covers all the engineering functions, information flow and precise behavior of manufacturing system is needed[3][4] to realize integration.

NVMS is a precise computer model distributed on several computers in a network, which represents physical and logical behavior of a real manufacturing system. It can integrate diverse manufacturing activities and processes distributed geometrically using many new technologies in computer networks and databases[6][7]. The concept of NVMS, along with the concepts of *integration* and *collaboration*, promote sharing of information and knowledge among all the engineering and manufacturing activities. Figure 1. shows the structure of an NVMS[10].

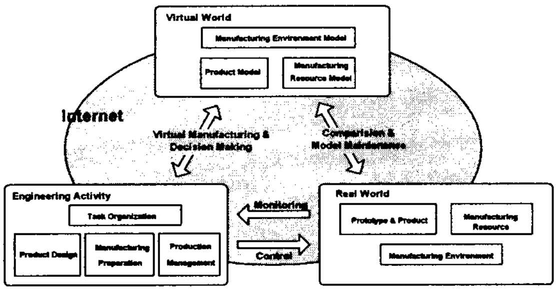
Virtual World
Manufacturing Environment Model
Product Model
Manufacturing Resource Model
Internet
Virtual Manufacturing & Decision Making
Comparision & Model Maintenance
Engineering Activity
Task Organization
Product Design
Manufacturing Preparation
Production Management
Monitoring
Control
Real World
Prototype & Product
Manufacturing Resource
Manufacturing Environment

Figure 1. The Structure of a Networked VM System

1.3 Quasi-Procedural Method

Quasi-procedural method is a data management programming method for large S/W systems having many sub-modules with complex relations[1]. For such S/W systems to successfully work, extensibility, flexibility and efficiency are required. Most conventional S/W systems use a procedural method, which manages sub-modules according to sequential relations and whose path is static. Thus, they are not free from the problem of low extensibility, flexibility and efficiency. In Figure 2, suppose that the user wants to know *Needed Output* when *Input A* is changed. All the black sub-modules must be processed together to acquire the *Needed Output*. In this example, 13 sub-modules must be processed. It will be more serious in optimization problems where an objective module and multiple constraints should be evaluated many times by changing the value of input data.

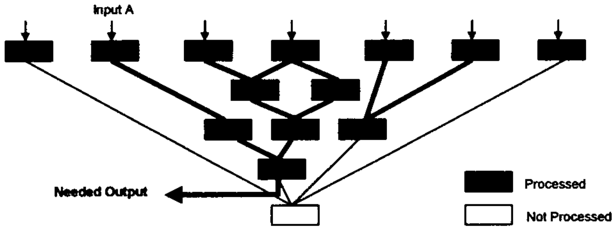

Figure 2. Example Case of the Procedural Method

The name, *quasi-procedural* signifies that the method combines both procedural and non-procedural approaches[1]. Unlike a procedural method, one doesn't have to call other sub-modules explicitly from a sub-module in the quasi-procedural method. When a sub-module needs data to be processed by another sub-module, it can call a built-in procedure which call the necessary sub-modules in the appropriate order and returns the computed result. This architecture provides the extensibility and the flexibility lacking in the procedural method, and enables the user to run the process to obtain the desired output without detailed knowledge about all the sub-modules and their interactions. In the quasi-procedural approach, all the modules are presented in accordance with the order of processing[1][5] as in the procedural approach.

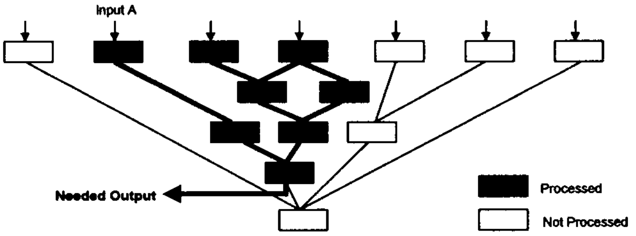

Figure 3. Example Case of the Quasi-procedural Method

Figure 3. shows an example case of a quasi-procedural method. In this case, 8 sub-modules must be processed to acquire *Needed Output*. It should be noted that the recognized processing path does not contain any other sub-modules irrelevant to process the desired output. There is another important characteristic that tells the quasi-procedural method from both procedural and non-procedural methods, *consistency maintenance*. *Consistency maintenance* is a routine by which the system determines when to re-process which variables in response to the changes of other data. (see Figure 4.)

