Run-time encapsulation of nested process models for managing complex business processes

Suk-Ho Kang[1], Yeongho Kim[2], Dongsoo Kim[3]

Department of Industrial Engineering, Seoul National University
[1] (shkang@cybernet.snu.ac.kr)
[2] (yeongho@snu.ac.kr)
[3] (soo@ara.snu.ac..kr)

Abstract

Workflow management systems (WFMSs) are becoming important technology to manage business processes more efficiently. Participants of a business process might be geographically dispersed and use various computing platforms while cooperating to achieve their business goal. Web-based workflow management systems can be a good solution to manage the business processes in a distributed and heterogeneous environment. Several major requirements of WFMSs for such environment are identified in this paper. Business processes occurring over a wide range of departments or different organizations tend to be very complex and hard to define precisely at the design time. A method of business process modeling, called nested process, has been proposed. It can provide an easy way of modeling complex processes and the foundation of run-time encapsulation mechanism that takes advantage of the modular features of the models. When the execution of a business process involves several different departments and organizations, most task performers in the process are responsible for a portion of the whole process, and thus participate only in the associated part. Also, when the process involves external participants, some sub-processes may have to be served by different workflow engines. In either case, a sub-process needs to be encapsulated, so that its control and work-contents can be protected in a proper level. To handle this situation it is required run-time encapsulation of processes. It proposes the modularization of a complex business process into several sub-processes (modules), and provides a structure for a hierarchical arrangement of the modules. The mechanism of run-time encapsulation is applied to a module that is the basic unit of process management in our research. We have developed a prototype system for realizing the concept for managing complex processes. Some important functions and features of the developed system are explained. The system enables effective cooperation among workers who are geographically separated.

Keywords: Nested process model, Run-time encapsulation, Workflow management system, Cooperation

1. Introduction

The term 'workflow' is used to describe a business process that is executed and managed automatically by a computer system. WFMS is a system that completely defines, manages and executes the workflow through the execution of software whose order of execution is driven by a computer representation of the workflow logic[2]. The importance of WFMS has greatly increased with the rapid changes of business environments. For example, consider enterprise information systems such as ERP (Enterprise Resource Planning), PDM (Product Data Management), SCM (Supply Chain Management), CRM (Customer Relationship Management), and EC (Electronic Commerce), which recently draw a great amount of attention from industries. Each of these systems includes the workflow management as its key component. This is mainly due to the fact that many functions of those systems involve processes composed of a set of tasks executed in a certain business logic.

WFMS provides several advantages. First, complex business processes can be easily handled since the business flow is automatically controlled. Second, it can expedite the execution of business processes with the assistance of digitalized delivery of information and documents. Third, it promotes consistent access to and increased accuracy of data, which leads to elevated reliability of the outcome of business processes. Fourth, the system allows ready, transparent auditing of business processes in real time. Finally, WFMS generally provides groupware functions that facilitate efficient collaboration during the business processes execution. Overall, WFMS can greatly improve the productivity of business processes in terms of cost and time. These are the main reasons why WFMS is regarded as the essential module to the enterprise information systems aforementioned.

The purpose of the system envisaged in this research is to provide run-time encapsulation mechanism for managing complex processes and implement the concept by developing a Web-based workflow management system fully compliant to the Internet. As the Internet users have been increasing explosively for the last several years, Web browsers such as Internet Explorer and Netscape Navigator become the de facto standard of user interfaces. This has motivated many vendors of information systems to develop Web interfaces to their systems. WFMS is no exception to the trend, and hence most vendors of WFMS have released Web versions of their products. The greatest advantage of Web-based WFMS is that users are not confined to their physical locations. That is, the users can have a ready access to the system at any time at anywhere. Another advantage comes from easy maintenance and update of client programs, because they do not have to be installed for each client user.

