워크플로우 및 워크플로우 관리 시스템의 새로운 조망
(A Fresh Look on Workflow and Workflow Management System)

한 동 수 [†] 심 재 용 [††]
(Dongsoo Han) (Jaeyong Shim)

요 약 본 논문에서는 워크플로우와 워크플로우 관리 시스템을 프로그래밍 언어 관점에서 분석하였다. 워크플로우 관련 데이타, 워크플로우 제어 구조 그리고 응용 프로그램 기동 등 많은 워크플로우 특성이 분산 병렬 프로그램의 해당 항목과 비교되었다. 그 결과 비록 사소한 차이는 존재하였지만 놀랍게도 그들 간에는 많은 유사성이 존재함을 확인할 수 있었다. 이러한 관찰에 근거하여 본 논문에서는 워크플로우 관리 시스템을 분산 병렬 프로그램 개발 플랫폼으로 조망하는 것을 제안하였다. 워크플로우 관리 시스템에 관한 이러한 새로운 조망을 통하여 워크플로우 시스템 사용자는 보다 일관성 있는 관점에서 워크플로우를 바라볼 수 있으며 워크플로우 관리 시스템 설계자는 워크플로우 시스템 설계의 일관성을 유지하면서 워크플로우 시스템에 대한 다양한 요구에 대응할 수 있게 된다. 또한 본 논문에서 제시한 워크플로우와 프로그램의 유사성은 워크플로우와 관련된 많은 분석 기법의 개발에 이미 개발된 프로그램 분석 기법을 원용할 수 있는 논리적 기반을 제공한다.

Abstract In this paper, we analyze workflow and workflow management system in programming language aspects. Many workflow characteristics such as workflow relevant data, workflow control structures, and workflow application invocations are studied and compared with those of distributed parallel programs. Although there exist minor differences between them, we found that there exist surprisingly many analogies between them. Based on this observation, we suggest to view workflow management system as distributed parallel program development platform. This new view on workflow management system provides users consistent view on workflow and workflow management system and with this view workflow management system designer can cope with arbitrary requests from the users keeping design consistency. Moreover the analogy between workflow and program provides a basis to apply program analysis techniques to the analysis of workflow.

1. 서 론

Since the needs of workflow system had been invoked, numerous workflow systems have been developed for either commercial or academic purposes [1, 2, 3, 4, 5, 6, 8, 23]. Early workflow systems have the inclination to focus on equipping functions and facilities that the workflow system is supposed to provide. However with this functional view, the workflow management system designers are apt to react to the requests on the workflow management systems impromptu.

A function, provided to support a workflow case may conflict with other existing functions or may cause interferences when processing other workflow cases which may result in complex construct of workflow management system. Sometimes it is even unclear whether the requests should be accommodated in the workflow management system

† 종신회원 : 한국정보통신대학원 공학부 교수
dshan@icu.ac.kr
†† 비 회 원 : 한국정보통신대학원 공학부
jaeyong7@icu.ac.kr
논문접수 : 1999년 12월 17일
심사완료 : 2001년 6월 20일

or not. For instance still there is no clear evidence whether inter-workflow instance communication facility should be provided or not in workflow management system. When the needs become clear by some crucial workflow cases, each workflow management system tries to equip the functions. These phenomena will continue as long as we view the workflow management system in functional aspects.

Workflow application developers also have to pay attention to the functions provided by the workflow management system that he is working on. That is why an expert of workflow application developer of one system cannot be an expert of other workflow management system immediately. He/She needs training to change the environment. It is more desirable that a workflow application developer can expect what functions or environment should be provided by workflow management system.

More consistent view on workflow and workflow management system should be provided based not on functions but on some foundation which is firm and easy to understand. In this paper we analyze workflow and workflow management system in programming language aspects for this purpose. We assume the distributed parallel program may be executed in HAD(Heterogeneous, Autonomous, Distributed) environment or in shared memory multiprocessors because the platform does not influence so much to our discussion. Although there are some minor differences between workflow and distributed parallel program, we have found that there are surprisingly many fundamental analogies between them. Even workflow management system can be interpreted as a platform to develop distributed parallel programs and workflow application development can be viewed as distributed parallel programming.

