USING COLLABORATION TASK IN *ORBWORK* ENACTMENT SYSTEM FOR *METEOR* WORKFLOW MANAGEMENT SYSTEM

by

ZHONGQIAO LI

B.S., Shanghai Jiao Tong University, China, 1996

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirement of the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2000

USING COLLABORATION TASK IN *ORBWORK* ENACTMENT SYSTEM FOR *METEOR* WORKFLOW MANAGEMENT SYSTEM

by

ZHONGQIAO LI

Approved:

____
Major Professor

____
Date

Approved:

____
Dean of the Graduate School

____
Date

DEDICATION

To Mom, Dad and Haibei

ACKNOWLEGEMENTS

This research project is part of the group effort for the METEOR Workflow research at the Large Scale Distributed Information System (LSDIS) Laboratory of the Computer Science Department at the University of Georgia. I would like to take this opportunity to thank Naval Research Laboratory (NRL) for the great cooperation efforts in this project and Iona Technologies for donating OrbixWeb software.

I am extremely thankful to my major advisor, Dr. Amit Sheth, for his invaluable guidance and encouragement throughout my entire research and academic years. I am also thankful to Dr. Krzysztof Kochut and Dr. John Millor for their constant advice and help with my thesis. I would also like to thank all the excellent people in the LSDIS Lab, especially Dr. Ismailcem Budak Arpinar, Dr. Mizuho Iwaihara, Zongwei Luo, Antonio Jorge Cardoso, and Yufeng Chen, whose help and encouragement have been a source of optimism whenever I am high or low. In addition, my special thanks are extended to all my friends for their kind support for me through the years in Athens, GA. Finally, I am sincerely grateful to my parents and Haibei who have always been encouraging and supporting me, and have lightened every day of my life. They are the reason I could success.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

It has been over ten years since the first workflow product was introduced. In the last ten years, many vendors have offered commercial, general-purpose workflow management systems to the market [KSM99, F95, GHS 95, SJo 96, A+ 97]. Meanwhile, lots of research institutions have been making research efforts in some workflow technology areas, such as workflow exception handling, adaptive workflows, transactional workflows and synchronization of concurrent workflows, to make workflow technologies mature.

In recent years, workflow technology has developed to some extend with those efforts and achievements, and workflow management systems (WFMSs) have been used by organizations that need to coordinate their processes. Nevertheless, those WFMSs are primarily focused on providing automation within organizational environment, especially on coordinating human activities, facilitating data management, document routing and reporting [SK97]. From our viewpoint, we believe work coordination, collaboration and information access are all keys to provide a comprehensive support for organizational activities. However, on the collaboration level, many systems do not perform well because current workflow systems are mainly process-oriented. They only define processes that integrate related tasks to execute in a predefined order. Those tasks in a process can only be coordinated by the WFMS, but cannot interact with other applications out of the workflow process, which is the principle function of collaboration.

As the Internet technologies grow fast, progress in video-conferencing and web-based collaboration has rapidly expanded the availability of collaboration systems and provided better opportunities for extending the collaboration feature in the workflow system. Recently, collaboration has become a critical issue for many organizations because the need for interaction across the organization boundaries has become more prevalent [C99]. Integration with collaborative software products in workflow systems provides means to support a highly interactive, collaborative workflow. The challenge in the collaboration area for organizations is how to effectively use the collaborative applications in their organizational process.

In this thesis, we focus our attention on the collaboration feature in the workflow context. We design the collaboration task model in WFMSs and implement a prototype of the collaboration task in *OrbWork* enactment system for *METEOR* WFMS integrating *CaTCH* (Collaborative and TeleConsulting for Healthcare) as a collaborative application (Both *OrbWork* and *CaTCH* were developed by LSDIS Lab, University of Georgia). This implementation provides the collaboration support in *METEOR* WFMS.

This chapter provides an introduction to the basic knowledge of the workflow technology including a background discussion on workflows, WFMSs and collaboration concepts. It also gives a brief overview of this thesis and the review on the related work.

1.1 Workflow and Workflow Management System

Today, organizations use WFMSs to streamline, automate, and manage business processes that depend on information systems and human resources (e.g., provisioning telephone services, processing insurance claims, and handling bank loan applications). A workflow is an abstraction of a business process. It consists of activities. A popular classification [GHS95] distinguishes three types of workflows:

- *Ad-hoc workflows*, workflows that are controlled by users at runtime. Users can react to situations not considered at design-time.
- *Administrative workflows*, predictable and repeatable complex workflows described in simple description language. Activities are mainly performed by humans.
- *Production workflows*, predictable and repeatable complex workflows which are predefined completely and in great detail using complex information structures and involve application programs and automatic activities.

Workflow management aims at modeling and controlling the execution of processes. A WFMS is a set of tools providing support for the necessary services of workflow creation, workflow enactment, and administration and monitoring of workflow processes [WfMC94]. There are many WFMSs in the market, which can be categorized as following [VW97]:

- Environments that are tailored towards a specific application domain: OASIS [M+96], ZOO [ILGP 96].
- WFMSs primarily for traditional (business-type) applications which could be used in scientific environments, although the latter were not initially foreseen as targets: METEOR [SKM+96], Mentor [Wod96], Mobile [BJS96], MQSeries [IBM98], and ePERT [PEL97].
- "Workflow-aware" systems for scientific applications: LabBase [SRG94], CRISTAL [MLK96].

We will discuss the concepts of workflow and workflow management system in more detail in Chapter 2.

1.2 Collaboration

Generally speaking, collaboration is a process by which people work together on an intellectual, academic, or practical endeavor. The collaboration can take many forms, either traditionally, such as in person, by letter or on the phone, or electronically, such as through email, video-conferencing or other advanced web-based collaborative applications [L99].

In this thesis, we will discuss computer aided Electronic Collaboration. Technology support for informal processes can be associated with a broad range of computer based technologies. Electronic mail has been the technology with broadest dissemination. Similarly, teleconferencing and video-conferencing have progressively been introduced to overcome the physical limitations of inter-personal and inter-group communication. These technologies are limited to the physical dimension of the communication, either time such as email, or space such as the telephone or conferencing facilities. In addition, the most relevant nature of emerging collaboration technology is the support for particular styles of group interaction processes. Examples of this style are consulting processes in collaborative environment, or decision processes that can be found as the object of support in multiple GDSSs (Group Decision Support Systems) [N+91].

Several classification types exist for collaborative technology. The basic classification divides the possible system into four classes according to the time-space and same-different combinations. Other aspects that have impact on the classification of collaborative systems include size of the groups, type and structure of the groups, or process support. We will provide more detail about collaboration in Chapter 2.

1.3 Description and Contribution of the Thesis

Researchers in collaborative applications and workflow systems have made progresses in developing these two systems separately. However, there are very few WFMSs having integrated collaborative applications. The progresses in collaborative applications, such as video-conferencing and web-based collaboration can provide huge potential opportunities for merging these distinct capabilities in WFMSs. Within such a system, a process can be coordinated and collaborated in organizational activities. This system can be sensitive to a process execution trigged by collaborative decisions, context-sensitive information updates, and other external events.

The goal of this thesis is to expand collaboration features in WFMS according to the workflow reference model defined by WfMC defines. We will provide an infrastructure for collaboration both within any high-level processes and for all the interactions and dependencies among them. The workflow system can support multiple execution paths as an integral part of collaborative problem solving. Collaborative applications can be invoked in a number of ways: automatically as an embedded part of a particular activity; as a tool that is potentially beneficial for a particular task at some specified points during the execution of a workflow process; or user-selectably on an ad-hoc basis.

With the concept of collaboration in workflow context, we will extend *METEOR* WFMS model by designing a collaboration model as the infrastructure for supporting collaboration in the workflow system. In particular, we will discuss collaboration task model, collaboration object structure and a general interface for workflow enactment systems and collaborative applications. Based on the collaboration model in METEOR WFMS, we will implement a prototype in OrbWork enactment system, a CORBA-based workflow enactment service developed in Java language. A collaborative workflow process is modeled as activities and tasks for people and system components participating in accomplishing the work with the collaboration. We use a collaborative application -- CaTCH (Collaboration and Tele-Communication for Health) in the prototype implementation.

In general we have made contributions to the workflow technology in collaboration area, especially for our METEOR WFMS we

- Introduce the collaboration concept, which allows WFMSs to have support for performing collaboration.
- Classify the collaboration types in workflow context.
- Provide an infrastructure for collaboration in workflow enactment service.
- Design a collaboration model for the workflow enactment system of METEOR WFMS.
- Implement a prototype of the collaboration model in OrbWork enactment system of METEOR WFMS.

1.4 Related Work

Recently good progress has been made in collaboration applications area and workflow area. With rapid Internet development, the collaboration, especially electronic collaboration (E-Collaboration) attracts more attentions in many industries. Participants in the collaboration prefer to interact online at anytime from anywhere. There are a number of projects involved in the E-Collaboration. For example, Northeast and Islands Regional Educational Lab at Brown University has several projects related to online collaboration for education. These projects allow the teachers to share the educational resources and ideas with the collaboration participants all over the world [L99]. Furthermore, many companies intend to do the collaborative business with collaboration partners online. It requires these companies to have the mission-critical collaboration solutions for their business models.

