

A Framework Supporting Dynamic Workflow Interoperation and Enterprise Application Integration

Myungjae Kwak

*Electronics and Telecommunications Research
Institute*

*161 Gajeong, Yusong, Taejon, 305-350,
South Korea*

mjkwak@etri.re.kr

Dongsoo Han, Jaeyong Shim

School of Engineering

Information and Communications University

*Yusong P.O. Box No. 77, Yusong, Taejon,
305-600, Korea South*

dshan@icu.ac.kr, jaeyong7@icu.ac.kr

Abstract

In this paper, we propose a framework supporting dynamic interoperation between heterogeneous workflow systems and workflow-based dynamic Enterprise Application Integration (EAI). To this end, based on Subflow Task Model and Multi-Tiered Dynamic State Transition Model, four major components are defined: The workflow engine, Adapter, Service Interface Repositories (SIRs), and XML messages. Workflow engine provides user for location transparency of sub-processes by encapsulating and dynamically binding both internal and external sub-processes. As a middleware, the adapter achieves location and system transparency with the help of the workflow engine by encapsulating and dynamically binding external sub-processes to its super-process. SIRs contains the service information of other systems. The Local SIR (LSIR) within an organization is an important component for dynamic EAI. The adapter looks up the LSIR at run time when it tries to find external services. Several XML messages enable the communications between heterogeneous workflow systems and enterprise applications.

Keywords: Dynamic Workflow Interoperation, Service Interface Repository, Wf-XML, Heterogeneity, Enterprise Applications Integration.

1. Introduction

In recent years, the requirements of interoperation between Workflow Management Systems (WfMS) crossing organizational boundaries and Enterprise Applications Integration (EAI) have increased. Especially in B2B electronic commerce, the requirements are more impending because most of the E-commerce processes related to many other companies' workflow systems and legacy enterprise

applications, such as Enterprise Resource Planning (ERP) systems, Customer Relationship Management (CRM) systems, and accounting management systems, cannot be completed within a workflow management system.

When a customer tries to purchase goods from the Internet portal, the Internet portal's business process would be started. If the goods purchasing process is associated with many business partners, such as publishers, credit companies, delivery companies, as well as the local enterprise applications, it cannot be completed without receiving completion notification from the partners and enterprise applications. This implies that inter-workflow system cooperation and EAI mechanism should be prepared for the complete support of E-business process. [23]

There have been several research works done to resolve the heterogeneity and improve the interoperability [5][16], and many WfMS vendors have tried to make their WfMSs interoperable with other WfMSs. To accomplish this mission, the Workflow Management Coalition (WFMC) has made considerable efforts and recently it released Wf-XML specification of standard message format for interoperation. The Wf-XML specification certainly improves the capabilities of the e-mail MIME binding specification. It provides a structured and well-formed XML body protocol that consists of message structures containing headers and data. It also has synchronous or asynchronous message-handling capability, independence from the transport mechanism and platform, and easy extensibility by using XML and dynamic workflow context data [3][24].

Many commercial WfMSs have their own mechanisms to interoperate with other WfMSs and to interact with enterprise applications. However in most of the WfMSs, user has to statically predefine every detail of the workflow process to make them interoperable with each other. This constraint makes the business process template very complicated when it has multiple alternative paths that can

be selected based on context data. For example, multiple business partners should be connected according to the kind of the goods, in the goods purchasing process.

On the other hand, businesses now find themselves in an environment that changes greatly and continually. Business information systems operating under these circumstances are expected to be able to cope with changes in the market both promptly and flexibly. Therefore, whenever new WfMSs or enterprise applications join or current WfMSs or enterprise applications change their service interfaces, the process should be redefined to get continuous services from other WfMSs and enterprise applications. If we make these changes transparent to the user or service requesting entity, we can expect many advantages. It can improve flexibility, scalability, and interoperability and the WfMSs can cooperate with other WfMSs and enterprise applications more effectively.

To this end, we propose a framework for dynamic interoperation between WfMSs and workflow managed dynamic integration for enterprise applications based on the subflow task model and the multi-tiered dynamic state transition model. Four main components are defined for the proposed framework: Workflow engine, Adapter, Service Interface Repository, and standard XML messages. The workflow engine and adapter make local WfMS transparent to the location and platform of the target system by encapsulating sub-processes in WfMSs or enterprise applications. If a service cannot be carried out by the local workflow system, the workflow engine asks an adapter to search the service from other workflow systems and enterprise applications. The adapter handles external sub-processes on behalf of the local workflow system and internal processes on behalf of the external workflow systems.