Another important consideration is given to a modular approach to effective design and execution of business processes. The complexity of business processes can vary depending on the number of components constituting the processes and their relationships. The processes that we are interested are executed in a distributed environment. In general, such a process is highly complex; it requires collaboration among participants in several different organizational units, the process itself may alter according to the change of situation and the interim results of the process, and several sub-processes can proceed in a parallel manner. We propose a nested modeling technique of business processes. This method allows the modularization of a complex business process into several sub-processes (modules), and provides a structure for a hierarchical arrangement of the modules. The mechanism of run-time encapsulation is applied to a module that is the basic unit of process management in our research.

2. Requirements for Managing Complex Processes

WFMSs, especially for managing complex business processes in a distributed and heterogeneous environment, need to fulfill requirements listed below. Among them nested modeling and run-time encapsulation are the most important requirements focused in this paper.

- Nested modeling: As a business process takes place over several departments and is often associated with other organizations, the process is usually very complicated. Nested modeling allows the design of such complex process by breaking it into several sub-processes.
- Run-time encapsulation: It is very common that the execution of a business process involves several different departments and organizations. Most task performers in the process are responsible for a portion of the whole process, and thus participate only in the associated part. Also, when the process involves external participants, some sub-processes may have to be served by different workflow engines. In either case, a sub-process needs to be encapsulated, so that its control and work-contents can be protected in a proper level.
- Platform independence: The server of WFMS – the workflow engine - needs to be platform-independent. This means that the server can be readily installed on any hardware and/or operating systems. An additional requirement is that the implementation should be reconciled with communication protocol.
- Location transparency: This requirement is included to support client users who may be geographically distributed or frequently changing their working locations. As a key to the location transparency, it would be desired for client programs to be downloadable from the server.
- Effective notification: WFMS for an inter-organizational process involves many external users. Since such a user is not supposed to routinely check the tasks assigned, the user is often unaware of the tasks delivered. This requires the system to have an effective notification mechanism of task delivery.
- Information sharing: In a distributed, heterogeneous environment, information sharing would be more restrictive due to organizational barriers. A system is needed to facilitate joint ownership of the required information.

3. System Architecture

The overall system architecture is illustrated in Figure 1.

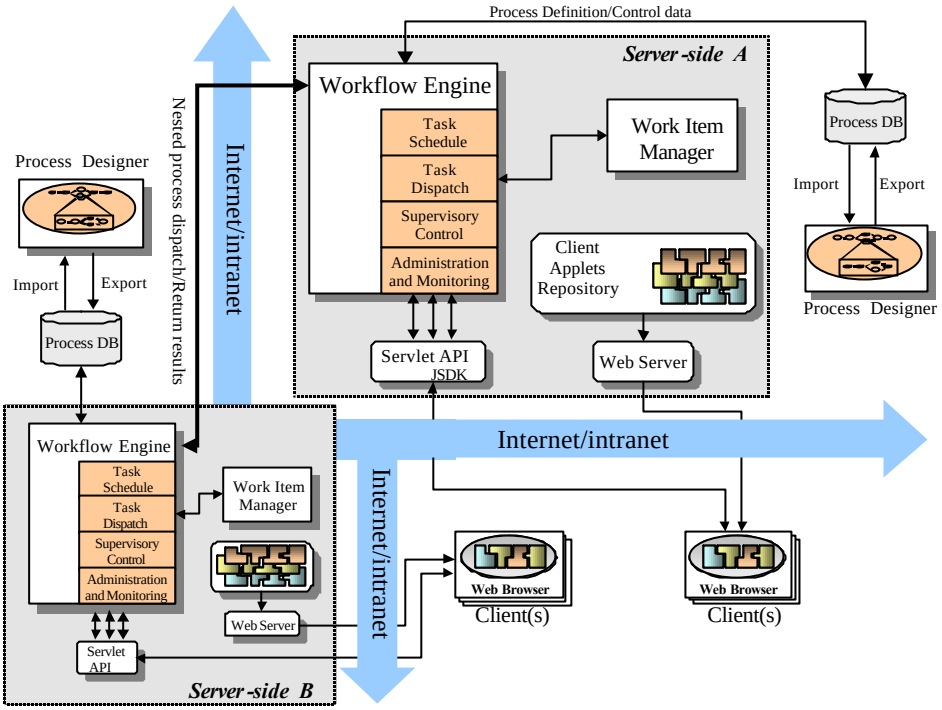

Fig. 1 System architecture [9]