Most collaborative business processes involve multiple organizations to interact with each other. These processes require the organizations to do collaboration with each other because different organizations may have different business modes. Such interorganizational processes can be performed in interorganizational workflow systems. Many research efforts have been seen on the interorganizational workflow system area. The interorganizational workflow exceeds the individual organizational boundary and treats every organization as a sub-domain in the whole workflow domain. The high-level workflow process can be thought of as a virtual business process and the whole workflow domain as a virtual enterprise [SVA99]. The workflow process goes beyond a single enterprise boundary and is constructed by combining the services provided by different companies collaboratively. Apparently, most interorganizational processes need to do collaboration to communicate with partners, especially to exchange process information among them. In addition, interorganizational workflow technology can be used for E-business. Companies use the technology to enter new markets and meet the challenges of global markets. The effort of entering new markets can be seen in marketSite from CommerceOne [C99], MOPPET [ADT99], iMarkets [O98] etc. Other companies have integrated workflow systems as a component in their products, such as most ERP systems (e.g. SAP R/3, BaanERP, and PeopleSoft). These products support collaboration between the workflow systems and other components. A new bred of products will appear to dynamically create and support virtual communities of commerce partners.

The interorganizational workflow technology requires interoperability between different WFMSs. Interoperability may require WFMSs to interact with each other to exchange workflow information. Standards are necessary to achieve success in the workflow interoperability area. In general, we can classify interoperability for workflow systems into two categories. One is specifications for modeling and workflow description, and specifications for run-time interoperability. WfMC's Process Definition Interchange (Interface 1) standard falls into the first category. The other is OMG's jointFlow standard [OMG98] and WfMC’s Interface 4 [WfMC94], which aim to support exchanging process enactment information or interoperability at run-time. Similarly, Collaborative workflows require WFMSs to interact with collaborative tools. It’s necessary to have interface standards to integrate collaborative tools in WFMSs seamlessly.

1.5 Organization of the Thesis

Chapter 2 describes the concept of workflow technology, workflow reference model and concept of collaboration. Chapter 3 provides a review of METEOR WFMS’s model and architecture that are designed by LSDIS Lab, the University of Georgia. Chapter 4 presents the detail of the collaboration model in METEOR WFMS, including collaboration concept in workflow context and collaboration task model, collaboration object structure and general interface. Chapter 5 has a brief introduction of OrbWork enactment system for METEOR WFMS and CaTCH (Collaboration and Tele-Consulting in Healthcare) application used as the collaborative application in this thesis. Chapter 6 contains the implementation detail of collaboration model in OrbWork enactment system for METEOR WFMS. Finally Chapter 7 concludes this thesis and suggests possible future work in this project.

CHAPTER 2

THE WORKFLOW MANAGEMENT SYSTEM MODEL AND COLLABORATION

In this thesis, we need to present two concepts first. One is workflow and workflow management system. It includes definitions of the workflow technology and workflow reference model. The other is collaboration, collaboration types, and collaboration in the workflow context. With these two concepts, we will introduce run-time subject-oriented collaboration to WFMS's run-time service.

2.1 Workflow and Workflow Management System

Workflow management is a diverse and rich technology and is now being applied over an ever-increasing number of industries. Workflow is also a general term which may refer to different things at different levels such as process specification and enactment at system level, or process modeling at business process level [ED97].

In recent workflow research, there are still many definitions of workflow. In this thesis, we use a simple and effective one that is given on the workflow management review paper of [GHS95], the tutorial materials of [She95] and the interview of [J98]: *workflow* (or *workflow process*) is a computer-assisted (or automated) activity involving the coordinated execution of multiple tasks performed by different processing entities. A *task* defines a logical unit of work in a workflow that related to a specific commitment, adding value to a product or service of an organization. Workflow tasks are heterogeneous in nature. These tasks could be manual, or automated, either created specifically for the purpose of the workflow application being developed, or possibly already existing as legacy programs. A workflow also defines task dependencies that specify how tasks in a workflow are coordinated for execution in a semantically correct order [KS95]. *Workflow management* is the automated coordination, integrated control and communication of activities as is required to satisfy workflow process [SGJ+96]. A WFMS is a set of tools providing support for the necessary management service of workflow creation (which includes process definition), workflow enactment, and administration and monitoring of workflow process [WfMC94]. WFMSs manage the flow of activities among participants according to inter-task dependencies, and coordinate user and system participants, together with the appropriate data resources that may be accessible directly by the system to achieve defined objectives by possibly imposing deadlines. The coordination also involves passing task data from participant to participant in correct sequence, ensuring that the participants fulfill the required contributions, and taking default actions when necessary. Although there are many commercial products in the market claimed WFMS or workflow-enabled, we think that a WFMS must have the following essential ingredients to classify it as a workflow automation solution (e.g., *METEOR* WFMS):

- The ability to specify and model workflow;
- The ability to coordinate and manage the execution of workflow (enactment service);
- The ability to handle workflow exceptions;
- The ability to monitor the status of workflow process and measure the workflow process;
- The ability to analyze, simulate and test the behavior of the workflow process.

The developer of a workflow application relies on tools for the specification of the workflow process and the data that the workflow process manipulates. The specification tool cooperates closely with the workflow repository service, which stores workflow definitions. A workflow process is based on a formalized workflow model that is used to capture data and control-flow between workflow tasks. A workflow enactment service (including a workflow manager and a workflow runtime system) consists of execution-time components that provide a execution environment for the workflow process. A workflow runtime system is responsible for enforcing inter-task dependencies, task scheduling, workflow data management, and for ensuring a reliable execution environment. *OrbWork* is such an enactment system. Administrative and monitoring tools are used for management of user and work group roles, defining policies (e.g., security, authentication), audit management, process monitoring, tracking, simulation, and reporting of data generated during workflow enactment.

2.2 Workflow Reference Model

There are now numerous workflow products available on the markets with certain workflow capabilities. Workflow technology has been used in a wide range of application areas such as banking, finance, insurance, healthcare, telecommunication, manufacturing, and document management. Although the workflow technology in these areas are successful, there is a big obstacle of wide application of the workflow technology – standards. Almost every workflow vendor has its own workflow model, specification language, and APIs. Every workflow product vendor has its own standard means there is no standard in workflow area. Hence, standards are the most important factor in making workflow pervasive. In recent years, many efforts by Workflow Management Coalition (WfMC) have made significant progress in setting up workflow standards. In addition, other standards, such as CORBA (Common Object Request Broker Architecture) from OMG, COM from Microsoft also made a great progress on workflow technology.

In this thesis, we plan to integrate collaborative applications in WFMSs. We treat collaborative applications as workflow client applications in WFMSs. Hence, we choose WfMC Workflow Reference Model as the basic model. In this section, we will provide the brief review of WfMC workflow reference model.

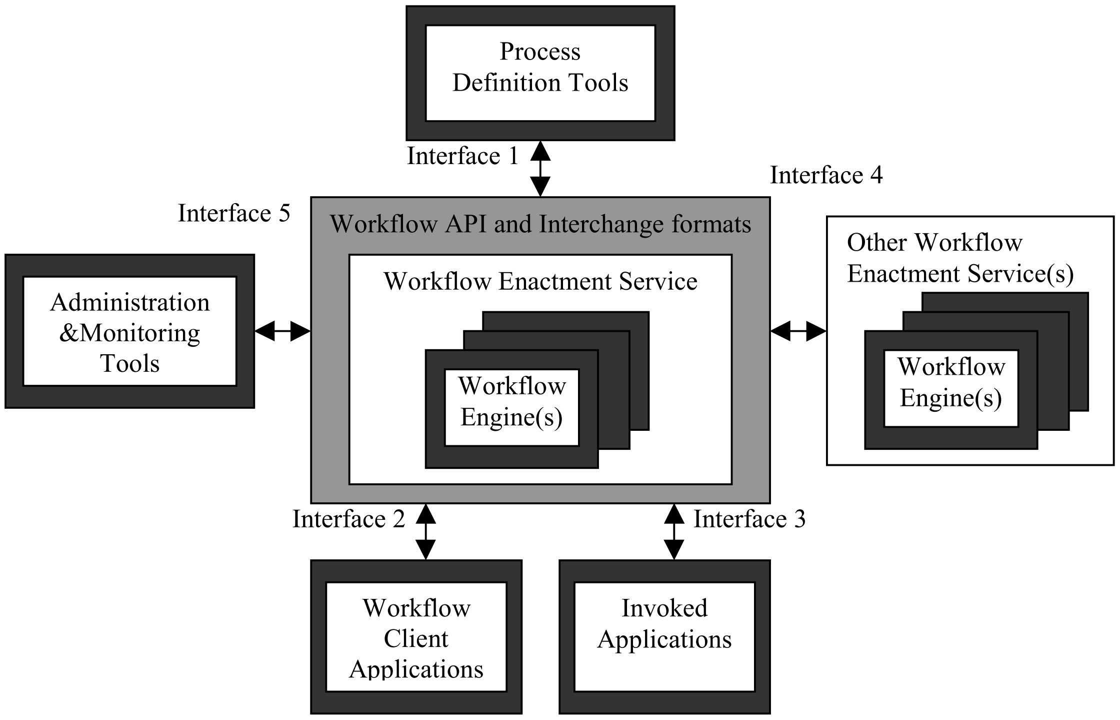

Process Definition Tools
Interface 1
Interface 4
Interface 5
Workflow API and Interchange formats
Workflow Enactment Service
Workflow Engine(s)
Administration &Monitoring Tools
Other Workflow Enactment Service(s)
Workflow Engine(s)
Interface 2
Interface 3
Workflow Client Applications
Invoked Applications

Figure 2.1 WfMC Workflow Reference Model

WfMC defines WFMS as a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines which are able to interpret the workflow process definition, interact with workflow participants and invoke the use of other tools and applications. The standardization work of WfMC is centered on the workflow reference model (Figure 2.1). The reference model specifies a framework for workflow systems, identifying their characteristics, functions and interfaces. The focus has been on specifying the five APIs that surround the workflow engine. These APIs provide standard means of communication between the workflow engines and the workflow clients (including other workflow components such as process definition and monitoring tools). Since this thesis will focus on workflow enactment service, workflow client applications and the interface between them, we will present these two parts in the model. Please refer to [WfMC94] for information of other parts.