This new view can provide consistent view of workflow management system to the workflow management system designers as well as workflow application developers. When some conflicts are met in designing a workflow management system, the system designer can cope with the situation judiciously keeping design consistency. Meanwhile workflow application developers can understand the workflow application developing process more clearly and can arrange the order of development with confidence. This consistent view on workflow and workflow management system can help them to forecast or pin out the spot of problem easily when developing workflow applications.

Furthermore this new view on workflow management system provides concrete basis for workflow management system designers to approach subtle issues. For instance previously mentioned inter-workflow instance communication problem can be reduced to the inter-process communication problem that can be considered to have been resolved already in a way. Which guides the way of implementation to support inter-workflow instance communication mechanism in workflow management systems. Moreover the analogy between workflow and program provides the basis to apply program analysis techniques to the analysis of workflow. We developed a technique by following this approach[22]. In the technique, set based program analysis technique is used to the access conflict analysis of concurrent workflow definition.

The paper is organized as follows. In section 2, we introduce the definition and the meaning of workflow and workflow management system briefly. In section 3, we compare the variable declaration and workflow relevant data definition. In section 4, we explain control structures. In section 5, we compare procedure call with application invocation. In section 6, we compare program execution step with workflow execution step. In section 7, we suggest new view on workflow management system. Related works are introduced in section 8 and we draw conclusion in section 9.

2. Preliminaries

Workflow is defined as "the computerized facilitation or automation of a business process, in whole or part" and workflow management system is defined 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" in WfMC standard[7]. Although the definition certainly describes workflow and workflow management system precisely and succinctly, it is rather focused on the functions of workflow and workflow management system.

The functional view on workflow and workflow management system may be adequate to help a layman of workflow understand them. However the functional view may underestimate the potential value of workflow and workflow management system. Although workflow is originated from business area, if we consider workflow not as business process automation but as general program, workflow can cover much wider area. All the distributed parallel programs in HAD(Heterogeneous, Autonomous, Distributed) environment can be considered as workflow. Superficially, workflow is exposed as the integration of loosely coupled distributed programs or processing of distributed user inputs. But actually it is a program as a whole. The programs need not to automate only business processes.

In fact the procedure of a workflow application development is coincident with that of a program writing largely. There is no determined procedure in developing a workflow application yet. Nevertheless following steps can be thought as one of general workflow application development steps accepted currently. We assume some environments like workflow participant and organization structure are already prepared:

1. Workflow applications for automatic tasks are developed and registered.
2. Workflow relevant data are defined.
3. Activities are defined with attributes such as which workflow application would be invoked in the activity.
4. Control flow and data flow between activities are defined.

Similarly, we write a program according to the following steps:

1. Procedures or libraries are prepared.
2. Variables are declared.
3. Program statements which may be procedure calls and the way of their connections are denoted.

The step 3 includes the steps 3 and 4 of the previous workflow application development procedure. As can be understood from the comparison of the previous two procedures, the workflow application development procedure and the program development procedure are very similar which is another strong background that we can view workflow as program. In the following sections we find and study the bases which back up the argument that workflow is distributed parallel program in HAD environment.

3. Variable Declaration and Workflow Relevant Data Definition

When we view a workflow in terms of a program, the relevant data definition in designing a workflow process can be interpreted as the action of defining variables when writing a program or a procedure. Various kinds of data types, which range from scalar data types such as integer, real, character types to composite data types such as array, list, and record types, are supported by workflow management systems in general[7].

Workflow management system has to allocate space to accommodate the defined workflow relevant data according to the defined data types for each process instances. The space allocation for workflow relevant data in workflow management system is the same action that a compiler allocates memory for the declared variable in a program when translating the program.

Since concurrently invoked applications in the same context of a workflow instance can access the same workflow relevant data simultaneously, access control mechanism to the shared relevant data should be provided in workflow management system. This is a similar situation when multiple threads try to access the shared data at the same time in shared memory parallel programming