The system consists of Process Designer, Workflow Engine, and Workflow Client. Process Designer is a stand-alone client application used by a designer to model business processes. A process model is translated into a format which the Workflow Engine can interpret, and is exported into Process DB. A model in the Process DB can be imported into the Process Designer, and then it can be modified and/or be embedded in other processes as a sub-process. We use ODBC (Open Database Connectivity) for the communication between the Process Designer and the database.

Workflow Engine executes the process flows established by the Process Designer. The engine creates or updates a control schedule based on the status of process execution. It identifies the performers appropriate for a given task, and dispatches the tasks according to their corresponding process definitions in DB. Once the Workflow Engine releases the tasks, Work Item Manager manages them. Users check out their tasks from the Work Item Manager. Together with this pull approach, a simple push technique is used. Whenever a task is released to a performer, the engine automatically sends an E-mail to notify the performer. This frees the user from the burden of being constantly online. Since E-mail can be read using portable devices such as PDA and mobile phone, the task assignment is instantly alerted to the relevant task performer. Also, this approach could be an advantage to users who utilize the system not so frequently. Additional services that the engine provides include supervisory controls such as expediting, aborting, suspending some tasks, monitoring of active processes, and provision of statistics on processes and tasks that have already been completed.

Workflow Client is the user interfaces. Workflow users can be classified as initiators, task performers, supervisory managers, and system administrators, according to their roles and responsibilities. Since all the user interfaces are developed using Java applets, they can be downloaded via a Web browser. This provides the location transparency, another important requirement noted in Section 2. It also enables easy maintenance, which is a general advantage of using Web-client/server system. To promote the requirement of information sharing, a client interface is interlinked with a document management system, which also runs on the Web environment. However, due to the space limitation, this is not considered in this paper.

A distinguished feature of our system is that multiple WFMSs at remote sites can interact with each other over the Internet. The figure illustrates that the system is installed at two sites A and B. Since it is implemented in pure Java, it ensures the platform-independence, which is one of the important requirements mentioned in Section 2. While executing a process, the two systems are able to interact using the run-time encapsulation mechanism. This is further described in Section 4.3.

We employ Java servlets[5], which provide API (Application Programming Interface) for communication between client and server. Java servlets are modules that extend request/response-oriented servers, such as Java-enabled Web servers. Servlets are to the server what applets are to the client. Servlets differ from applets in that servlets do not run in a Web browser or with a Graphical User Interface (GUI). Instead, servlets interact with the servlet engine, like Jrun and Servletrunner, running on the Web server through requests and responses. A client program, which could be a Web browser or some other program that makes connections across the Internet, accesses a Web server and makes a request. The servlet engine that runs with the Web server, which returns a response to a servlet, processes this request. The servlet in turn sends a response in HTTP form to the client.

Servlets are an effective replacement for CGI scripts. They can handle multiple concurrent requests in a multithreaded environment, which is one of the reasons why we adopt this technology. Servlets can be embedded in many different servers because the servlet API assumes nothing about the server's environment or protocol. Servlets have become most widely used within HTTP servers; many web servers support the servlet API. Therefore, we came to a conclusion that servlets are an efficient development tool for WFMS in a distributed and heterogeneous environment. As servlets are developed in Java, which is relatively easy to program, the overall development efforts and time could be reduced.

4. Run-time Encapsulation of Nested Process Models

4.1 Nested Process

In general, the structural aspect of a process is defined as its constituent tasks and the precedence relations among them. In this conventional approach, a task is considered as a component of process. In a nested process, however, a task can further be mapped onto another process model that contains more detailed specifications of the task. This means that a task is a parent of the process, while being a child of a high level process at the same time. Now, the parent-child relationship alternates between tasks and processes. In this model, a task called 'Root task' becomes the top most object, unlike in the conventional process models.