The *workflow enactment service* provides a run-time environment in which process instantiation and activation occurs, utilizing one or more workflow management engines, responsible for interpreting and activating part, or all, of the process definition and interacting with the external resources necessary to process the various activities. A *workflow engine* provides the run time execution environment for a workflow instance in workflow enactment service. Several functions can be handled by the workflow engine, including interpretation of the process definition, control of process instances, navigation between process activities, sign-on and sign-off of specific participants, identification of workitems for user attention and an interface to support user interactions, maintenance of workflow control data and workflow relevant data, passing workflow relevant data to/from applications or users, an interface to invoke external applications and link any workflow relevant data, supervisory actions for control, administration and audit purposes.

Interaction occurs between the client application and the workflow engine through a well-defined interface embracing the concept of a *worklist* – the queue of work items assigned to a particular user (or, possibly, group of common users) by the workflow engine. At the simplest level, the worklist is accessible to the workflow engine for the purposes of assigning work items and to the worklist handler for the purpose of retrieving work items for presentation to the user for processing. Activation of individual work items from the worklist may be under the control of the workflow client application or the end-user. A range of procedures is defined between the workflow client application and the workflow enactment service to enable new items to be added to the worklist, completed activities to be removed from the worklist, and activities to be temporarily suspended. Application invocation may also be handled from the worklist handler, either directly or under the control of the end-user. *Client application interface* (*Interface 2*) provides APIs to manage the workflow worklist. It includes worklist manipulation, process status management, process and activity control, and worklist connection and disconnection.

2.3 Collaboration and Collaboration in WFMS Context

2.3.1 Collaboration

Collaboration is a process by which people work together on an intellectual, academic, or practical endeavor. In the past, that has meant in person, by letter, or on the telephone [KLP99]. In recent years, there are many ways in which groups and individuals collaborate online. We can call them Electronic Collaboration or E-Collaboration. Also, there are many ways to view collaboration. Considering this on a continuance based on the degree of interaction required, the spectrum runs from contributing to resources sharing to collaboratively designing and implementing a product such as a new tool or closely working together in mentoring relationship. This continuance reflects the degree to which an individual need communicate with others in the group (Figure 2.2) [L99]. We can categorize the collaboration in four types: *Resource Sharing, Open-ended Discussion, Focused Discussion* and *Mentoring.*

- *Resource Sharing*: Colleagues support each other's work by contributing to a shared collection of resources. The resources typically are documents, plans or standards. Usually at this level little interaction is supported. Participants are both invited to submit to the collection as well as extract from it.

- *Open-ended Discussion*: The range here is from resource sharing to more content directed conversation. The interaction is typically short lived and focuses on pragmatic matters. Colleagues engaged in the discussion with others in an area in which participants come and go and no attempt is made to focus the group.
- *Focused Discussion* (Subject-oriented discussion): Colleagues here typically participate in discussion with smaller groups of people. The content is focused on a particular subject. This might take place as a part of an organizational workflow process.
- *Mentoring*: Mentoring is a special relationship involving a high degree of interaction. In this case one or more individuals are engaged in a relationship of support involving a high degree of communication.

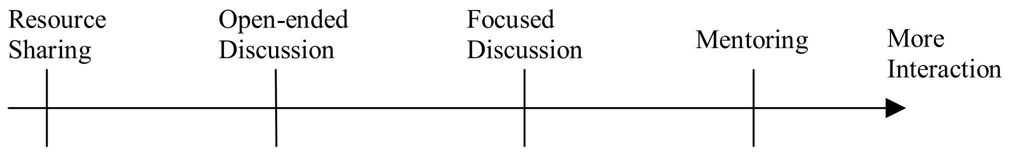


Figure 2.2 Collaboration Categories

Besides the classification described above, there is another basic classification that divides the possible systems into four classes according to the time-space and same-different combinations [GAP97] (Table 2.1).

Table 2.1 Basic Collaboration Classification

	Same Time	Different Time
Same Space	In person meeting	Sharing documents
Different Space	Video-Conference	Email, letter

With rapid growth in Internet development, the Electronic Collaboration area is receiving increasing attention. E-Collaboration is a kind of collaboration that connects individuals electronically via the Internet using tools or through access to sites on the World Wide Web [KLP99]. The Internet-based applications allow collaboration participants to communicate anytime, from anywhere to any place. People from different parts of a building, state, country, or continent can exchange information, collaborate on shared documents and ideas or work together.

E-Collaboration can have many different forms. Some of the more common activities include the following [L99]:

- *Discussion groups* are focused around a subject or a specific activity, goal or project. Some groups are open-ended, allowing users to solicit information from each other. Other groups, more structured, may use a moderator to guide the discussion by making comments, suggestions and connections.
- *Data Collection and Organization* activities use databases and search engines to organize and retrieve data. Users contribute data individually to a shared database and retrieve data from it when needed. Data can be in the form of reference links, research papers, and other information formats.
- *Sharing documents* is involved in the collaboration, from simply displaying the documents to having several people work on them simultaneously. Collaborators can display documents online and discuss the content via e-mail, video-conference or other collaborative tools. They can also use annotation systems to comment on shared documents and editing tools to co-edit documents on line.
- *Synchronous communication* activities, such as Internet video-conference and Internet chat, differ from the other types of activities in that they happen in real time, over a short period. Video-conferencing is like a conference call with picture online. The technologies allow users to discuss idea, debate problem,

consult an issue, and share information when face-to-face interaction is desired but not possible.

- *Online workshop* allows the collaboration participants to learn something new without traditionally face-to-face meetings. It allows people to participate whenever and from wherever they want. The material for the workshop, as well as the workshop itself can take place via a discussion group.

As almost all the types of E-Collaboration described above occur with a specific subject, topic or goal, we can think that the E-Collaboration is subject-oriented, which is much different from the process-oriented nature of workflows. *Subject-oriented collaboration* provides a collaborative communication method or a collaborative space that is based on one specific subject. Because it can be done at any time, form anywhere, E-Collaboration has the ability to provide a pool of resources and professional companions from all over the world. It also provides more opportunities for interaction than doing face-to-face collaborative activities. Some kinds of collaboration have asynchronous nature that allows participants to contribute to the collaboration when it's convenient and to reflect on what other participants have said before responding. Beyond providing the chance to connect to other people and time to reflect on subjects of importance, E-Collaboration can have other benefits. For example, the participants can use more advanced technologies in the collaboration to present their ideas or information clearly. Hence, E-Collaboration can be adopted in many areas. For instance, many educators would like to figure out how to teach and reach students more effectively. They will exchange their curriculum and their teaching experience among themselves and give more prompt responses to their students' questions and requests through E-Collaboration methods. The E-Collaboration method used in education may save a lot of time for the teachers and students, and provide more sharing information to them.

Over the last two years, a lot of collaborative applications have emerged. There are two categories of collaborative applications in the marketplace*: real-time* collaborative applications and *web-based* collaborative applications.

- *Real-time* collaborative tools are a blend of electronic whiteboard, Internet chat and video-conference software. The collaboration process may involve synchronous communication through video-conference, application sharing and data sharing or instant messaging services (e.g., Microsoft's NetMeeting or Yahoo's Instant Messenger).
- *Web-based* collaborative tools are from outgrowth of traditional GroupWare. They are designed to provide a collaborative electronic space that can be configured for using by the participants with/without a great deal of administrative support. They may provide asynchronous consultation service whereby the data is provided to the consultant. (e.g. CaTCH developed in the LSDIS Lab, University of Georgia, Lotus's QuickPlace)

2.3.2 Collaboration in the Workflow Context

To carry out an E-Collaboration, we need to have a collaborative environment. The environment may be based on the individual tool, such as email, or on multiple tools. For example, in a healthcare organization, physicians may use a workflow system to coordinate their organizational process and use a collaborative tool to consult some cases with other physicians out of the organization.

As described in section 2.1, *workflow management system* is a set of tools providing support for the necessary services of workflow creation (which includes process definition), workflow enactment, and administration and monitoring of workflow processes [WfMC94]. Now we can explain two related concepts: *coordination* and *collaboration*. Coordination is understood as a process by which the individual activities of the group members become organized (in terms of inputs, outputs and scheduling). By an external entity, this organization leads to the predefined goal in such a way. Collaboration emphasizes the capability of self-organization of those group members, which makes progress to the final goal through informal and mutual adjustment [GAP97]. Based on these definitions, a *workflow management system* can be classified as a coordination tool, while interaction tools for group discussion, document sharing, data collection and synchronous or asynchronous communication can be classified as collaboration tools.

As a commercial and coordination technology, WFMSs have undoubtedly achieved significant success. It provides comprehensive support for organizational activities. It can be thought as a kind of process-oriented coordination system in the organizational domain. However, many WFMSs do not support collaboration technology or subject-oriented discussion in the systems. But we find in many real organizations, subject-oriented collaboration is useful. For example, a physician in a remote rural clinic may have limited medical resources, while the other physician(s) in a health center may have advanced and specialized high-tech healthcare facilities. In the rural clinic, workflow system is used for its organizational process. If the physician in the remote clinic needs to consult a particular patient's case with the one having the advanced healthcare facilities and having better experience, he could initiate consultation collaboration with the experienced physician who can be in the same clinic or out of the clinic. Hence, this collaboration may occur out of a workflow process and it's apparently subject-oriented. Unfortunately, within many current WFMSs, we can't do subject-oriented collaboration in the WFMS. They don't support the communication with others out of the workflow domain. If one of the participants of collaboration does not have access to the workflow system, the collaboration can not be executed in the workflow system. That means the workflow users can not initiate collaboration from the workflow system directly. To solve the problem, we have to extend the WFMS's functionality to support the collaboration feature.