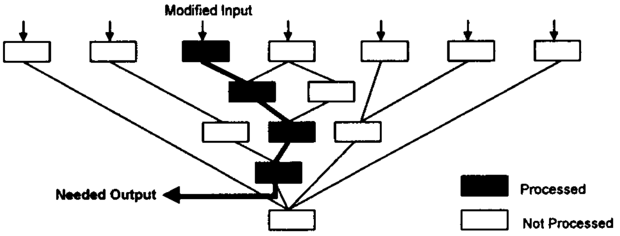

Figure 4. Consistency Maintenance

2. Concurrent Quasi-Procedural Method

2.1 Quasi-Procedural Method

In the quasi-procedural method, two special functions, GET and PUT, are used for interfacing between the modules and database. The GET function is called to load a required input into local data. Similarly, PUT is called to save a result to the database. The argument on GET and PUT functions is locally declared data to which a value is assigned in the database. Many data can be tentatively specified by some sub-modules during executions of sub-modules. In this case, another function, PUSH is needed. PUSH saves specified data while computed data is saved by PUT. Basic procedures of GET and PUSH functions in the quasi-procedural method are shown in Figure 5.[1][5].

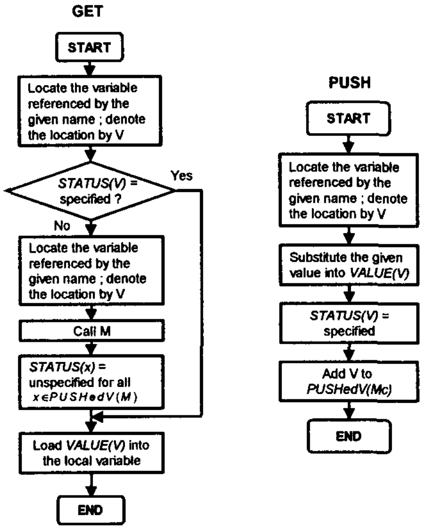
GET
START
Locate the variable referenced by the given name ; denote the location by V
STATUS(V) = specified ?
Yes
No
Locate the variable referenced by the given name ; denote the location by V
Call M
STATUS(x) = unspecified for all x∈PUSHedV(M)
Load VALUE(V) into the local variable
END
PUSH
START
Locate the variable referenced by the given name ; denote the location by V
Substitute the given value into VALUE(V)
STATUS(V) = specified
Add V to PUSHedV(Mc)
END

Figure 5. Basic Procedures of GET and PUSH

By appending a consistency propagation algorithm to PUSH function and a Ω-generation algorithm to GET function, the quasi procedural method is completed. The final procedures of the quasi-procedural method are shown in Figure 6.[1].

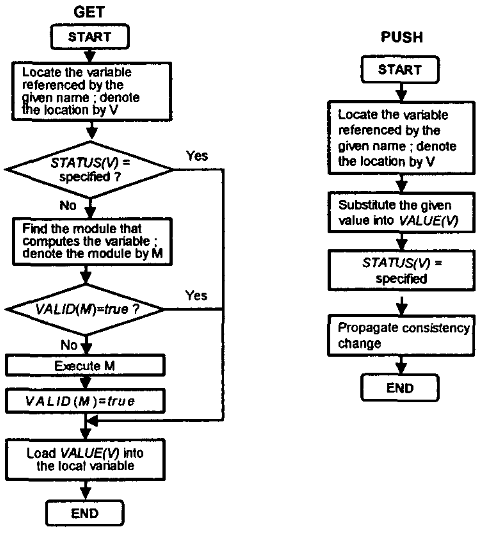
GET
START
Locate the variable referenced by the given name ; denote the location by V
STATUS(V) = specified ?
Yes
No
Find the module that computes the variable ; denote the module by M
VALID(M)=true ?
Yes
No
Execute M
VALID(M)=true
Load VALUE(V) into the local variable
END
PUSH
START
Locate the variable referenced by the given name ; denote the location by V
Substitute the given value into VALUE(V)
STATUS(V) = specified
Propagate consistency change
END

Figure 6. Procedures of GET, PUSH with Consistency Maintenance

2.2 Concurrent Quasi-Procedural Method

In order to apply the quasi-procedural method to the execution of multi-functional users distributed, the *Concurrent Quasi-Procedural Method* is developed. *Concurrency* is a characteristic associated with the use in distributed and parallel processing in multi-tasking environments. The hardest part in implementing the quasi-procedural method for parallel processing is the identification and scheduling of the executable modules concurrently. Since the quasi-procedural method is capable of linking a module to the execution path by the reversed order, identification and scheduling problems concerned with the parallel processing become important.