By integrating these four components a workflow system can cooperate with other workflow systems and enterprise application to complete an E-business process without the detailed information of external systems. We call this type of interoperation *dynamic workflow interoperation and EAI*. Dynamic workflow interoperation and EAI means that external sub-processes are determined and bound to the main process at run time. The proposed framework is adopted and evaluated based on simple scenarios by using our workflow system, ICU/COWS [1].

The remainder of the paper consists of four major parts. Next section describes efforts for interoperation among workflow systems. In the third section, we discuss the requirements of dynamic workflow interoperation and EAI. The fourth section shows the proposed framework including design principle, overall architecture, workflow engine, SIR, adapter, and Wf-XML messages. In the fifth section, the example application and evaluation of the proposed framework are discussed. The conclusion is discussed in the last section.

2. Related Works

Many WfMSs have been designed and implemented to support the interoperability in a distributed heterogeneous environment [3][9][18][21][22]. In this context, the WfMC defines the mechanisms that workflow system vendors have to implement so that their systems may request another workflow system to select, instantiate, and enact process definitions. WfMC recommends that the requesting workflow system also be able to pass context data that consists of workflow relevant data or application data and receive back status information and the results of the enactment of the process definition, which can be done in a way that is as transparent to the user as possible. WfMC identified several models of interoperability, scopes of workflow interoperation, and levels of the each model of interoperation [17].

The WfMC released two protocol specifications for interoperation standard (Interface 4). One is e-mail MIME binding that uses asynchronous interaction via an e-mail as the transport with MIME encoding. The other is Wf-XML, which uses synchronous interaction via an XML. The WfMC also released the Wf-XML specification providing some other enhanced capabilities referring to the SWAP (Simple Workflow Access Protocol) proposed by IETF in May 2000 [5][16]. By exploiting workflow interoperability based on Wf-XML, it becomes straightforward to integrate both process and application data into an overall DTD for transport through the messaging service. The use of global process identifiers (URI) across a business community will facilitate monitoring and controlling between trading partners [3].

The OMG Workflow Management Facility standard (known as the jointFlow specification) is based on WfMC standards [3]. The specification defines an object-oriented framework and interfaces for distributed workflow systems. It enables workflow process execution control, monitoring, and interoperability between different workflow systems [11].

CrossFlow was proposed by the German National Research Center for Information Technology (GMD) to support interoperability between WfMSs. Seven European institutes formed the consortium of the CrossFlow ESPRIT project. This project combined advanced process support in dynamic virtual organizations with contract-based service trading. CrossFlow introduced the service-oriented model whose central concept is service that is comprised of the service provider and the service requester. This project focuses on how the service contracts between trading partners are established [4][19].

The WISE project proposed a WWW catalogue and a business-modelling tool to achieve inter-organizational workflow called virtual enterprise [25]. The main idea is to integrate the services of the different companies with the

same manner within a single corporation. To make this idea a reality, WISE proposes a mechanism for the participants to publish their services. A user can find a service through the WWW catalogue and define a process based on the WWW catalogue.

Fabio Casati [20] presented a workflow event model that allows flexible interoperation among processes in different organizations in the WIDE project [26]. He extends the traditional workflow model by enabling the traditional workflow to publish and subscribe to events. The event dispatching mechanism that he proposed treats the events according to the priorities of event publication, business convention, cost, etc.

M. Oba et al. [28] proposed a new type of workflow for EAI. The new workflow for EAI uses a CORBA interface to integrate different types of systems and uses a three-tiered state transition model based on a database to dynamically control processes.

J. Mann [27] supposed that EAI varies greatly in scope from a simple task like merging two applications together, to a long-range, global project involving many applications and automating critical business processes and defined three levels of integration to represent the variation in scope: direct, brokered, and workflow. A direct integration combines the applications themselves without any additional software. If a message broker is used, applications send and receive messages from the broker only. Workflow is useful in the situation that tasks involve teamwork and unique conditions and require complex business logic.

3. Requirements of Framework Supporting Dynamic Workflow Interoperation and EAI

According to the selecting time of the target cooperating workflow system, the interoperations between workflow systems can be classified into two types: *Static* and *Dynamic*.

(1) *Static interoperation*. The business process designer predefines every detail of potentially connectable workflow systems that perform nested sub-processes or chained processes. In this case, workflow process execution is supposed to follow the pre-defined path and thus it is not easy to add a process or change the configuration of the process definition at run time.

(2) *Dynamic interoperation*. The binding of the sub-processes of target workflow systems is performed at run time according to special binding rules. This form of interoperation brings several advantages over static interoperation. That is, the process designer needs not predefine all of the potentially connectable sub-processes, and the process definition becomes simpler. Moreover, the configurations of service interfaces can be changed and new systems can be added easily, even while the main

process is running. Therefore dynamic interoperation is a more flexible and scalable form of interoperation than static interoperation.