In the WFMS context, collaboration technology means participants of collaboration can communicate with each other, share information, exchange ideas or make a decision through collaborative tools between the process or beyond the WFMS scope.

A WFMS has two major parts: *design-time* and *run-time* system. As a result, there can be two kinds of collaboration in WFMS context: *design-time collaboration* and *run-time collaboration.*

- *Design-time collaboration* takes place when several workflow designers work together to design a workflow process. These designers may collaborate their design work together. This kind of collaboration is mainly *documents sharing* collaboration and *data collection* collaboration. For example, the workflow design is stored in a central site and all designers can access the workflow design concurrently. They need to collaborate with each other to guarantee the consistency and operability of the design. This project won't cover the *design-time collaboration* and we will not discuss this kind collaboration any more.
- *Run-time collaboration* takes place when a workflow process is executing. The collaboration is related with the workflow process, and may occur in the workflow process optionally or arbitrarily. The collaboration in healthcare scenario described above is such run-time collaboration. In addition, this kind of collaboration is subject-oriented apparently, which provides a collaborative communication method or a collaborative space that can be used for the users in and out of the workflow domain.

We define a collaborative workflow as a workflow in which the user can perform run-time collaboration with some collaborative applications. The difference between collaborative workflows and regular workflows is that the collaborative workflows have the collaboration functionality in coordination-nature workflow processes. Users can perform collaboration and coordination combined in collaborative workflows by using collaborative applications in the workflow run-time system.

For the run-time collaboration in workflow context, we define a formal syntax specification as follows. Collaboration C in a workflow is a tuple C = (N, A, O, I), where N is a collaboration task, A is a collaborative application, O is a set of collaboration objects, and I is an interface between collaborative applications and WFMSs. A collaboration task N is N = (M), where M is a set of task attributes including general and specific attributes such as name, host, input/output data, data mapping and task realization etc. An collaboration object CO in O is a tuple CO = (D, U, R), where D is a set of data fields in the object, U is the URL or reference of the collaboration object which allows workflow users to access the object, and R is the receiver(s) of the object. The interface I in C is a tuple I = (L, E, F), where L is a collaboration notifying mechanism, E is an extractor of collaboration object information and F is a collaboration tool invocation the for workflow collaboration task.

The syntax model of collaboration in WFMSs describes the components of the collaboration. We will discuss more detail about how to design the run-time collaboration in the METEOR WFMS in Chapter 4.

2.4 Necessity for Integrating WFMS and Collaboration

We think most current WFMSs support coordination only. They can provide good management services to business processes, such as coordinating the execution of tasks, administrating and monitoring workflow process. Generally, WFMSs perform well in process-oriented activities.

However, a significant amount of activities are related to the collaboration. In these situations, WFMSs are unable to provide collaboration services necessarily. For example, in a remote rural clinic with limited advanced medical facilities and medical expertise, a physician want to schedule a consultation on a particular patient's condition with other physician(s) having the advanced healthcare facilities and experience. If the WFMS in this rural clinic doesn't support collaboration online, the physician has to talk to other physician(s) on the phone or by other traditional means. But sometimes the complex conditions of the patient are hard to be described in words, such as X-ray pictures or video clips. It is better to describe it if the collaboration partner can see the X-ray pictures or other media stuff at the same time. Here is the issue. Is it practical that the physician takes all the record to go to other place far away from the rural clinic? Apparently, it is impossible. Hence, if the workflow system can support the collaboration, we can solve the problem easily. We will describe the healthcare application scenario in detail in Chapter 4.

As analyzed above, we think integrating collaboration feature with WFMSs is very useful. The collaborative workflow can support a lot of activities having collaboration in nature.

In this thesis, we put our attention to the run-time collaboration, which are subject-oriented. We will design a Collaboration Model in *METEOR* WFMS and implement a prototype of the collaboration model in OrbWork enactment service for METEOR with *CaTCH* (Collaborative Tele-Consulting for Healthcare) system as the collaborative tool to present how the subject-oriented run-time collaboration is carried out in the WFMS. From the thesis, we contribute our effort in the following areas, such as adding a Collaboration task model in the *METEOR* model, introducing a method to implement the run-time subject-oriented collaboration in the run-time system of WFMS, integrating the collaborative tools in the WFMS and producing a general interface between the collaborative tools and WFMSs.

CHAPTER 3

THE METEOR WORKFLOW MANAGEMENT SYSTEM MODEL AND ARCHITECTURE

The objective of the original METEOR (Managing End-To-End Application) project [KS95] at Bellcore was to facilitate and support the automation of enterprise-wide operations (workflows) that execute on a heterogeneous, autonomous and distributed operating environment. To achieve this objective, the original METEOR model [KS95] offers a well-defined model and an intermediate language for specifying the workflows and the tasks, a compiler for the language, and a run-time enactment system that provides functionality to monitor the progress of the workflow execution, enforce the inter-task dependencies and manage the execution of each individual task.

The new METEOR [SKM+96, WS97] workflow model developed in LSDIS (Large Scale Distributed Information System) Lab, the University of Georgia extends the original METEOR model in terms of both the logical model and the run-time architecture to support large-scale multi-system workflow applications in heterogeneous and distributed operating environments. In this thesis, we use METEOR to represent the new METEOR workflow model developed in LSDIS LAB, the University of Georgia.

This chapter gives an overview on the basic concepts of the METEOR workflow model and provides a brief discussion of the overall architecture of the METEOR WFMS.

3.1 The METEOR Workflow Management System Model

A *workflow* in METEOR can be modeled as a collection of tasks, a set of inter-task dependencies, data sets and Exceptions. In this section, we will review the basic concepts of the METEOR workflow model, including the concepts of the METEOR task model, the different types of inter-task dependencies to construct a workflow, data model, exception model and the workflow intermediate language used to specify a workflow. For briefness, we will only discuss the task model and the inter-task dependencies in this chapter. Please see [K99] for detail of other parts of METEOR model.

3.1.1 Task Model in METEOR

A task is an operation or a sequence of operations that are submitted for execution in the context of a workflow. It represents an abstraction of activities, which can be performed by a variety of processing entities, depending on the nature of the task. A task can be performed (realized) by a computer program, a database transaction, or possibly by a network of interconnected tasks called a sub-workflow. An analogy exists between a programming language procedure and a workflow task. Like procedures, workflow tasks may have input/output parameters, used to transfer data in/out of the workflow task, or possibly the whole workflow. However, unlike in procedures, a given set of input parameters is not matched with a corresponding set of output parameters. The whole workflow can be regarded as a single task, and may have a collection of input and output parameter [K99].

In the METEOR model, task structures are modeled by using directed graphs [KS95]. Figure 3.1 shows the general task model in METEOR. The nodes in the graph represent the externally visible states of a task, while the arcs correspond to a task's permissible internal state transitions. An internal transition can be either controllable or uncontrollable. An internal transition is said to be controllable if an enable arc outside of the task can trigger the transition; otherwise, the internal transition is said to be uncontrollable.

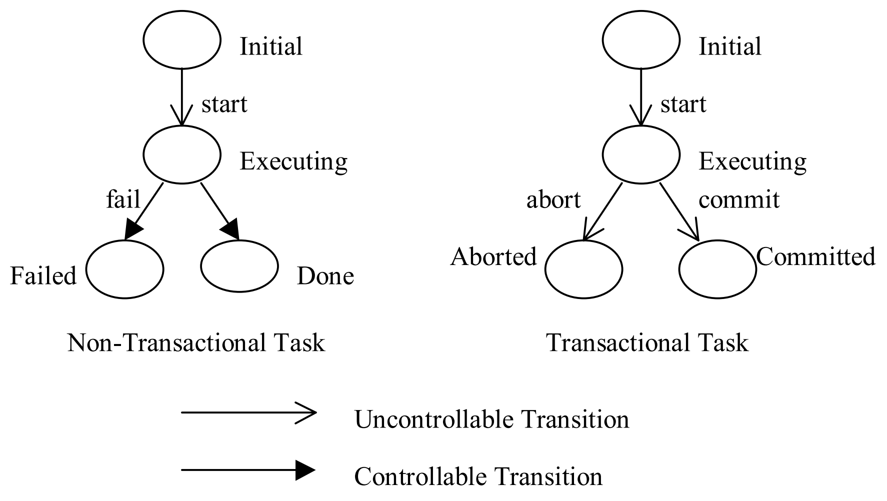


Figure 3.1 General Task Model in METEOR

One of the characteristics of the METEOR workflow model is that it provides well-defined task structures to model heterogeneous workflow tasks in the real world. Tasks in the general task model could be *simple tasks* or *compound tasks*. A simple task is an atomic activity that represents an indivisible logical unit in a workflow, while a collection of related simple tasks can be organized hierarchically to form a *compound task*. Both simple tasks and compound tasks can be transactional and non-transactional in nature [KS95].

A task may be invoked, analogously to a procedure call. A task invocation creates a *task instance*, which, in case of a task being realized by a sub-workflow, creates a sub-workflow instance. A *task invocation* specifies a number of input parameters that must be supplied for a proper task realization. Some parameters of an invocation may have defined default values. Such parameters are called *optional parameters*. Non-optional parameters are called *required parameters*. Each task must have at least one invocation.