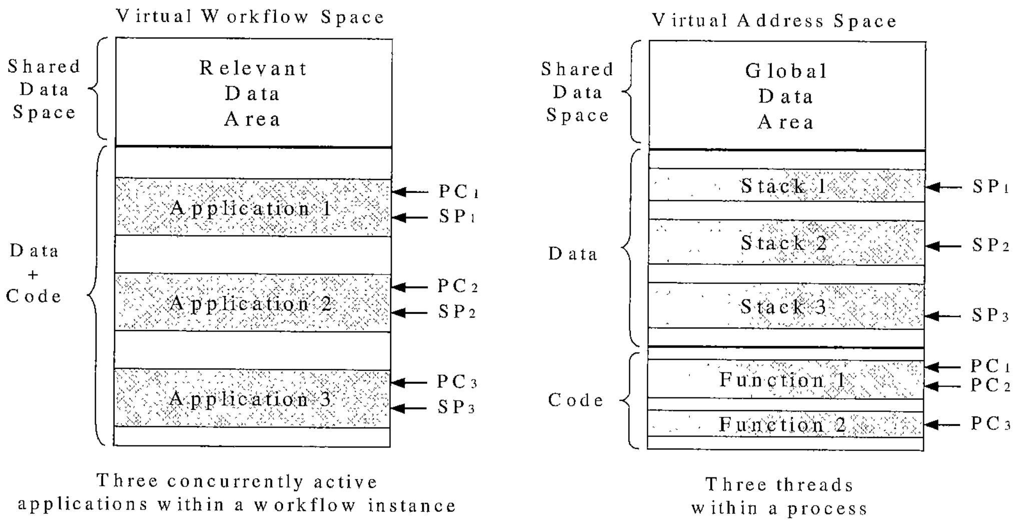


Fig. 1 The memory layout of virtual workflow space and virtual address space of threads

environment. Figure 1 shows the memory layout of multiple threads in a process in virtual address space and the logical relation of concurrently active workflow applications in a workflow instance in virtual workflow space. Although the workflow applications do not share physically the same memory space and can be located in remote sites each other, the logical relation between applications are very similar with that of threads in a process.

Race condition can be raised and mutual exclusion mechanism[15] in which each thread has exclusive access to those parts of memory that it is changing is provided for the control of the simultaneous access in thread execution environment[19]. Similar mutual exclusion mechanism for concurrently invoked workflow applications has to be provided in workflow management system.

Two approaches to resolve race condition of workflow relevant data can be considered in workflow management systems. One approach is to provide lock and unlock primitives to workflow application programmer that they could guarantee exclusive access to the shared relevant data. This approach is equivalent to multi-threaded programming in shared memory multi-processors in which each thread exclusively accesses shared program area using mutex variables[19]. Usually shared resources are accessed in the shared program area.

The other approach is to denote the workflow relevant data as shared resource when defining them like some parallel programming languages approach. Workflow management system automatically protects the shared resource when it is accessed simultaneously. FCFS(First Come First Served) strategy is often adopted as the scheduling mechanism for the processing of multiple simultaneous requests.

The second approach seems to be more proper in workflow systems because it is difficult to expect that the workflow application developers grasp the shared resource exactly in global workflow definition environment. The sharing information is better understood by the whole workflow process designer with the help of workflow application developers.

4. Control Structures

The control structures used in workflow process modeling are very similar to those of parallel programs. Sequential, conditional branch, and iteration are the typical control structures for sequential programs. Parallel fork and join control structures are included additionally for parallel programs. Corresponding structures are expressed by the connection of the edges and nodes in workflow process definition graphs.

Sequential control structure is denoted by connecting single out-edge node with single in-edge node. Branch control structure is specified by connecting an XOR node with multiple single in-edge nodes. Iteration control structure is expressed either by an iteration node or by forming a cycle in the process definition graph. Parallel fork control structure is specified by connecting an AND node with multiple single in-edge nodes. Join control structure is denoted by connecting a collecting node with multiple single out-edge nodes. Goto statement can be simulated by connecting a source node to a destination node arbitrarily. Figure 2 shows the control structures for parallel programs and their corresponding control structures in workflow definition graphs.

Although more detailed control flow can be expressed in parallel programs, almost all the control structures of parallel program can be expressed by workflow process definition graphs. Note that in early programming language theory, restricted use of control structures has been recommended to enhance the readability of a program[16, 17]. For instance, careless use of goto statement such as the goto statement entering inside a loop is pointed out as an ill programming style and its arbitrary use is not allowed in many programming languages[15]. Block structured programming in which only sequential, conditional branch, and iteration is the main control structure and the limited usage of goto statement like escape from a loop or a conditional statement is allowed. Thus good control structure in workflow modeling can be constructed by observing the block-structured programming style when connecting activity nodes. Workflow modeling tool may allow only the confined way of connections between nodes to help modelers construct workflow process with good control structure.