EAI is defined as the process of integrating multiple applications that were independently developed, that may use incompatible technology, and that remain independently managed [27]. As large enterprises move toward integrating existing applications and developing new ones, middleware that provides a coherent and easy to use integration capability is increasingly important. A workflow can be defined as the automation of a business process, as a whole or part, in which work items are passed from one participant to another for action, according to a set of procedural rules [11]. In this context, a workflow system becomes an important element of EAI because the workflow system can coordinate and control the flow of events among applications. This form of EAI is called workflow-based EAI.

Similar to workflow interoperation, according to the selecting time of the target enterprise application, the workflow-based EAI can be classified into two types: *Static* and *Dynamic*.

(1) *Static EAI*. The business process designer predefines every detail of potentially connectable enterprise applications. In this case, the workflow process execution is supposed to follow the pre-defined path, and thus it is not easy to add a process or change the configuration of the process definition at run time.

(2) *Dynamic EAI*. The binding services of target enterprise applications are performed at run time according to special binding rules. This form of integration brings the same advantages as dynamic interoperation.

According to the location of the system by which the sub-processes are controlled, those sub-processes are classified into two types. Both sub-processes should be uniquely identifiable with a service name and a process name to dynamically bind them.

(1) **Internal sub-process** means a sub-process that consists of one or more tasks and is controlled by the local workflow system where the main process is running.

(2) **External sub-process** is a sub-process that consists of one or more tasks and that is controlled by other workflow management systems or enterprise applications through the adapter. Enterprise applications can be treated as other workflow management systems because the two systems are alike in the context in which they provide a certain service for the local workflow management system, even though they can be accessed by different protocols and interfaces.

Figure 1 shows a business process that is related to several other systems. When a customer raises an order through the business portal website, the main process is started. First the CRM process within the CRM system, which is an enterprise application, is invoked to store the customer's information and use the data for marketing. As

business environments and marketing strategies now change greatly everyday, the CRM process can be changed frequently. This means that static EAI costs more than dynamic EAI to make the changes effective.

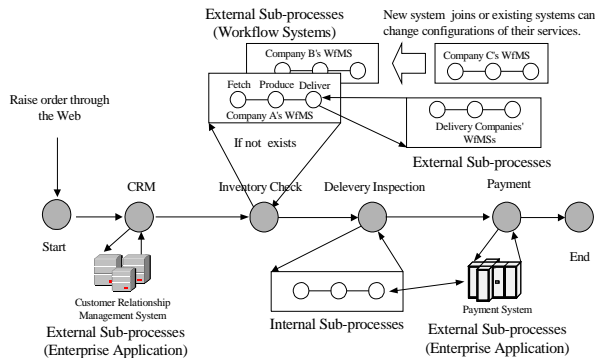


Figure 1. An example of the e-Commerce process

For the next task, if the portal doesn't have any inventory of the order, the local workflow management system orders the goods from trading partners and then the process in the trading partner's workflow system would be started. In this case, a lot of trading partner's workflow systems may be related to the main process and moreover those trading partners' systems may be related with other trading partners' systems. This means that if the workflow designer predefines all of the connectable of sub-processes of related workflow systems in a business process template as conventional systems do, the main process may become too complex. Moreover, if we modify the main process, whenever a new system joins or some systems change the configuration of their services, it is extremely inefficient.

Those problems can be solved by dynamic workflow interoperation and workflow-based dynamic EAI, which can be achieved by encapsulating the sub-processes, binding target sub-processes at run time, and delegating the service to the selected system based on subflow task model and multi-tiered dynamic state transition model.

4. Framework of Dynamic Interoperation and EAI

4.1. Design Principle

The goal of this paper is to propose a framework so that the service requester (local workflow system) needs not care about the way that service delegation is carried out at run time. That is, service providers (other workflow systems or enterprise applications) binding, proper sub-process selection among the candidates, and request message sending to the selected workflow systems or enterprise applications should be performed transparently to the service requester.

In achieving the goal we propose a framework supporting dynamic workflow interoperation and EAI. The

framework can be accomplished based on a subflow task model, a multi-tiered dynamic state transition model, and a cached information inconsistency handling model by using object-oriented concepts such as encapsulation, dynamic binding, and delegation.

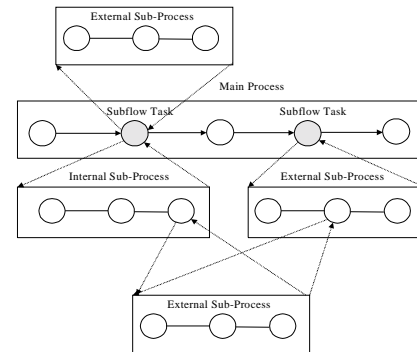


Figure 2. An example of subflow task model

4.1.1. Subflow Task Model

A workflow process is defined by the automation of procedures where information and tasks are passed between participants according to a defined set of rules to achieve, or contribute to, an overall business goal [11].