Tasks with multiple invocations are permitted. Note that a task with multiple invocations is analogous to an overloaded procedure. A task invocation may have no parameters at all. In this case, it is analogous to a procedure with no parameters.

A task instance terminates when its realization terminates. A task realization either succeeds or fails, which is then reflected by the task entering its success final state (either Done or Committed) or its failure final state (either Fail or Abort). A task may enter its failure state due to a failure of its realization, which is described by a suitable exception object. In this case, the task is said to throw this exception.

3.1.2 Designing Networks (Maps)

A *network* represents the core of the workflow activity specification. It is a collection of tasks and transitions, which represents a sub-workflow (or possibly the whole workflow, i.e. the network of the topmost task). Since a network is one type of the realization, it is always associated with some task, called its *parent task*. A single network defines a relationship among workflow tasks, transferred data, exception handling, and roles.

A network has one or more *start tasks* and one or more *final tasks*. Each final task must specify if the network succeeds or fails once the task terminates. If a final task specifies that the whole network fails, it must indicate which exception should be thrown as the result of the failure.

3.1.2.1 Transition

A *transition* joins two tasks and represents a transfer of activity in the workflow from the source task to the destination task. Precisely, the transition joins one of the final states in the source task with the initial state of the destination task. An *external transition* is a special type of a transition, in which the two participating tasks (source and destination) are not in the same workflow. An implied external transition leads to a start task of the network. Similarly, an implied transition leads from the final task, and is used to notify the external entity that the network has terminated. External transitions are also used to specify synchronization points with some external events. Typically, external transitions may be used to specify communication and synchronization between two independent workflows.

A transition may have an associated set of data classes that are shipped along with the transition when it is activated. Additionally, each transition has an associated Boolean condition. The transition may be activated only if the condition is true in the run-time. The classes that are associated with an input transition to a task are called the task's *input classes*, and those appearing on an output transition are called *output classes* of that task. A task's output class, which is not its input class, is created by the task. An object instance of the specified class is created by the workflow run-time enactment system. A task's input class, which is not its output class, is dropped. For each task, a mapping from the input classes of a task to the required parameters of one of its invocations must be defined. Similarly, a mapping from output to the output classes must be defined too.

A group of input transitions is called an AND-join if all of the participating transitions must be activated for the task to be enabled for realization. An AND-join is called enabled if all of its transitions have been activated.

A group of input transitions is called an OR-join if one of the participating transitions is activated for the task to be enabled for realization. An OR-join is called enabled if one of its transitions has been activated.

A group of transitions are said to have a *common source* if they have the same source task and all lead either from:

- its success state, or
- its fail state with the same thrown exception.

A group of *common source transitions* may form AND-split, OR-split, loop or fork. A group of common source transition is called an *AND-split* if each of the transitions in the group has the condition set to true. It means that all of the transitions in the group are activated when the task completes. Note that an AND-split creates parallel paths (executed in parallel by a workflow instance).

A group of common source transitions called an OR-split is an ordered list of transitions where all but the last transition may have arbitrary conditions. The last transition on the list has the condition set to true. An OR-split may be associated with a task's failure state. In this case, the condition of the last transition does not have to be set to true and if none of the conditions is satisfied, the whole network terminates with failure, and its parent task enter its failure state with the same exception.

A loop is a special case of an OR-split, where the list is composed of exactly two transitions. A loop requires that all of the paths beginning with one of the transitions and none of the paths beginning with the other transition cycle back to the source task. Note that if a loop has both transitions set to true, the second transition is useless. Such a loop is infinite.

A fork is an ordered list of common source transitions in which the first transition has the condition set to true, and the remaining transitions may have arbitrary conditions. The semantics of a fork is as follows. Once the source task completes, every transition in fork, for which the condition is evaluated to true, is activated. The first activated transition is the continuation of the original workflow instance, from now on called the *parent instance*. Any other activated transition forks off (spawns) a new workflow instance, called the child instance. Each child instance is considered identical to its parent up the point of the fork. A transition originating a child instance receives copies of the data objects. Original objects are only available to the parent instance.

3.2 The METEOR Workflow Management System Architecture

METEOR's architecture includes a collection of four services and tools, implemented as separate modules. Collectively, they are called EAppS (METEOR Enterprise Application Suite of tools and services). The four components are EApp➢Builder, EApp➢Repository, EApp➢Enactment, and EApp➢Manager. EApp➢Enactment includes services: OrbWork and WebWork. Both OrbWork and WebWork use a fully distributed open architecture. OrbWork is better suited for more demanding, mission-critical enterprise applications requiring high scalability, robustness and dynamic modifications. Figure 3.2 shows the overall architecture of METEOR WFMS [KSM99].

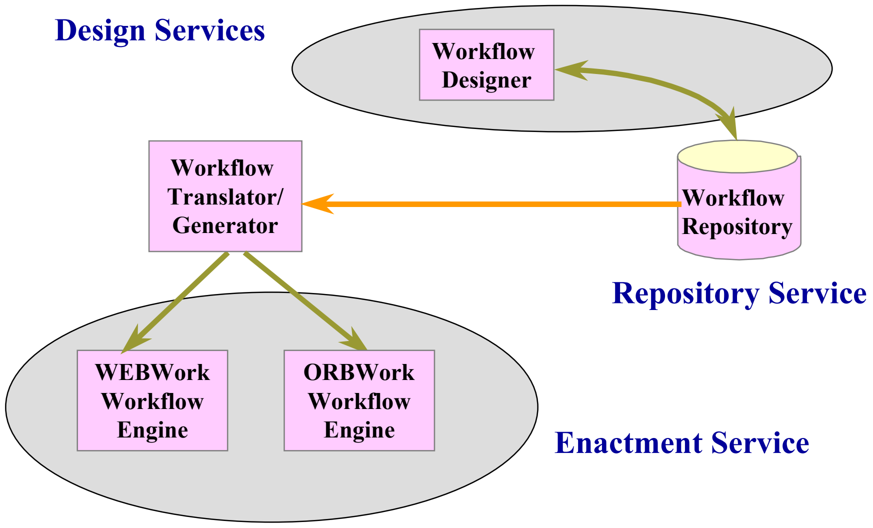


Figure 3.2 METEOR Architecture

3.2.1 Workflow Builder Service

The METEOR graphical designer is used to develop a workflow application, in some cases leaving no extra work after a designed workflow is converted to a workflow application by the runtime code generator. This service has three main components used to specify the entire map of the workflow, data objects manipulated by the workflow, as well as the details of task invocation, respectively. This service has the capability to model complex and varied tasks in a high-level concept and easy to use the manner that shields the designer of the workflow from the underlying details of the infrastructure or the runtime environment. It also provides the user very few restrictions regarding the specification of the workflow. The designer assumes no particular implementation of the workflow enactment service. Its independence from the runtime supports separating the workflow definition from the enactment service on which it will ultimately be installed and used. Workflow process definitions can be stored in the workflow repository managed by the METEOR repository service.

3.2.2 Workflow Repository Service

A workflow repository is a database of information about workflow processes, data, organizations and other perspectives of workflow design such as task realizations, communication protocols, external applications, and resource interfaces. Storing workflow design information in a repository brings the advantages of having repository functions such as versioning, lifecycle management, querying of objects, and sharing the design information between various tools. The METEOR Repository Service is responsible for maintaining information about workflow definitions and associated workflow applications. The workflow designer, which use the graphical workflow design tool, communicates with the repository service and retrieves, updates, and stores workflow definitions. Although workflow processes within different enterprises have common elements, they are typically designed from scratch. Using a common workflow repository or transforming design information between repositories, workflow design information can be shared within an enterprise or between different enterprises, thus providing a basis for common understanding and shared business intelligence.

The workflow repository will play a central role in the METEOR WFMS architecture. Workflow design information provided by the workflow designer developed by NRL and LSDIS Lab, the University of Georgia and possible other design tools (such as other workflow or enterprise design tools, or a specialized tool for enterprise organizational modeling) will be stored in the repository. These tools can share or reuse the design information by interacting with the repository. Currently, METEOR's two run-time systems, namely OrbWork and WebWork need workflow specification files in special formats to use in their internal processing. Therefore, workflow design information in the repository needs to be transformed into these formats. Other supportive tools, such as a monitoring tool, a browsing tool (to navigate and query the objects in the repository), and an administrative tool will also benefit from repository functions.

3.2.3 Workflow Enactment and Manager Service

The function of the enactment service is to provide execution environment for processing and monitoring workflow instances and administering available workflows. At present, METEOR provides two enactment services: OrbWork, discussed later in this paper, and WebWork. Each of the two enactment services has a suitable code generator that can be used to build workflow applications. In the case of OrbWork, the code generator outputs specification files for task schedulers, including task routing information, task invocation details, data object access information, user interface templates, and other necessary data. The code generator also outputs the code necessary to maintain and manipulate data objects, created by the data designer. The task invocation details are used to create the corresponding "wrapper" code for incorporating legacy applications with relative ease. We will discuss more detail about OrbWork enactment service in Chapter 5.