Consider the example nested process structure in Fig. 2. A circle indicates a task, and a rectangle a process. Tasks are classified into two types, primitive task and nesting task, which are represented by a thin line and a thick line, respectively. An arc represents the precedence relation between two tasks. In the figure, a nesting task is always deployed into a sub-process. That is, the details of the nesting task are modeled in the sub-process. This process is called a nested process. A primitive task cannot be broken down any further. In the example, the Root task, T_0, is at the uppermost level, and its detail is modeled in process P_1. Among the tasks of P_1, T_{12} and T_{14} are nesting tasks, which are again deployed to process P_2 and P_3. The hierarchical relationship can appear at multiple levels, and there is no limit to the number of levels. The example also illustrates that a process definition may be used several times, like P_4.

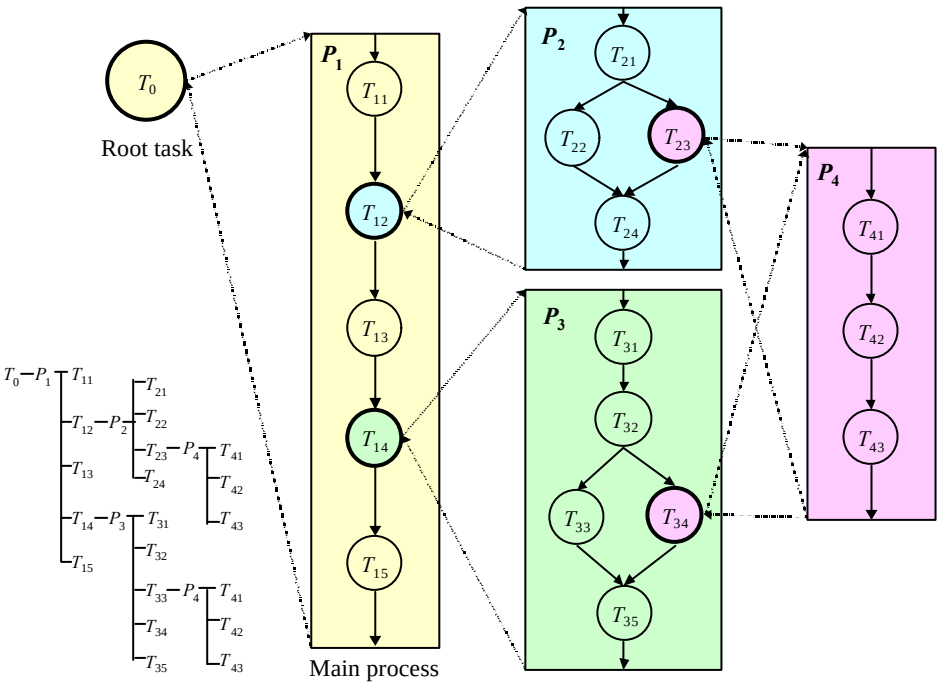

Fig. 2 Tree structure of a nested process

The concept of nested modeling can be very useful for an inter-organizational setting as well as intra-organizational one. Since a business process in a distributed environment involves many participants working for different organizations, it is not unusual that the process involves several sub-processes that are to be carried out by the participants in different organizations. In our approach, such a sub-process is represented as an abstracted task at the super level process model. A person who is in charge of the whole process may design a model at the top most level. This model may include several nesting tasks that cannot be thoroughly modeled at the top most level. Usually, the execution and control of such a nested process is delegated to a person who has an expertise in the sub-process. Therefore, it would be more desirable to hand over the detailed definition to the person. The definition of a nesting model in the super level may include required inputs and outputs only.