5. Application Invocation vs. Procedure Call

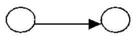
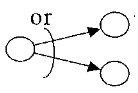
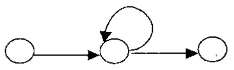
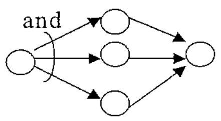
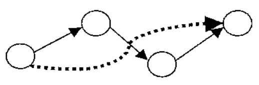
Control Structures	Parallel Programs	Workflow Graphs
Sequential	S1; S2	
Conditional Branch	**If** *condition* **then** S1 **else** S2	
Iteration	**while** *condition* **do** S	
Parallel Fork	**parbegin** S1; S2;...;Sn **parend**	
Parallel Join		
Arbitrary Branch	**goto** L	

Fig. 2 Control structures of parallel programs and workflow graphs

Application invocation during a workflow execution can be regarded as procedure call in a program execution. Hence the input/output workflow relevant data delivery between workflow application and workflow management system can be interpreted as the parameter passing of procedure calls in a program. Although numerous parameter passing mechanisms have been suggested and studied intensively[15], call-by-value and call-by-reference parameter passing mechanisms are most widely accepted. The input/output delivery of workflow relevant data between workflow application and workflow management system in workflow management system can be implemented referring to the parameter passing mechanisms.

In the call-by-value parameter passing mechanism, the value of actual parameter is copied to that of formal parameter and the value of the formal parameter is copied back to the actual parameter after the called procedure is ended. To implement call-by-value mechanism in the input/output data delivery between workflow applications and workflow management system, not the reference but the value of the workflow relevant data should be passed from workflow management system to workflow application. Since application agent often relays the data between workflow application and workflow system, it reserves space to keep the data temporarily. Note that the update of the delivered data is not reflected to the workflow system until the invoked application finishes its work and the control is returned to the application agent or workflow system.

In the call-by-reference parameter passing mechanism, the reference of the actual parameter is passed to the called procedure and they share the same memory location. The update of the contents of the formal parameter has the same effect as changing the contents of the actual parameter immediately. To implement call-by-reference mechanism in the input/output data delivery between workflow applications and workflow management system, not the value but the reference of the workflow relevant data should be passed from workflow management system to workflow application. Thus the update of the workflow relevant data through the reference has the effect of changing the workflow relevant data immediately within the application. Figure 3 compares between data transfer by call-by-value and call-by-reference.

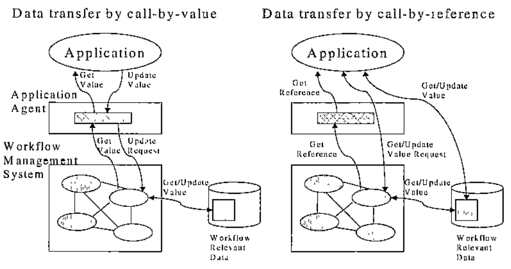


Fig. 3 Data transfer by call-by-value and call-by-reference

Each of the approach has its pros and cons. When we consider the situation that multiple workflow applications try to access the same workflow relevant data simultaneously which is similar situation when multiple threads tries to access the same memory location at the same time, call-by-reference parameter passing seems to be more suitable in workflow processing environment. Some workflow systems trace the change of each workflow relevant data to cope with the system failure. To enable the tracing in a workflow system, the workflow system has to maintain history information whenever the contents of the delivered workflow relevant data have been updated. When the data delivery is implemented by the way of call-by-reference parameter passing, more minute control of the execution environment is possible. Which can help the workflow system have reliable error recovery mechanism. But in mobile environment where a workflow client could be disconnected from the workflow server, call-by-value parameter passing mechanism is more proper.

6. Workflow Execution Step vs. Program Execution Step

We define workflow execution step from the start of a activity to the end of the activity. Program execution step is defined from the start of a program statement to the end of the statement. A workflow execution step may involve several sub-execution steps and each sub-step usually involves an invocation of an application program which usually involves numerous program execution steps. Thus the workflow execution steps and the program execution steps appear in turns when a workflow instance is executed in a workflow system.