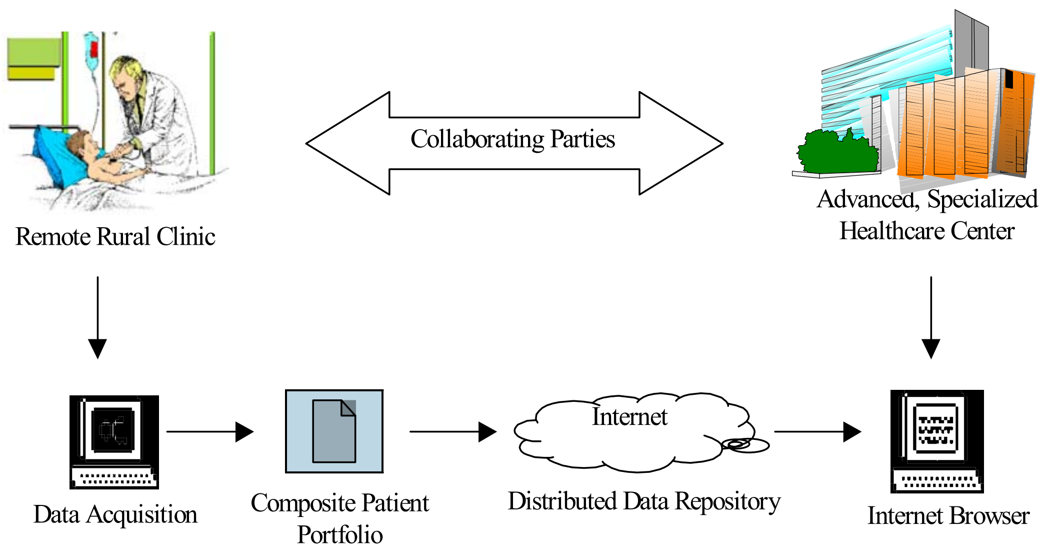
CHAPTER 4

COLLABORATION MODEL IN METEOR WORKFLOW MANAGEMENT SYSTEM

In this chapter, we will introduce our collaboration model to WFMSs, including the collaboration object structure and general interface between WFMSs and collaborative tools based on the healthcare scenario. Before we present our collaboration model to WFMSs, we give a healthcare application scenario in the workflow system with collaboration

4.1 An Application Scenario with Collaboration

In order to understand the requirement of the run-time subject-oriented collaboration in the METEOR WFMSs, let us think about an example scenario in the healthcare area. Suppose collaboration takes place between physicians in different locations at anytime. The physicians are collaboration partners. Figure 4.1 briefly shows the structure of the healthcare collaboration scenario. A physician in a remote rural clinic with limited medical resources is the user of a workflow process in the remote clinic. Sometimes, a patient's condition is more complicated and the physician is not able to handle it within the rural clinic with its limited resources or the physician does not have much experience with the patient's condition. Hence, he probably has to consult it with one or several physicians in other health centers with advanced high-tech healthcare facilities. In order to do consultation, patient information is bundled as a collaboration object and stored in a web accessible repository, and the collaboration partner accesses the patient information to provide his professional comments. In this case, the consultation is out of the workflow process itself because one of the participants may not be in the remote clinic. And the collaboration object can be passed through the tasks in the workflow process. Because the current METEOR WFMS does not support this kind of collaboration, we need to introduce a collaboration model to the workflow system to make the collaboration with other outside participants executable.


Remote Rural Clinic
Collaborating Parties
Advanced, Specialized Healthcare Center
Data Acquisition
Composite Patient Portfolio
Internet
Distributed Data Repository
Internet Browser
Figure 4.1 Healthcare Collaboration Scenario

A typical medical scenario would involve a physician to initiate collaboration and create a collaboration object for his patient. The collaboration object in this case is basically a “view” of the patient’s current condition and problems. It would typically contain information regarding the patient’s medical history, allergies, symptoms, possible cause etc. Each collaboration object would encode patient information, such as video clips, images, X-rays database views, lab reports and possibly audio commentary from the physician about the case. There will be no limit on the amount of information to be packed.

After the collaboration object is created, it is then "shipped" over the Internet to a physician who can view all the components of the object. The receiver physician then responds back to the consulting physician using a similar approach. Thus a collaboration object moves back and forth between the various collaboration partners till a acceptable result is obtained.

In this scenario, the collaboration optionally depends on the patient's condition because we can not foresee the collaboration task will occur definitely when we design the workflow process for the remote clinic. We will discuss the two modes of the collaboration task in the METEOR system. In addition, the collaboration happens out of the workflow process and involves participants in and out of the workflow process. Besides above requirements, the collaboration also needs interaction with the workflow process.

Figure 4.2 displays the basic architecture of a simple healthcare workflow with collaboration in the remote clinic using the task map of the workflow process. In a normal case, when a patient goes to the remote clinic, a nurse will do the registration for him and record the appearance symptom. This is done by the *PatientRegister* task (Human task). Then he has to do some medical tests and lab tests first, which is done by the *MedicalTest* task (Non-Transactional task). Meanwhile, the patient's medical history is retrieved by the *RetrieveData* task (Transactional task). When the test reports and the patient's medical history are available, the physician will analyze the patient's symptom. Here we have an optional collaboration for analyzing symptom task. In *CollabDecision_AnalyzeSymptom* task (Human task), the physician reviews the symptom first. If he feels the patient's condition is quite complex and he can't handle it perfectly, he will decide to initiate a subject-oriented collaboration with other physician who has more advanced healthcare facilities or more experience. Otherwise, he will go to *Analyze_MedicalReport* task (Human task) directly from the decision task, skipping the collaboration task. If he decides to do the collaboration, the *AnalyzeSymptom* task, which is a Collaboration task, is executed.

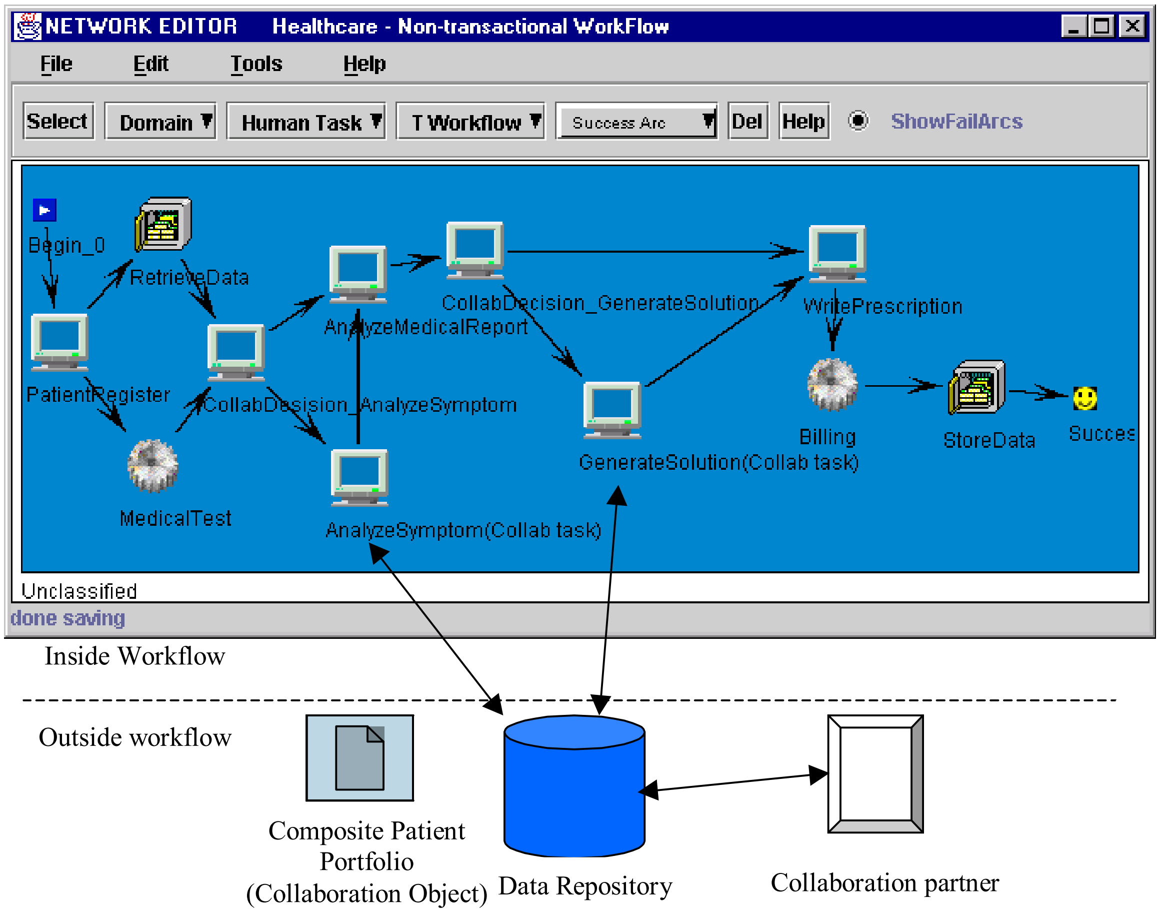


Figure 4.2 Simple Healthcare Workflow with Collaboration

The collaboration has been initiated and there is interaction between the participants of the collaboration. After the collaboration is done, the workflow instance will go to next task, *Analyze_MedicalReport* task (Human task). After analyzing the symptom and medical report, the physician will decide a solution to patient's condition. Here is another optional collaboration task. In the *CollabDecision_GenerateSolution* task, if the physician is not quite sure that the solution he has is correct, he will decide to initiate another subject-based collaboration with some physicians who have more experience or more advanced healthcare facilities, which is done in *GenerateSolution* task (Collaboration task). Otherwise, he will go to *WritePrescription* task (Human task) directly. Note that this collaboration in this task is not related to the collaboration in analyzing symptom task. After the collaboration is completed, the *WritePrescription* task is executed. Then the billing information is generated, which is done in *Billing* task (Non-Transactional task). Finally, all the data, including medical report, solution, prescription and billing information, is stored to the database in the remote clinic.