Furthermore, the consistency in the representation and structural semantics throughout the model makes it straightforward to control the processes. A nested process is in fact a basic unit that can be designed, executed and monitored independently. The execution steps of nested process are explained as follows. A process execution begins with a call to the Root task enclosing the process. This immediately launches the process, and the execution proceeds according to the precedence relations of its constituent tasks. When it encounters a nesting task, the super level process is hanged up temporarily until the nested process is completed and returns its results. The nested process is executed in a similar manner. On the completion of the top most process, it returns the results to the Root task. Then, the Root task terminates the whole process. Such a mechanism can be used for implementing interoperability among heterogeneous systems. Not only is this mechanism simple, but it also provides consistency for multi-level nesting.

Several advantages of the nested process modeling are summarized as follows.

- Nested process modeling is a typical top-down approach that is generally conceived as an easy way of designing and analyzing even a very complex process.
- It provides a theoretical background of taking advantage of the object-oriented approaches, such as encapsulation, polymorphism, and inheritance, for modeling processes and developing workflow management systems.
- Frequently used process models can be provided as a form of library, and this increases the reusability of process models and thus reduces the efforts needed to design processes.
- Since each process is an autonomous unit that is manageable and controllable in distributed environment, it forms the basis of encapsulation during run time.
- The nested model can be easily extended to a model for interoperability between heterogeneous and distributed workflow systems.

4.2 Attributes and Behaviors of the Process Model

The attributes and behaviors of tasks and processes are described here. The component modules of the system interact by exchanging messages (return codes) and data. We basically follow a similar approach with those previous researches[1][8]. Task status types determined by the return codes from normal clients are as follows.

- NotReady: Initial state
- Ready: A task is ready to begin, but not executed yet.
- Doing: A task is currently being executed.
- Done: A task is normally terminated.
- Suspended: A task is temporarily stopped.
- Rejected: A task is rejected.
- Aborted: The whole process is cancelled.
- Failed: A task is failed, which can start an alternative task.

Since, in the nested process model, tasks and processes are dealt with in an identical way, both of them have the same status type. All tasks are initialized with the status of 'NotReady'. If all the preceding tasks of a task have completed, the status changes to 'Ready'. When the workflow engine assigns a 'Ready' task to a task performer, its status becomes 'Doing'. Completion of the task changes the status depending on the replied return code. The state transition diagram is similar to other workflow systems. The final task status can be one of 'Done', 'Failed', and 'Aborted.'

A process flow can be either serial or parallel. For the parallel process flows, we consider the following four types of split, each of which can be associated with the same type of merge. The split and merge type is specified for every task.

- Concurrent split [merge]: The end of a concurrent split task triggers all of its immediately succeeding tasks in

parallel. On the other hand, to trigger a concurrent merge task, all of its immediately preceding tasks should be terminated.

- Alternative split [merge]: The end of an alternative split task triggers all of its immediately succeeding tasks in parallel. This is identical to the concurrent split. The difference is, an alternative merge task can be triggered when at least one of its immediately preceding tasks is terminated.
- Exclusive-OR split [merge]: The end of an exclusive-OR split task selects only one of its immediately succeeding tasks. All the other tasks are ignored.
- Conditional split [merge]: The end of a conditional split task first checks the conditions to determine which task is to be followed. The conditions are specified as an attribute of conditional split task.

4.3 Run-time Encapsulation

In this section, the concept and mechanism of run-time encapsulation are described. Also, we explain how the Workflow Engine implements run-time encapsulation.

(1) Mechanism of Run-time Encapsulation

The developed system provides the capability of run-time encapsulation by using the nested process model in the previous section. As is already described, processes are nested when it is hard to define a process at one model in its entirety. In such a case, the process model is not only encapsulated at the design time, but the encapsulation can also occur during run-time. This is called the run-time encapsulation of sub-processes.

A sub-process for run-time encapsulation should be a nested one in the process model. The sub-process is a completely independent process model which can be executed by a different workflow engine. The run-time encapsulation requires that the sub-process have an interface to its nesting task. Therefore, the parent process model containing the nesting task defines the input and output of the sub-process. That is, the model need not have the features specific to the sub-process, which can be defined in a WFMS that executes the sub-process.