Capability of concurrent quasi-procedural method developed in this paper includes[10] :

- Adaptation to the distributed processes
- Scheduling of modules
- Multi-functional users
- Database Integration
- Script Property

The first step of implementing the concurrent quasi-procedural method is to build a user management function into quasi-procedural method. To do this, a *user variable* is added to the module information and functions to add the concept of user to each module. The second step is distributed module execution. It means module execution can be performed by multi-users in parallel. To solve this problem, the *Mark* function is introduced. In the concurrent quasi-procedural method, modules of different users are just marked instead of actual execution itself, and the execution of those modules will be done only by a proper user. In the concurrent quasi-procedural method, a new function CHECK is developed. It uses a specified user as an input and checks whether execution is possible or not by the validity of the module of users, the contents of MARK array and REQUESTERS of the module and the user. VALID is re-defined to handle the validity of multi-users for each sub-module. The procedures of a concurrent quasi-procedural method developed in this paper are shown in Figure 7.[10].

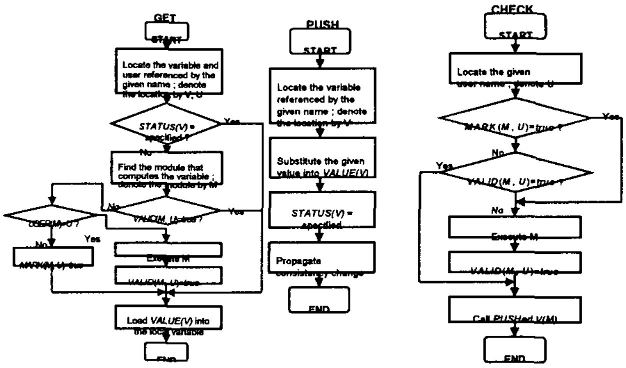
GET
START
Locate the variable and user referenced by the given name ; denote the location by V, U
STATUS(V) = specified ?
Yes
No
Find the module that computes the variable ; denote the module by M
VALID(M, U)=true ?
Yes
No
USER(M)=U ?
Yes
No
MARK(M,U)=true
Execute M
VALID(M, U)=true
Load VALUE(V) into the local variable
END
PUSH
START
Locate the variable referenced by the given name ; denote the location by V
Substitute the given value into VALUE(V)
STATUS(V) = specified
Propagate consistency change
END
CHECK
START
Locate the given user name ; denote U
MARK(M, U)=true ?
Yes
No
VALID(M, U)=true ?
Yes
No
Execute M
VALID(M, U)=true
Call PUSHed V(M)
END

Figure 7. Procedures of the Concurrent QPM

3. Operating Networked VM System

The *operating system of a* NVMS, *whose name is "Collaborative Engineering System, CES"*, and agent-based control systems are implemented to manage diverse activities in an NVMS. The main algorithm used in an agent-based control system is the Concurrent Quasi-procedural Method developed in this paper. This system is developed in WWW environments using the concurrent quasi-procedural method developed in this paper. The basic structure of the collaborative engineering system developed in this paper is shown in Figure 8.. This system is *constructed in a web-based environment, and a user can* participate in collaborative engineering using his or her own personal computer.

The capability of an agent-based operation system based on the concurrent quasi-procedural method includes :

- If some data is changed or a user executes a module, it can update data and propagate consistency automatically. So, validations of modules are conducted on-line.
- When a certain data is needed, it automatically decides engineering activities to be executed with their path to get the needed output. Furthermore, it provides estimation of time and cost.
- It can tell each operator if the execution of his or her job is possible in his or her own terminal on networks considering current status of all sub-modules.