From the healthcare scenario, we know each workflow instance may not have collaboration, or may have collaboration once or twice. All of the collaboration is subject-oriented, that means their subjects in the collaboration are independent. The data objects in the collaboration can be shared with other tasks in the process. The participants of the collaboration may be from one or more organizations, which makes the collaboration cross the organizations' domain. Executing collaboration across organizations' domain is a little complicated because these organizations may have heterogeneous systems. We think that the collaboration occurring in one organization is easier than that occurring among several organizations. But in general, the intra-organization collaboration is a special case of the inter-organizational collaboration. We will implement the inter-organizational collaboration in this thesis.

4.2 Collaboration Task Model

In the METEOR WFMS, the *run-time subject-oriented collaboration* occurs when the user of the workflow instance decides to collaborate with other people about a specific subject in or out of the workflow system using a kind of collaborative tool independent of the WFMS. To represent the activity of the collaboration in the METEOR system, we need to add one more task type, named *Collaboration task*.

In the current MEREOR workflow system, it has four basic task types except *Collaboration task*: *Human task*, *Transactional task*, *Non-Transactional task* and *Network task*. A workflow process, which is designed via METEOR workflow designer tool, consists the combination of these task types. The design mode for the collaboration task is different from that for the other four basic tasks. At the design-time, we may not foresee the situation in the run-time totally. For collaboration, we are not sure it will take place definitely at run-time. What we know is the possibility of a task that may generate collaboration. In this case, the collaboration is optional. For example, in our healthcare scenario, the decision for the collaboration of analyzing symptom is made at the run-time based on the condition of the patient. With the uncertainty at the design-time, we can use optional collaboration task mode in the designer. This task provides the run-time user a method to perform the collaboration with other people when necessary. Due to the uncertainty of performing the collaboration task, there should be a task to make the decision on whether it will occur directly before the collaboration task. This decision task has a condition value to make an OR split for its outgoing arcs. One is to the collaboration task; the other is to the task(s) directly after the collaboration task. Meanwhile, the task(s) directly after the collaboration task should have an OR-join for its incoming arcs. Figure 4.3 shows how the optional collaboration task is normally designed in the METEOR's designer.

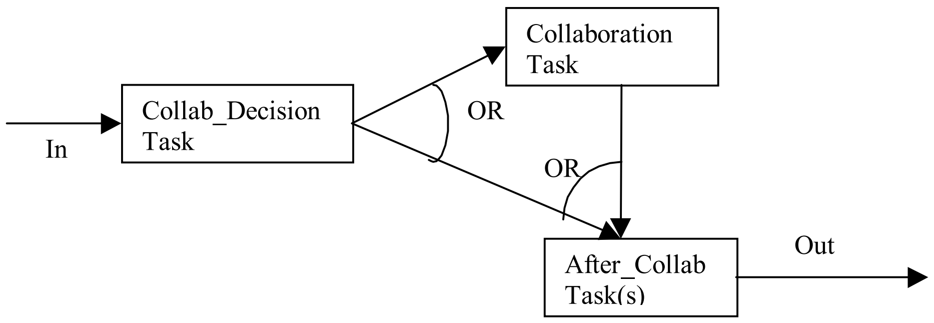


Figure4.3 Optional Collaboration task design mode

If we foresee that the collaboration task will occur at the run-time definitely, we can design the workflow process with an absolute collaboration task. We call this kind of collaboration is absolute. Figure 4.4 shows how the absolute collaboration task is designed in the METEOR's designer. This mode is the same as the mode in other basic tasks.

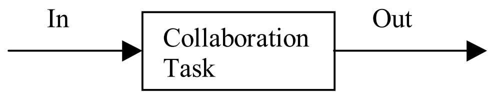


Figure 4.4 Absolute Collaboration task design mode

Having understood the requirements of the run-time subject-oriented collaboration in METEOR WFMS, we conclude that the collaboration task in workflow process has its special features as well as the general features of all the task types. Figure 4.5 shows the run-time status of the collaboration task.

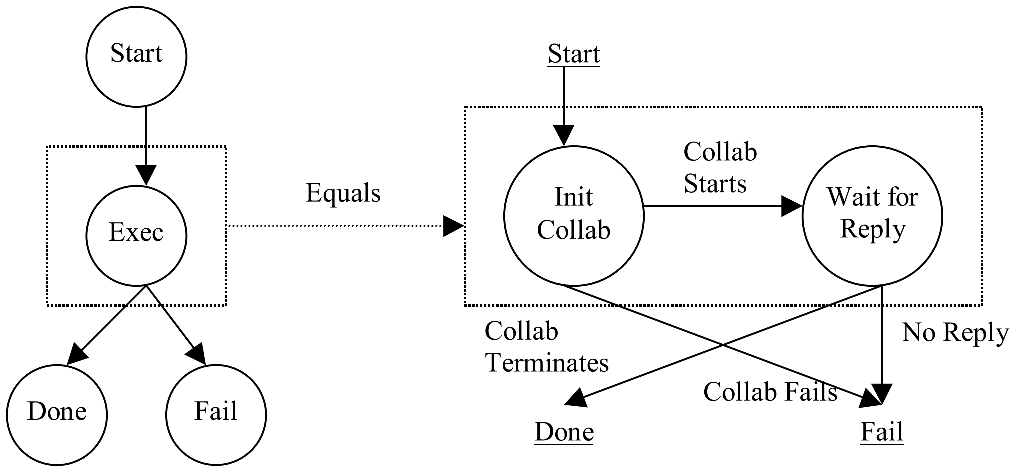


Figure 4.5 Status model of Collaboration task

The top level of status model in collaboration task is the same as the other task types in METEOR system. It has *Start*, *Execute*, *Done* and *Fail* states. But in *Execute* state, it has two stages: *Initiate Collaboration* state and *Wait for Reply* state. After the *Start* state, the task is in *Initiate Collaboration* state, which initiates a subject-oriented collaboration in the run-time. The initiation of collaboration includes invoking the collaborative tool and generating the collaboration object. If the collaboration is initiated successfully, it goes to *Wait for Reply* state to wait for the reply from the partner of the collaboration. If the collaboration is not initiated successfully, such as the collaborative tool can't be invoked, it leaves the *Execute* state and goes to *Fail* state directly. In the *Wait for Reply* state, there are two outgoing directions too. One is successful direction to *Done* state, which means that the collaboration is completed. The other direction is to *Fail* state, which means the collaboration can't be finished successfully, such as no reply in a reasonable duration.

In METEOR WFMS, each task type has its own special attributes besides the general attributes. The following attributes are for each task type:

- Name: Name of the task.
- Type: Type of the task. (Collaboration, Human, Transactional…)
- Security assignment: ACL (Access Control List) list for the task. It defines which object the task has rights to access.
- Host: The name of a host that the task resides on.
- A set of invocations, where each invocation has a name and a list of input parameters (Each parameter has name, type, and optionally default value). An invocation can be thought as an input parameter.
- Output parameters: Each parameter has name and type
- A list of thrown exceptions
- Absolute constrains: they include temporal, access, and condition on inputs.
- Short description (annotation): it briefly describes what the task should do.
- Task realization (implementation): it provides the task implementation.

A task may be realized by various means. For example, it may be accomplished by running a computer program, or by a human. A more complex task may have to be realized by a sub-workflow of other tasks. Specific task type realizations introduce additional attributes. Task realization with collaboration task form a hierarchy, as shown in Figure 4.6. A task realization may be either non-transactional, transactional, or workflow. For more detail, please refer to the METEOR system model in Chapter 3.

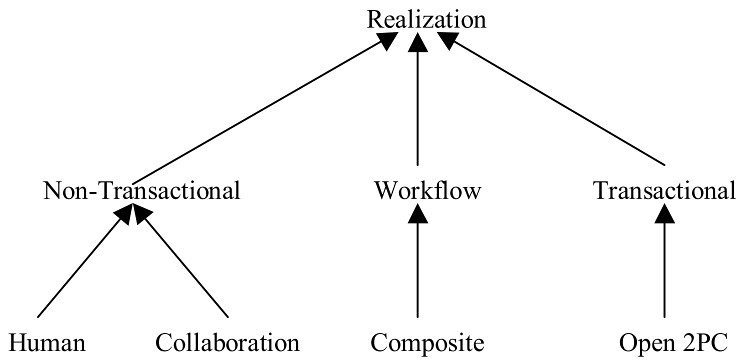


Figure 4.6 Task realization hierarchy

Besides the general attributes and realizations for all task types, the Collaboration task has its own special attributes. The realization is performed by people participating in the collaboration using the collaborative tool. The realization terminates after the user in the workflow instance gets the reply from the collaboration partner. Additional attributes for the task include:

- HTML forms: these HTML forms display the input parameters of the task and the links to the collaborative tool.
- Mapping of input/output parameters to form fields
- Collaborative tool name (type): this attribute tells METEOR system which collaborative tool will be used in this task.

- Collaborative tool invocation: this field gives METEOR run-time system a method to invoke the collaborative tool. It can be a link or a button to invoke the tool to initiate the collaboration. This attribute may have the location of the server of the collaborative tool and the file type the tool accepts.
- Collaboration object structure: this attribute gives METEOR system a data structure that describes the collaboration object. The detail information will be discussed in the next section.

After integrating the Collaboration task model in the METEOR WFMS model, we can have run-time subject-oriented collaboration when the workflow instance is executed. The collaboration task model should be implemented in both design-time system and run-time system, such as some special attributes should be given in the design-time, which requires that the design tool has the GUI to allow the designer to give these attributes' values, and the two stages of the execution of the task should implemented in the run-time system. We will discuss the detail about the implementation in the later chapter.