In fact, a process for run-time encapsulation is a unit of execution and control. When such a sub-process is called, it receives minimally required information including the input, due date, and output. In order to do this, the nesting task should know the location of the workflow engine that will execute the sub-process. Then, the control of execution and management of the sub-process is handed over to the engine, and this triggers the sub-process. It is possible that the WFMS does not have a process model for the sub-process. In this case, the model can also be delivered if they use the same format of process models. Otherwise, a new sub-process needs to be created. Still another possibility is that a sub-process model, which can either be an existing one or one that is received from the parent engine, can be freely modified at the child workflow engine. Since the encapsulation localizes the modification, it does not affect the parent process at all. Upon the completion of executing the sub-process, the engine notifies the results to the higher-level engine.

The run-time encapsulation provides several advantages, particularly for the distributed and heterogeneous computing environments. First, the encapsulation facilitates cooperation among different departments and organizations, since they are able to take part in the same business process. Second, the scalability of WFMS can be enhanced because distributed workflow engines can manage sub-processes independently. Third, the stability of WFMS would be increased. Since a complex process is separated into a set of smaller units of sub-processes, problems in any sub-process does not influence other processes. Fourth, it is easy to dynamically adapt process models since the modification of process definitions is localized. Finally, it provides a scheme of monitoring authority at various levels. Managers at a high level would be more interested in the abstract status of sub-processes than details in the low level. The monitoring of the low level processes by users is also possible if they have the proper authority.

(2) Implementation of Run-time Encapsulation

Workflow engine supports run-time encapsulation of nested process models. The procedure performed by the engine for handling nesting tasks is described as follows.

If a task is a nesting task, the procedure first checks a responsible engine that will execute the nested sub-process. The first case is that the engine running the procedure carries out the sub-process by itself. The procedure checks the availability of the sub-process model. If the model is available, the procedure creates an instance of the sub-process, gets its first task, and sets the task's status to 'Ready'. Since this task can also be a nesting one, the procedure calls the Handling_Nested_Process() function recursively. When the sub-process model is unavailable, a new model is designed and is executed.

The other case is the execution of the nested process at a remote engine. The procedure notifies the remote engine, and then the control is passed over to the engine with TaskInstanceID. While executing this sub-process, the run-time encapsulation is applied. To implement the mechanism, we have considered three coupling modes; tight coupling, intermediate coupling, loose coupling.

A designer specifies a relevant coupling mode as an attribute of every nesting task. When notifying a nesting task to a remote engine, its coupling mode is transmitted in the argument of 'cmode'. The other three arguments are 'id', 'pd', and 'ntinput', which respectively contains the nesting task identification, sub-process model, and other required information including input documents, due date, description, and so on. The transmission of sub-process model can be optional. This means that the remote engine can use either the sub-process model provided by the host engine, or its own model. It is also possible to create a new model and modify an existing one before launching the sub-process. On the completion of the sub-process, the remote engine returns the execution results.

If the task is a primitive one, the engine identifies a performer of the task. When the performer is defined as a group of people, the engine chooses one of them by calling a workload balancing procedure. Now, the task status is set to 'Doing', and the task is dispatched to the performer. At the same time, the engine sends an E-mail message to the performer to notify the arrival of task. This is implemented using Java Mail API.

5. System Usage

Workflow client users consist of build-time users and run-time users. Build-time users generate process model using Process Desinger. Run-time users participate in the system via the Web interfaces.

5.1 Build-time Usage

Process Designer is a program that provides the build-time function of workflow system. It has a GUI where a process designer creates and edits nested process models. The overall structure of a process can be viewed on the Process Hierarchy window in a tree structure. The Process Hierarchy window is similar to a folder structure. Selecting a nesting task in this window activates the diagram window containing the corresponding nested task. Other functions include error checking in process models, exporting completed models to databases, and importing sub-processes for nesting. The Process Designer is implemented in Visual C++, and the GUI components are developed using the class library of Objective Diagram, Stingray, Inc.