As the object-oriented hierarchical process structure describes in Figure 2, if a task (subflow task) can represent many alternative internal or external sub-processes that are explained in Section 3.3, we can expect many advantages. The main process definition can be much simpler, and moreover, if the sub-processes are bound to the main process not at build time but at run time, we can expect benefits of dynamic workflow interoperation and EAI. In this light, the subflow task model is the base of our framework. The following are the benefits of dynamic sub-process binding.

- Process definition is much simpler
- Reusability of frequently-used process definition is much improved
- Some parts of a business process definition can be omitted from the main process definition.
- Easy reconfiguration of the sub-processes
- Easy selection of appropriate sub-processes at run time among many internal sub-processes or external sub-processes according to the requirements. This improves scalability and flexibility.

4.1.2. Multi-Tiered Dynamic State Transition Model

The workflow enactment service may be considered as a state transition machine, where individual process or activity instances change states in response to external

events (for example, completion of an activity) or to specific control decisions made by a workflow engine (for example, routing to the next activity step within a process).

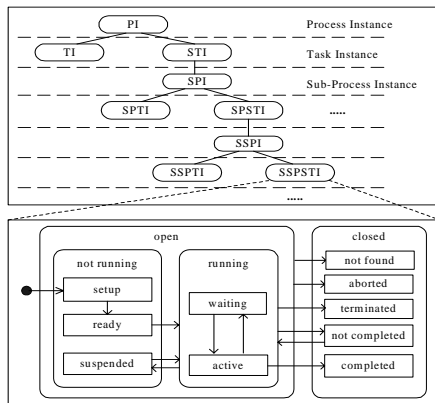


Figure 3. Process and task state transition model

The framework supporting dynamic workflow interoperation and EAI is based on the hierarchical subflow task model, where a task within a process can represent many alternative sub-processes that consist of one or more tasks. Moreover, those sub-processes can also contain subflow tasks that can represent many alternative sub-processes and control decisions for selecting proper sub-processes are made at run time. This means that many sub-processes can be hierarchically bound in a process and the proposed transition model must be able to represent the states of the dynamically created sub-process instances.

To handle such situations, the proposed framework uses a multi-tiered dynamic status transition model. The status transition model does not predefine how many sub-process instances and sub-process task instances can be created, but dynamically determine them according to the context data of the task. Figure 3 shows the hierarchical multi-tiered state transition model. The transition diagram is applied to all kinds of process instances and task instances, and the states of the newly created sub-processes and task instances are created and maintained dynamically.

4.1.3. Cached Information Inconsistency Handling Model

When a target workflow system performs a service, various exceptions can occur if SIR doesn't have any information of the requested service or the information of service requester is different from the target systems' service interface. The information inconsistency occurs because the proposed framework maintains two types of SIR (LSIR and GSIR) for the dynamic workflow interoperation and EAI and the information of external services in LSIR is cached information. There can be three possible situations for all of the exceptions, and, among

those three situations, 'not updated' and 'update but inconsistent' may occur because of the cached information inconsistency.

- (1) **Not available.** This is the situation in which when adapter searches information of a service, there is no matched information for that service.
- (2) **Not updated.** This is the situation in which workflow systems and enterprise applications changed configurations of their local service interfaces but didn't update either the LSIR (enterprise applications) or the GSIR (workflow systems within other organizations) with the changed information.
- (3) **Updated but inconsistent.** This means that workflow systems within other organizations changed configurations of their local service interfaces and also updated the GSIR, but LSIR was not updated with the changed information.

To handle these exception and inconsistent situations, there can be three possible states as shown in Figure 3.

- (1) **Not Found.** This is the situation where the information of a service is not available. In this case, the local workflow engine waits for the administrator's handling or search another service interface based on specific rules.
- (2) **Aborted.** This is the situation where the service providing system decides that it cannot perform the requested service since the service is replaced by another service with the same interface or other problems occur. In this case, the local workflow engine waits for the administrator's handling or search another service interface based on specific rules.
- (3) **Terminated.** This is the situation where the services are identical but some of the tasks within the requested sub-process are omitted through Business Process Reengineering (BPR). The service providing system aborts the omitted tasks' instances and completes the sub-process.
- (4) **Not completed.** This is the situation where the services are identical but the target system aborts the running process, starts again with a changed template, and completes the substitution.