Superficially the workflow execution step is distinguished from the program execution step in that the grain size of a workflow execution step is larger than that of a program execution step. But more important property of the workflow execution step is that it returns the control back to the workflow management system whenever it finishes its step. Which does not happen in the program execution step during a program execution. Work-flow management system registers change of status and maintains the event history information in the permanent storage area when it receives control from the outside of the system. This enables workflow system to cope with the system failure and dynamic workflow reconfiguration[11,12] during a workflow execution which is impossible in a program execution.

On the other hand, the mixed appearance of workflow execution step and program execution step in a workflow execution make clear the limit of the recovery of a workflow system. That is when a failure is occurred the workflow execution step should be the start point for the recovery unless some primitives for recovery are not imbedded in programs.

7. Workflow Management System vs. Operating System

Workflow management system is to workflow instance what operating systems is to program in execution. Workflow management system manages submitted workflow instances to complete its work. It allocates space for the workflow instances and arranges the order of execution according to its scheduling mechanism. Operating system manages memories and schedules the execution of processes. It allocates spaces in memory to execute programs and decides the next program to be executed when CPU is freed.

The difference between the workflow manage-ment system and operating system is not in the functions of each system but in the way of imple-mentation. The different way of implementation originates from the different characteristics between programs and workflow instances. Workflow management system usually allocates space for workflow relevant data in permanent storage area because workflow instances often durate for a long time. When failure occurs during the processing of a workflow instance, enactment of the workflw instance should be restarted not from the start point of the process but from the nearest point to the failure point where the system can recover. Thus the maintenance of execution history in permanent storage area while processing workflow instances for the reliable management of the workflow instances is crucial in workflow management system.

On the contrary, operating system allocates space in memory for the image of a program. When fault happens during the program execution, it is enough that the operating system can simply restart the program again when recovered. Thus the main-tenance of execution history in permanent storage area during a program execution is not necessary.

Various priority based scheduling methods have been adopted in operating systems. In workflow management system, priority of workflow instance is the major factor in selecting the next workflow instance to be handled when multiple workflow instances wait for their turns. But higher priority of a workflow instance does not always imply its earlier end of execution which is usually true for programs in operating system. This is because

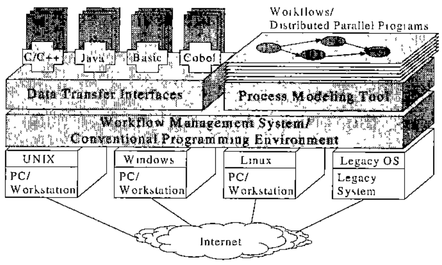


Fig. 4 New view on workflow management system

workflow instances have to wait for the end of many outer actions that can not be controlled by workflow management system. Thus the priority information of a workflow instance is more reasonable and effective to be used in alerting the workflow participants of the workflow instance that the work is urgent.

8. New View on Workflow Management System

When we view workflow application development as distributed parallel or thread programming in HAD(Heterogeneous, Autonomous, Distributed) environment, the workflow management system can be regarded as the distributed parallel program development platform in HAD environment like Figure 4. Since various application programs developed in different languages in different sites can be involved in a workflow application, the workflow management system should be able to integrate the applications.

Two facilities are needed for the integration. First, shared space for the application programs for sharing information should be provided. Workflow relevant data area can be used for this purpose. The area can be regarded as global area for the application programs involved to the same workflow instance. They can communicate one another through the workflow relevant area like the threads in a program communicate through the global data area.

Second, interfaces for languages to access the workflow relevant data should be provided respectively. Using these interfaces, application programs developed in different languages can communicate each other and a workflow instance can achieve its goal by coordinating the application programs. For remote applications in distributed environment to access the shared workflow relevant data, some mechanism should be provided to use the interfaces in anywhere the application is executed. Currently, there is no standardized mechanisms and the implementation is dependent on workflow management system vendors. Using the interfaces, application programs in remote sites to be integrated to build a workflow application can be developed in the illusion that they are in the same program sharing the same memory space like the threads in the same program shares their global memory space.

9. Related Work

There are many views on workflow management system. Some view workflow management system just as a router such as image or document. The others view workflow management system as a glue to integrate applications[10]. Recently, workflow management system is recognized as the key component in enterprise application integration(EAI) area. Alonso et al.[20] designate workflow management systems are defined by the functionality of commercial systems and that has been the major source of their limitations. Their view coincides with our view on workflow management system in a way.