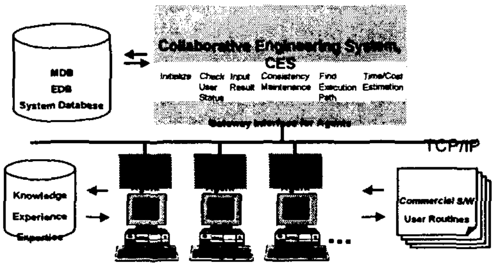

Figure 8. The Structure of CES

For a case study, a scenario of collaborative engineering, including 14 activities, is formulated. The design agent, agents for preparation activities in manufacturing, including process planning and production planning, and agents for virtual manufacturing activities, including virtual operation and a virtual factory are implemented and operated for design and production of an End Cap, a sub-part of electric motor. Figure 9. shows the *Collaborative Engineering Scenario* and results of operation.

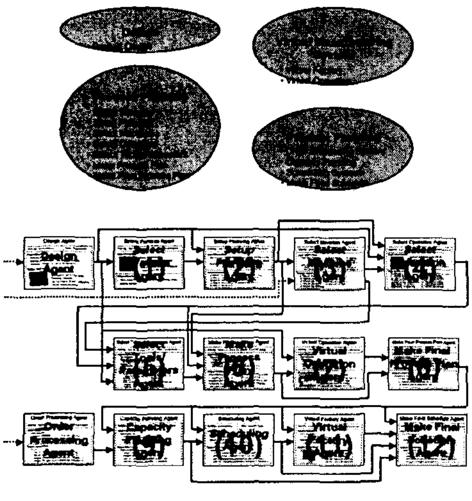

Figure 9. Collaborative Engineering Scenario and Results

4. Conclusion

The NVMS is a new architecture to promote collaboration among multi-functional and diverse engineering and manufacturing activities distributed geographically. A new operating system with high degree of extensibility and flexibility is essential for successful collaboration among diverse engineering activities due to the high level complexity of the targeted system.

In this paper, we presented an agent-based operating system model for a collaborative engineering system in Networked VM System. It is based on the concurrent quasi-procedural method, which is a goal-driven data management technique, developed as an extension of the quasi-procedural method. We expect it to greatly improve the efficiency of operating systems, since it eliminates redundant evaluations by identifying the activities and their execution paths associated with the given object data.

The operating system developed in this paper can promote collaborative engineering among multi-functional users, widely distributed. The operating system is also very useful for developing suitable tools for distributed and network-centric collaborative engineering.

References

[1] Masami Takai, "A New Architecture and Expert System for Aircraft Design Synthesis", *Phd. Thesis of Stanford University*, 1996.

[2] H.-J. Bullinger and J. Warschat, *Concurrent Simultaneous Engineering Systems*, Springer, 1996.

[3] K. Iwata, M. Onosato, K. Teranoto, S. Osaki, "A modeling and Simulation Architecture for Virtual Manufacturing Systems", *Annals of the CIRP*, Vol. 44/1/1995, pp. 379~383.

[4] K. Iwata, M. Onosato, K. Teramoto, S. Osaki, "Virtual Manufacturing Systems as Advanced Information Infrastructure for Integrating Manufacturing Resources and Activities", *Annals of the CIRP*, Vol. 46/1/1997, pp. 335~338.

[5] Kolb, M. A., "A Flexible Computer Aid for Conceptual Design Based on Constraint Propagation and Component-Modeling", *AIAA/AHS/ASEE99 Aircraft Design, Systems and Operations Meeting*, AIAA paper No. 88~4427, 1988.

[6] Kyo Il Lee, Sang Do Noh, "Virtual Manufacturing System - a Test-Bed of Engineering Activities", *Annals of the CIRP* Vol. 46/1/1997, pp. 347~350.

[7] Kyo Il Lee, Sang Do Noh, Young-Geun Han, "Virtual Manufacturing System on WWW", *30th CIRP International Seminar on Manufacturing Systems*, 1998, pp. 245~250.

[8] Peihau Gu and Andrew Kusiak, *Concurrent Engineering, Methodology and Applications*, Elsevier, 1993.

[9] M. Starbek, "The Influence of Concurrent Engineering on Launch-to-finish Time", *30th CIRP International Seminar on Manufacturing Systems*, 1998, pp. 70-78.

[10] Noh Sangdo, "Networked Virtual Manufacturing System for Collaborative Engineering", *Phd. Thesis of Seoul National University*, 1999.