4.2. Overall Architecture of Framework

The proposed framework supports the dynamic interoperation between workflow systems and workflow-based dynamic enterprise application integration in distributed heterogeneous and Internet environments. In this framework, four features, workflow engine, adapter, LSIR/GSIR, and Wf-XML messages, serve important roles.

Figure 4 shows the overall structure of the framework. Service providers, which want to announce their services, can register, update, and delete their service information by sending XML messages to the GSIR but enterprise applications register and update only LSIR with their service interfaces. Whenever the service information of

GSIR is changed, it broadcasts the information to the other organizations' LSIRs. Of course, the LSIRs should continue to poll the GSIR to keep their data consistent with GSIR's data. Since a LSIR has the data about enterprise applications as well as the data in GSIR, LSIR acts as a repository for workflow-based dynamic EAI.

If a workflow system performs a subflow task in the middle of the execution of a workflow process, the workflow engine would determine whether or not the local system could provide the service that the subflow task represents. In this light, the workflow engine should have flexible and scalable architecture for selecting and binding sub-processes at run time. Once the workflow engine finds that the local system cannot provide the service, it requests the adapter to find an appropriate external service from the external services repository pool and delegates the service to it. In this context, the workflow engine provides user for transparency about sub-processes because the user or service requester can consume a service only if it requests the service with the service name and process name without knowledge of the location and system of the target sub-process.

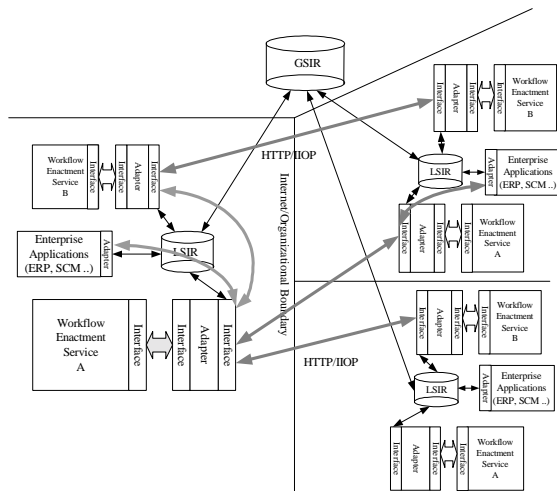


Figure 4. Overall architecture of interoperable ICU/COWS

The adapter selects an appropriate sub-process by searching LSIR with a given service name and process name. Once the serviceable workflow system or enterprise application is found, the adapter asks the selected system to provide the service by sending XML messages. This means that the adapter provides location transparency about external sub-processes. Therefore this framework allows the service requester to use all the external sub-processes like an internal sub-process regardless of organization, location, and platform.

XML messages are used to communicate with other workflow systems and enterprise applications and to access GSIR to update LSIR. Even though Wf-XML messages can

be transmitted over various protocols, some of them are not proper for some situations because of their characteristics. For instance, when an external sub-process is placed at a remote site, HTTP is better than other protocols because HTTP, which is a ubiquitous protocol in the Internet, is a lighter weight protocol than IIOP. The SMTP is not applicable because it cannot support synchronous operations.

4.3. Workflow Engine

The workflow engine serves important roles in this framework. It should be able to decide whether or not a service can be performed by the local system. Once it decides that the local system cannot provide a service, it creates a workflow manager for managing the external sub-process and delegates the service to the adapter for starting the external sub-process.

4.3.1. Key Classes of Workflow Engine

The workflow engine consists of five key classes. Among the five classes, *Workflow Requester*, *Global Manager*, and *Local Factory* are instantiated at system setup time. *Workflow Manager* and *Task Manager* are instantiated at run time responding to the requests from the instance of *Workflow Requester* [13]. The functionalities of each class are as follows.

- **Workflow Requester (WR)** plays the role of an interface between workflow participants and a workflow engine for creating a process instance or between Subflow *Task Manager* and *Global Manager* for creating a *Workflow Manager* and sub-process instance. In a case where the local system provides service for the external system, the *Adapter* uses WR as an interface to *Global Manager* and *Workflow Manager*.

- **Global Manager (GM)** manages the workflow engine and controls the creation of *Workflow Manager* and *Task Manager*. It delivers the creation request from WR to an appropriate *Local Factory*. It also decides the type of *Workflow Manager* based on the context data containing the service name and the process name from Subflow *Task Manager* by retrieving internal service interface repository. If an appropriate process does not exist inside the system, it requests creating an external *Workflow Manager* to *Local Factory* and sends the context data to the created external *Workflow Manager*, which manages the external sub-process.

- **Local Factory (LF)** creates an instance of *Task Manager* and *Workflow Manager* according to the creation request from the GM. It contains information of instances of *Task Manager* and *Workflow Managers* that it has created for load balance. In the case of an external sub-process, it only creates *Workflow Manager*, which manages the external sub-processes.