Some puts the value of workflow management system in its capability of the separation of business logic from application logic which enables an expert in an application domain can conduct the business process design without the help of programmer[1]. These views on workflow management system certainly expresses some benefits of using workflow management system. However it can not be the total view of workflow and workflow management system. When we view workflow and workflow management system in different angles, we can find much more benefits that they can provide.

In this paper we viewed workflow management system as a distributed parallel program development platform. Once we view workflow as program and workflow management system as distributed parallel program development platform in HAD environment, a workflow management system can be viewed as a platform or mechanism to integrate distributed applications like EAI(Enterprise Application Integration). CORBA/DCOM have become popular as an architecture to integrate distributed objects developed in different languages [9,13,14]. While CORBA/DCOM helps one to build distributed objects and to organize programs with the distributed objects, workflow management system helps one to generate new upper level service programs by organizing distributed applications developed in different languages. Some of which may be developed using CORBA/DCOM objects.

In some sense, workflow management system provides much more handy means to integrate distributed applications especially for legacy programs than CORBA/DCOM mechanism because they should be changed into CORBA/DCOM object for them to be used in CORBA/DCOM environment. Only small number of input/output statements needs to be embedded into a legacy program for the program to be integrated into a workflow.

Ranno et al.[21] propose a script language aimed at expressing tasks composition and inter-task dependencies of distributed applications. They illustrate class templates to represent their workflow model and by which they show distributed applications can be integrated using the class templates. Their approach is certainly one of direction to leverage the distributed application integration capability of workflow management system. However they did not reach the point like some analysis techniques for workflow can be borrowed from the program analysis techniques.

The arbitrary integration of remote applications in complex workflow control structures might raise serious race problems that can not be detected easily. To let the people fully enjoy the benefits of the workflow management system, more deliberate support that makes them free from raising of race conditions in workflow management system level. Few workflow management systems provide mechanisms to control such situation currently. The view of this paper can be helpful in pointing the direction to resolve such problems.

10. Conclusion

In this paper we have studied the analogies and differences between workflow and programs. Surprisingly, we found many analogies between them than we had expected before we started this study. We found a workflow application development is similar with distributed parallel programming sharing the same memory address space. Almost all the control structures of workflow can be expressed in the parallel program control structures and the workflow relevant data definition is the same action of global variable declaration in thread or parallel programming.

We regarded workflow management system as a platform for distributed parallel programs in HAD environment. This new view provides consistent view on workflow management system to the workflow management system designers as well as workflow application developers. When architectural conflict occurs in designing a workflow management system, the system designer can cope with the situation judiciously keeping consistency. Workflow application developers can understand the workflow application developing process much more clearly and can arrange the development procedure. They can forecast or pin out the problem spot easily when developing workflow applications.

Furthermore this new view on workflow management system provides concrete bases for workflow management system designers to approach subtle issues. For instance inter-workflow instance communication problem can be reduced to the inter-process communication problem that can be considered to have been resolved already in a way.

Finally the analogy between workflow and program provides the basis to apply program analysis techniques to the analysis of workflow. One may refer to the techniques that have developed or resolved already in programming language areas when he encounters similar problems in workflow area to those in program language areas. Which leads us to develop set-based access conflict analysis in concurrent workflow definition[22]. But this is a just starting point of this approach. Note that while workflow is relatively new area, more than 50 years of research works have been accumulated in programming language areas.

References

[1] Peter Lawrence, "Workflow HandBook" published in association with the workflow management coalition, 1997.

[2] S.McCready, "There is more than one kind of workflow software," ComputerWorld, November 1992.

[3] S. Das, "ORB Work: A Distributed CORBA-based Engine for the METEOR2 Workflow Management System," Master's thesis, University of Georgia, Athens, GA, March 1997.

[4] D. S. Han, H. J. Park, "Design and Implementation of Web Based Business Process Automating HiFlow System," Journal of Korean Information Science Society(C) : Computing Practices, Vol. 4, No. 1, Feb. 1998.

[5] B. R. Silver, "The BIS Guide to Workflow Software: A Visual Comparison of Today's Leading Products," Technical report, BIS Strategic Decisions, Norwell, MA, September 1995.