5.2 Run-time Usage

There are four client interfaces, which are the initiator, normal client, supervisory control and monitor, and system administrator. All the client modules are made of Java applets, which are developed using JBuilder 2.0 and JDK 1.2.

The initiator module is used to launch an instance of process model. The interface for normal clients is for a task performer to receive task information from a workflow engine and to respond the engine with task results. The module manages a list of tasks for each user. The system administrator module provides general functions of system administration, including management of Users, Role, and Authority.

With supervisory control and monitor module, a supervisory manager can monitor the state of processes that are currently being executed, and place a proper order of expediting, suspending, resuming, or aborting. It also provides statistics on finished processes, including overall performance, task history, workloads for each task performer, and so forth. Only authorized users can use these services.

A zoom in/out function is provided for the monitoring services. A high level manager is generally interested in abstracted information about the whole process. On the other hand, a low level manager would require more detailed information concerning a certain sub-process. The run-time encapsulation proposed in this paper supports this modularized monitoring service. Fig. 4 shows a monitoring applet. A Process Hierarchy tree is popped up on the window at the left hand side. A user can select a sub-process that the user wants to monitor. In our example, the user selects a sub-process executed by a remote engine. The information on the selected sub-process is instantly displayed on the main monitoring window. It includes general process descriptions, engine location, coupling mode, input and output packets. The sub-process model can also be presented in another window as shown at the bottom of Fig. 4. According to the coupling mode, the abstraction level of monitoring and control is determined. Since the coupling mode is 'Tight coupling' in the example, the remote engine provides the status of output packet. Also supplied by the remote engine are the sub-process model, launch time, due time, and process managers. The user can intervene the execution of the sub-process by clicking one of the supervisory control buttons at the bottom of the main window. If other coupling modes are used, the buttons are disabled.

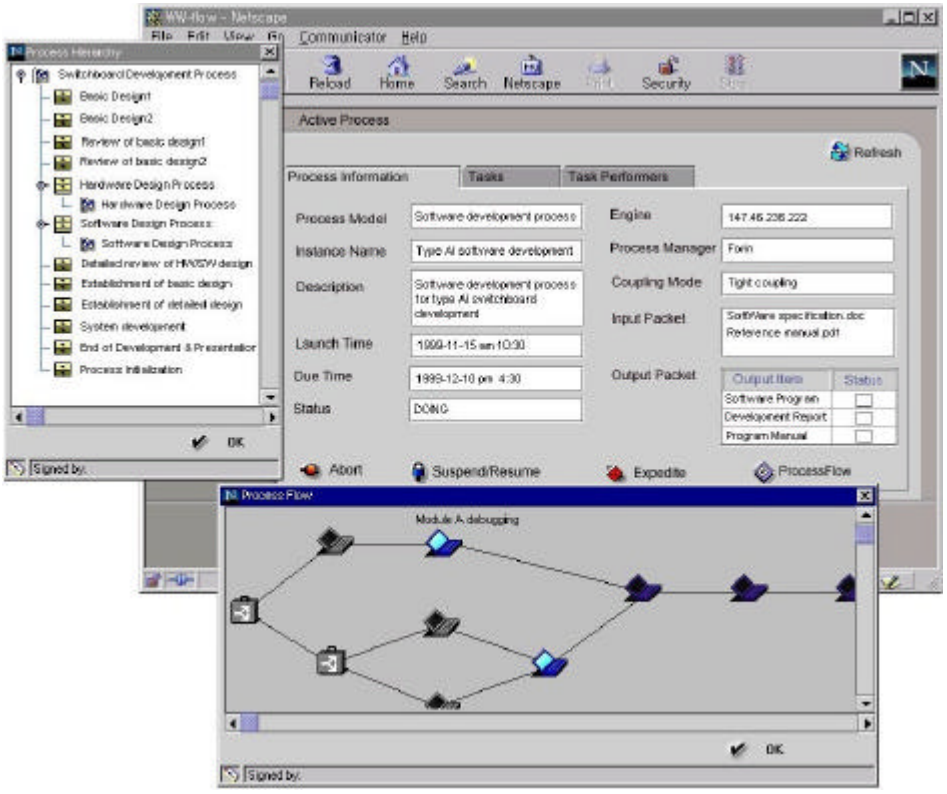

Fig. 4 Remote sub-process monitoring [9]