■ **Workflow Manager (WM)** is responsible for managing *Task Managers* that are for tasks within a workflow process. It is not concerned about task specific data but data in the workflow process level such as workflow process name, number of tasks, references of *Task Managers*. WM has two different types: internal WM which manages internal processes and external WM which manages external processes. External WM uses the Adapter to manage the external workflow process.

■ **Task Manager (TM)** is responsible for managing a task within a workflow process. TM has all the required data for executing its managing task. TM is classified into Subflow TM, Application TM, Loop TM, and None TM.

■ **Adapter** is a proxy gateway to communicate with external systems and is not an engine component. It decides the proper external sub-process based on context data from WM by searching LSIR. In the case of the service provider, the Adapter requests WR to create WM and process instance and it communicates with the created WM on behalf of the external system.

4.3.2. Control Flow of Workflow Engine

First a client or service requester uses WR to start a main workflow process. WR delegates this request to GM and GM requests LF to create a GM and TMs based on the main process template. After the setup procedure, the main process is started. If a Subflow TM takes over the control of the process in the middle of execution, the same setup procedure would be repeated. That is, the Subflow TM requests WR to create an instance of proper sub-process at run time based on a given service name and process name without detailed information of the target sub-process. GM receives this request from the WR and decides whether or not the service can be provided by the local workflow system. If the local system can provide the service, it requests LF to create internal WM and its TMs. If not, it requests LF to create only external WM. After this repeated setup procedure, the Subflow TM asks either the internal or the external WM to start its sub-process. In this case, external WM requests the adapter to execute the external sub-process.

4.4. Adapter

In an object-oriented concept, delegation is a more general way of extending a class's behavior that involves one class calling another class's methods rather than inheriting them [8]. As a mediator to delegate a service from the local system to other systems, the adapter is designed to be connectable to any workflow systems as a middleware by using the delegation concept. Usually the adapter object is an object that receives method calls on behalf of another object. In our framework, an adapter object sends and receives XML messages on behalf of the

workflow engine. That is, the adapter encapsulates other workflow systems and enterprise applications and dynamically binds them to the local workflow system. The Wf-XML processor within the adapter translates the programming language level method invocations to Wf-XML messages and vice versa.

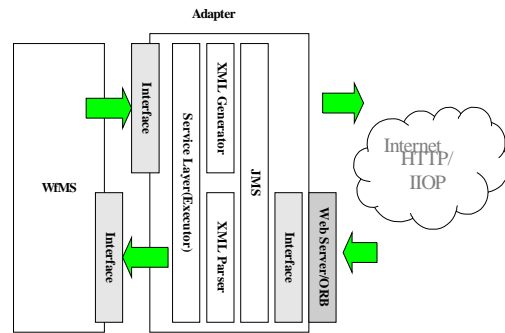


Figure 5. A simplified architecture of adapter

The adapter has internal and external interfaces, a service layer, an XML processor, a Java Messaging Service (JMS), and a web server. Internal interfaces are defined for the workflow engine. External interfaces are defined for the external workflow systems and enterprise applications. The service layer consists of executors' instances. The executor manages request and response XML messages. Figure 5 shows the conceptual view of the adapter, web server, and workflow system.

In the viewpoint of target workflow systems or enterprise applications, target workflow systems or enterprise applications can construct their own adapters. Those adapters must have facilities that can process standard XML messages and translate the passing data to their own data format.

4.5. Service Interface Repository (SIR)

SIR is a storage facility that contains service information of workflow systems. Many EDI frameworks have similar facilities that include information from other EDI systems. For example, ebXML and RosettaNet run a service repository and master dictionary. As shown in Figure 6, in our framework, instead of the simple repository, to reduce the overhead of information retrieval using XML messages and prevent system failure, the SIR is divided into two levels: GSIR and LSIR. The LSIR that is managed by LSIM is placed within an organizational boundary. The fact that LSIR keeps service interface information of enterprise applications for workflow-based dynamic EAI as well as information of GSIR allows the proposed SIR to have more benefits than other EDI frameworks' SIR. The local services of LSIR present the information of the enterprise application.

When the main parts inventory management process is started, the workflow system A requests the adapter to check the inventory of parts, and the adapter searches the information of the inventory management system that is an enterprise application from the LSIR and asks the system to gather information of the required parts. When the information of required parts are gathered, the third task's manager requests the adapter to purchase the required parts, and the adapter also searches an appropriate workflow system within the organization from the LSIR and asks the system to purchase the required parts. To handle the request, the parts purchasing process within workflow system B is started, the subflow task, that raises actual supply orders, requests the adapter to provide the required parts, and the adapter searches available suppliers' workflow systems from LSIR and asks them provides the required amount of parts.