[6] Nortel & University of Newcastle upon Tyne, "Workflow Management Facility Specification," Revised Submission, OMG Document Number: bom/98-03-01, 1998.

[7] Workflow Management Coalition Specification Document, "The Workflow Reference Model," Version 1.1, November 1994.

[8] Joint Submitters, "Workflow Management Facility," Revised Submission, OMG Document Number: bom/98-06-07, July 4, 1998.

[9] Z. Yang and K. Duddy. "CORBA: A Platform for Distributed Object Computing," In ACM Operating System Review, Vol. 30, No. 2. Pages 4-31. April 1996.

[10] D. Georgakopoulos, M. Hornick, A. Steth, "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure," Distributed Parallel Databases, Klewer Academic Publishers, Volume 3, Number 2, pp. 119-154, 1995.

[11] Clarence A. Ellis, Keddara K and Rozenberg G., "Dynamic Change within Workflow Systems," Proceedings of the ACM SIGOIS Conference on Organizational Computing Systems, Milpitas, CA., 1995.

[12] Manfred Reichert, Peter Dadam, "A Frame work for dynamic changes in workflow management system," Proceeding of DEXA'97, 1997.

[13] Object Management Group Inc. "The Common Object Request Broker: architecture and Specification," OMG Document Revision 2.2, Feburuary 1998.

[14] E. Frank and III. Redmond, "DCOM: Microsoft Distributed Component Object Model," IDG Books Worldwides, 1997.

[15] L. Lamport, "The Mutual Exclusion Problem," Journal of the ACM, Volume 33, Number 2, pp. 313-348, 1986.

[16] N. Wirth, "On the Composition of Well-Structured Programs," ACM Computing Surveys, Volume 6, Number 4, page 247, 1974.

[17] J. Donaldson, "Structured Programming," Datamation, Volume 19, Number 2, Page 52, 1972.

[18] Kenneth C. Loudon, "Programming Languages : Principles and Practice," PWS-KENT Publishing Company, 1993.

[19] B. Nichols, D. Buttlar, J. P. Farrell, "Pthreads Programming," O'Reilly & Associates, Inc. 1996.

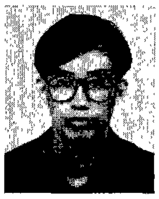
[20] G. Alonso, D. Agrawal, A. El. Abbadi, C. Mohan, "Functionality and Limitations of Current Workflow Management Systems," submitted to IEEE Export Journal 1997.

[21] F. Ranno, S. K. Shirivastava, S. M. Wheater, "A Language for Specifying the Composition of Reliable Distributed Applications," 18th International Conference on Distributed Computing Systems, 1998.

[22] M. K. Lee, D. S. Han, J. Y. Shim, "Set-Based Access Conflict Analysis of Concurrent Workflow Definition," to be appear in Information Processing Letters, Elsevier Science.

[23] D. S. Han, J. Y. Shim, C. S. Yu, "ICU/COWS : A

Distributed Transactional Workflow System Supporting Multiple Workflow Types," IEICE Transactions of Information and Systems, Vol. E83-D, No. 7, July 2000.



한 동 수

1989년 서울대학교 계산통계학과 학사. 1991년 서울대학교 계산통계학과 석사(전산과학). 1996년 일본 교토대학교 정보공학부 박사(컴퓨터 공학). 1991년 ~ 1992년 (주)삼성전자 연구원. 1996년 4월 ~ 1996년 7월 일본 NEC C&C 연구소 연구원. 1996년 9월 ~ 1997년 10월 (주)현대정보기술 책임연구원. 1997년 11월 ~ 현재 한국정보통신대학원 정보공학부 조교수. 관심분야는 분산 워크플로우 관리 시스템, 병렬 컴파일러, 고성능 컴퓨팅

심 재 용

1994년 서강대학교 전자계산학과 학사. 1996년 서강대학교 전자계산학과 석사. 1998년 3월 ~ 현재 한국정보통신대학원대학교 공학부 박사과정. 1996년 1월 ~ 1998년 2월 (주)현대정보기술 정보기술연구소 연구원. 관심분야는 상호연동성과 적응성 특징을 갖는 워크플로우 관리기술, 정확성을 보장하는 워크플로우 관리기술, 분산 객체 기술, 이동컴퓨팅 기술