6. Conclusions

In this paper, Run-time encapsulation mechanism for managing complex business processes is proposed. The design and development of a web-based WFMS for implementing the concept is described. Due to the advantages that a workflow management system provides, it is regarded as an essential module to many recently emerging enterprise information systems. Our goal is to provide an effective management of complex business processes across distributed and heterogeneous environments. Several requirements are identified for the distributed and heterogeneous WFMSs. Nested process modeling plays a key role in satisfying the goal. Taking advantage of the top-down design concept, object-orientation, and process reusability gives an easy way of modeling complex business processes during the designing phase. More importantly, this approach supports run-time encapsulation. A nested sub-process is a unit of execution and control, which can be executed independently by another WFMS. It improves the process adaptability, scalability, stability, cooperation, and monitoring effectiveness of WFMSs. The server is implemented in pure Java, and thus it can be readily mounted on any platform. The client modules, all developed using Java applets, give users location transparency.

We have reached to a conclusion that, for a complete interoperability among heterogeneous computing environments, it is very important to have a standard format of process definitions. We believe that the concept of nested process modeling is to be included in the standard. The intended goal of our research is partly achieved since it interacts only between servers using the engine we have developed. The interoperability requires that all workflow engines involved in a process support a common set of APIs. One alternative is using the SWAP compatible process definitions in XML. Another important issue for further research is to incorporate the WFMS into such information systems as enterprise resource planning, supply chain management, and order tracking system, so that they are interoperable.

ACKNOWLEDGEMENTS

This research was supported by the Korean Science and Engineering Foundation (KOSEF) under grant no. 97-02-00-09-01-3.

References

[1] A. Kumar and J. L. Zhao, “Dynamic Routing and Operational Controls in Workflow Management Systems,” Management Science, Vol. 45, No. 2, pp253-272, February 1999

[2] D. Hollingsworth, “The Workflow Reference Model,” Workflow Management Coalition Specification, WfMC-

TC-1003, Jan. 1995

[3] F. Casati, S. Ceri, B. Pernici, G. Pozzi, "Deriving Active Rules for Workflow Enactment," Proceedings of 7th International Conference on Database and Expert Systems Applications, Springer-Verlag Lecture Notes in Computer Science, pp94-110, 1996

[4] G. A. Bolcer and G. Kaiser, "SWAP: Leveraging the Web to Manage Workflow," IEEE Internet Computing, Vol. 3, No. 1, pp85-88, 1999

[5] J. D. Davidson, "JavaTM Servlet API Specification, v2.2," Sun Microsystems, Inc., November 1998

[6] J. Lee, M. Gruninger, Y. Jin, T. Malone, A. Tate, and G. Yost, "The PIF Process Interchange Format and Framework," PIF Working Group, May 24, 1996

[7] J. Miller, A. Sheth, K. Kochut and X. Wang, "CORBA-Based Run-Time Architectures for Workflow Management Systems," Journal of Database Management, Special Issue on Multidatabases, Vol. 7, No. 1, pp16-27, 1996

[8] J. Miller, D. Palaniswami, A. Sheth, K. Kochut, and H. Singh, "WebWork: METEOR's Web-based Workflow Management System," Journal of Intelligent Information Management Systems, Vol. 10, No. 2, pp185-215, 1997

[9] Y. Kim, S.-H. Kang, D.S. Kim, and J.S. Bae, "WW-flow: A Web-based Workflow Management System With Runtime Encapsulation," IEEE Internet Computing, Vol. 4, No. 3, 2000