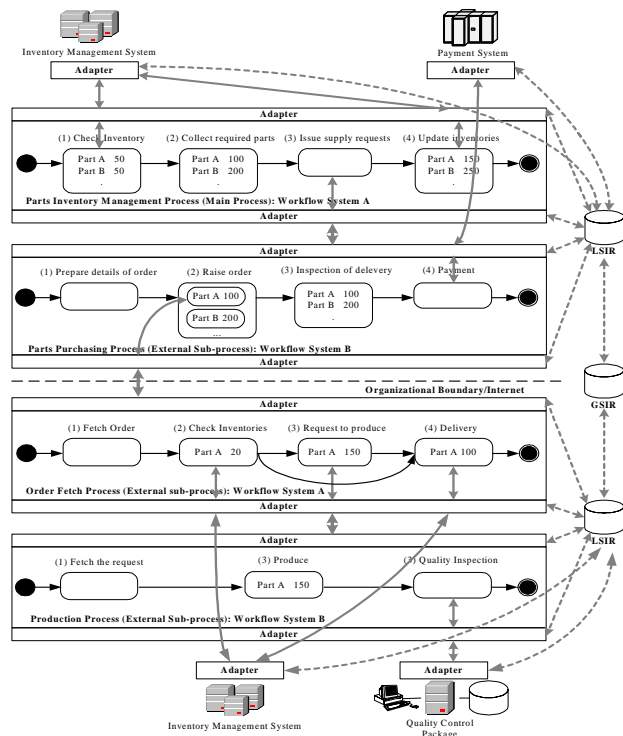


Figure 7. Parts inventory management process of a motor company

Similar to the workflow systems of the motor company, the workflow systems within the parts makers use other workflow systems or enterprise applications to serve the request.

5.2 Evaluation of the Proposed Framework

If the process designer should have to predefine every potentially connectable sub-process in the main business

process as if other conventional workflow management system does, the main process might be very complicated and the user should redefine the process whenever new systems join or some systems change their service information. Heterogeneity among workflow systems and enterprise applications is another hurdle.

To handle such situations, the integrated framework supporting dynamic interoperation among workflow management systems and workflow-based dynamic EAI is proposed based on the subflow task model and the multi-tiered dynamic state transition model using objected-oriented concepts such as encapsulation, delegation, and dynamic binding. The subflow task model makes the main process simpler by separating a complicated process into several sub-processes and allows the workflow systems and enterprise applications to add and maintain their services easily. The multi-tiered dynamic state transition model allows the local workflow system to monitor the state transition of the target systems. Based on the subflow task model and state transition model, four major components are defined for the proposed framework: a workflow engine, an adapter, LSIR/GISR, and XML messages.

The four components allow this framework to be effective even if the target service providers' systems are located far from the service requester's system and the platform of the target system is different from that of the service requester's system. The framework also allows new system to join easily and the main process can reflect the changes of service interfaces. This means that flexibility and scalability are much improved.

On the other hand, the proposed framework is built by using the adapter, which is a kind of Message-Oriented Middle (MOM). Therefore, the proposed framework is appropriate for loosely coupled interoperation and integration among heterogeneous workflow systems and enterprise applications. Among the four models of interaction between workflow systems, chained processes and nested sub-processes interoperation can be achieved.

6. Conclusion

Peter Wegner stated that the interoperability of two or more software components is a scalable form of reusability [12]. In the same context, the interoperability of heterogeneous workflow systems might also be defined as a scalable form of reusability and inter-organizational cooperation, because it gives more chances to one workflow system to make reuse of another workflow system's functionality to finish a business process.

In this paper, a framework supporting dynamic interoperation among workflow management systems and workflow-based dynamic EAI is proposed based on the subflow task model and the multi-tiered dynamic state transition model using objected-oriented concepts such as encapsulation, delegation, and dynamic binding. The four

components are defined for the proposed framework: workflow engine, adapter, SIRs, and Wf-XML messages. They increase scalability, flexibility, and interoperability of a workflow system by encapsulating and dynamically binding sub-processes to the main process.

7. References

- [1] D. S. Han, J. Y. Shim, and C. S. Yu, "ICU/COWS: A Distributed Transactional Workflow System Supporting Multiple Workflow Types," *IEICE Transactions on Information and Systems*, Vol. E83-D, No. 7, July 2000.
- [2] H. Lieberman, "Using Prototypical Objects to Implement Shared Behavior in Object Oriented Systems," *Proceedings of OOPSLA*, Sept. 1986.
- [3] James G. Hayes et al., "Workflow Interoperability Standards for the Internet," *IEEE Internet Computing*, p. 37-45, May-June 2000.
- [4] J. Klingemann et al., "Deriving Service Models in Cross-Organizational Workflows," *Research Issues on Data Engineering: Information Technology for Virtual Enterprises*, 1999.
- [5] K. Swenson, "Simple Workflow Access Protocol (SWAP)," Internet Draft, 7 Aug. 1998; available online at (<http://www.ics.uci.edu/~ietfswap>).
- [6] L. Fischer and Workflow Management Coalition, "Workflow Handbook 2001," Future Strategies Inc. Book Division, 2000.
- [7] L. Heiko et al., "Virtual Enterprise Co-ordinator – Agreement-Driven Gateways for Cross-organizational Workflow management," *Proceedings of the International Joint Conference on Work Activities Coordination and Collaboration*, 1999.
- [8] M. Grand, "Patterns in Java: a catalog of reusable design patterns illustrated with UML," Wiley computer publishing, 1998
- [9] M. T. Schmidt, "The Evolution of Workflow Standards," *IEEE Concurrency*, vol. 7, no. 3, July-Sept. 1999.
- [10] Nobuyuki Kanaya et al., "Workflow Standard – Interoperability: XML-HTTP binding," Submission 2.0, Feb. 2000.
- [11] Object Management Group, "Workflow Management Facility Specification," Version 1.2, April 2000.
- [12] P. Wegner, "Interactive Software Technology," *Handbook of Computer Science and Engineering*, CRC Press, May 1996.
- [13] S. Lee et al., "A Pattern for Managing Distributed Workflows," 7th Pattern Languages of Programs Conference (PLoP 2000), Aug. 2000.
- [14] Simon S.Y. Shim, Vishnu S. Pendyala, Meera Sundaram, Jerry Z. Gao, "Business-to-Business E-Commerce Frameworks," *IEEE Computer*, p.40-47, October 2000.
- [15] Workflow Management Coalition, "Workflow Standard – Interoperability Wf-XML Binding," TC-1023, May 2000.
- [16] Workflow Management Coalition, "Workflow Standard – Interoperability: Abstract Specification," TC-1012, Oct. 1996.
- [17] Y. H. Kim et al., "WW-FLOW: Web-Based Workflow Management with Runtime Encapsulation," *IEEE Internet Computing*, p. 55-64, May-June 2000.
- [18] Yigal Hoffner, Heiko Ludwig, Ceki Gulcu, Pau Grefen, "An Architecture for Cross-Organizational Business Process," In *Proceedings of the Second International Workshop on Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2000)*. pp. 2-11, Milpitas, CA, June 8 and 9, 2000. IEEE Computer Society, Los Alamitos.
- [19] Fabio Casati, "Semantic Interoperability in Interorganizational Workflows," WACC workshop on cross-organizational workflows, San Francisco, CA, USA, February 1999.
- [20] Michael zur Muehlen, Florian Klein, "AFRICA: Workflow Interoperability based on XML-message," *Proceedings of the 1st International Workshop on Infrastructures for Dynamic Business-to-Business Service Outsourcing*. Stockholm, 2000
- [21] A. Dogac et al., eds., "Workflow Management Systems and Interoperability," NATO ASI Series, Springer-Verlag, Berlin, 1998
- [22] ebXML Technical Architecture Specification v1.04, 16 February 2001; available online at (<http://www.ebxml.org/specs/ebTA.pdf>)
- [23] R.J. Glushko, J.M. Tenenbaum, and B. Meltzer, "An XML Framework for Agent-base e-Commerce," *Comm. ACM*, Mar. 1999, pp. 106-116; available online at (<http://www.commerceone.com/xml/publications/framework.pdf>).
- [24] G. Alonso, U. Fiedler, C. Hagen, A. Lazcano, H. Schuldt, N. Weiler, "WISE : Business to Business E-Commerce," *IEEE 9th International Workshop on Research Issues on Data Engineering. INFORMATION TECHNOLOGY FOR VIRTUAL ENTERPRISES (RIDE-VE'99)*. Sydney, Australia, March 23-24, 1999
- [25] F. Casati, P. Grefen, B. Pernici, G. Pozzi, G. Sanchez, "WIDE Workflow Model and Architecture," April 1996 ; available online at (<http://dis.sema.es/projects/WIDE/Documents>)
- [26] J. Mann, "Workflow and Enterprise Application Integration," 2000; available online at (http://eai.ebizq.net/workflow/approaches_to_EAI_2.html)
- [27] M. Oba and N. Komoda, "Multiple Type Workflow Model for Enterprise Application Integration," *Proceedings of the 34th Hawaii International Conference on System Sciences*, 2001