4.3 Collaboration Object Structure

In the Collaboration task, the collaborative tool may create the collaboration object for the collaboration. The object may be stored in a repository out of the organization of the workflow process. The workflow process needs to get the object itself or the reference of the object from the collaborative tool and sometimes needs to pass it to other tasks in the workflow process. In the current METEOR WFMS, there is a data designer to design the normal workflow data object that can be transported through the workflow process. We can treat the collaboration object as a separate part of the normal data object. Based on the normal data object structure and data designer, we can design the collaboration object similarly. The collaboration object structure (Figure 4.7) will have the following attributes:

- Collaboration object structure name: Name of collaboration object structure. In general, one type of collaboration object structure for one collaborative tool.
- Collaboration object data type: this data type is similar with the normal data type. It includes all the data fields in the data object. One data field has the field name, field type and field value.
- Collaboration object data URL/reference: Through the collaboration object data URL/reference, the workflow user can retrieve and review the collaboration object
- Role/Name of the receiver of the collaboration object (optional): the attribute specifies who can access the collaboration object in the workflow process.

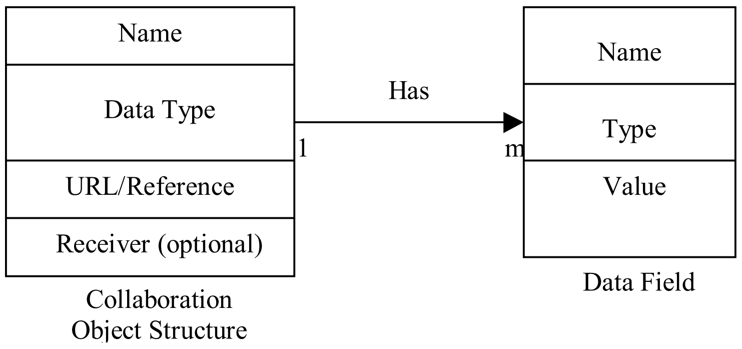


Figure 4.7 Collaboration object structure

The collaboration object structure for one collaborative tool can be specified in the workflow designer. Different collaborative tools may have different collaboration object structure. When we design a workflow process using the workflow designer, we need to decide which collaborative tool will be used in the run-time if there is collaboration in the workflow process. According to the collaborative tool, we should design the specific collaboration object structure. In the METEOR, the run-time system will generate the collaboration object dynamically according to the collaboration object structure in the designer. The collaboration object may be transported in the workflow process and can be accessed by some users in some tasks. In order to achieve the security feature, we just use the security mechanism in METEOR system for normal data object (assign the ACL to the collaboration object in the task). For more detail, please refer to the OrbWork information in the Chapter 5. Figure 4.8 shows how the collaboration object is transported in the workflow process. In Task1, after it finishes the collaboration, the task generates the collaboration object and the collaboration object URL/Reference and passes it to next task(s) with the normal data object. Task2 and Task3 are not allowed to access the collaboration object and its URL/Reference. This can be achieved by the ACL security mechanism, which means the access right of the collaboration object is not assigned to the tasks. But, Task4 has the access right of the collaboration object and it can retrieve and review the collaboration object itself or through the URL/Reference.

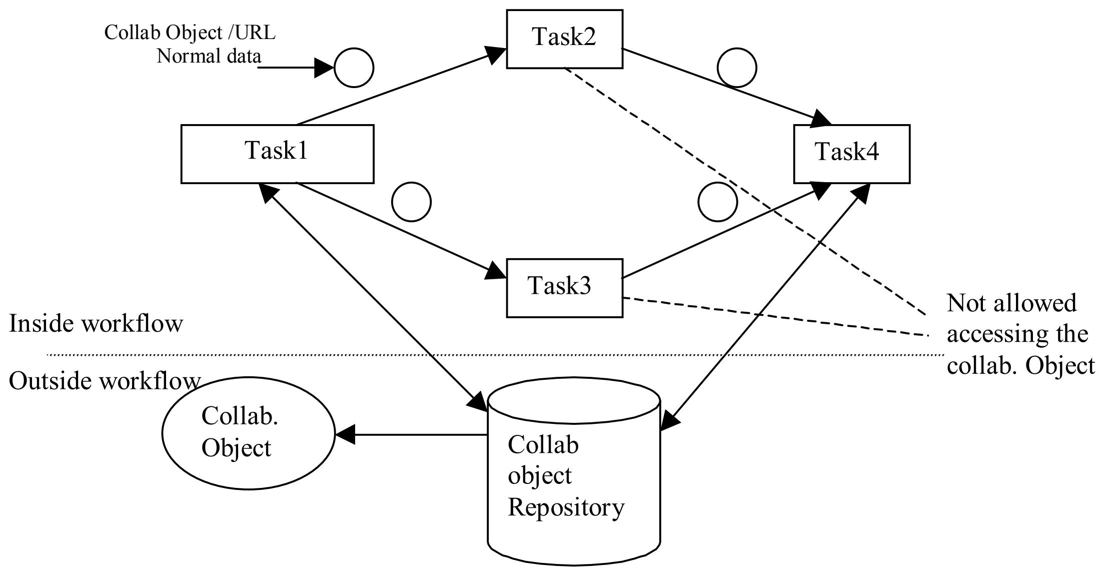


Figure 4.8 Collaboration object transported in the workflow process

4.4 General Interface between METEOR and Collaborative Tools

In order to have a run-time subject-oriented collaboration in the workflow process efficiently, it's better that we can integrate the collaborative tool with the WFMS seamlessly and make the communication between the WFMS and collaborative tool smooth. It requires us to have an interface between the WFMS and the collaborative tool. Our intention in this project is to make the collaboration task model suit for different collaborative tools that can be used in WFMS. Hence, it's ideal that the interface is suit for all collaborative tools. However, in the real world, the collaborative tools are heterogeneous systems. That means different collaborative tools have different architectures, models, implementations and data objects etc. It's hard to design a suit-for-all interface between the WFMS and all possible collaboration systems. What we can do with this issue is to abstract general features of the interface for as many as possible collaborative tools. Based on the general features, we can design the interface between the WFMS and collaborative tools as general as possible.

Let's discuss what functions the general interface can provide? Figure 4.9 shows the relationship among the workflow run-time system, general interface and the collaborative tool. First, the interface should have a method to invoke the appropriate collaborative tool that is defined in the collaboration task in the workflow designer. The method can be either static or dynamic. For dynamic invocation, we can pre-set the invocation method in the system independent of the WFMS. For example, we can have a new file type associate with the new collaborative tools. When we want to have collaboration in the workflow process, we just make the WFMS or the interface to deal with the file type, and the system (such as Windows) can invoke the collaborative tool automatically. The file type for the collaborative tool can be given in the workflow designer after it's registered in the system. For static invocation, we should describe how to invoke the collaborative tool in the interface. For example, we can write a script file to invoke the tool. When having collaboration, we just execute this script file to invoke the collaborative tool. In our opinion, the dynamic invocation is more flexible and easier to be implemented, because there is no need to write the script file for every collaborative tool, instead we just register the file type for it in the system. If the system doesn't support the dynamic invocation, we have to use static invocation.

Second, after the collaboration object is created by the collaborative tool, the object may be transported in the workflow process, which requires the WFMS's run-time system has to know some collaboration information, such as the collaboration object's URL/Reference for accessing or the receiver of the collaboration object (optional). The information could be extracted from the collaboration object structure. So the interface should have extractor functionality.

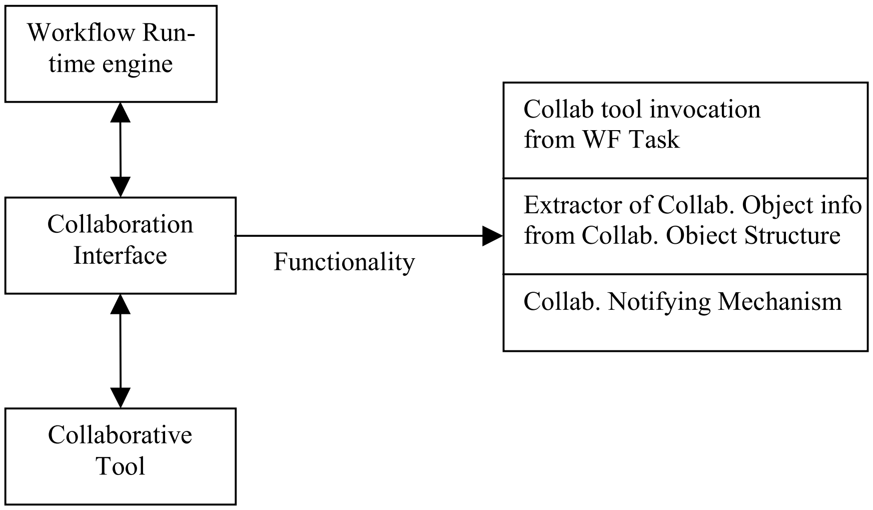

Workflow Run-time engine
Collaboration Interface
Collaborative Tool
Functionality
Collab tool invocation from WF Task
Extractor of Collab. Object info from Collab. Object Structure
Collab. Notifying Mechanism

Figure 4.9 Relationship among Workflow run-time system, General interface and Collaborative tool

Third, the collaboration has the interaction among the participants of the collaboration. One of the participants is in the workflow process. How do we notify the user in the workflow system when there is collaboration reply waiting for him? From this point, the interface should have a notifying mechanism to notify the user in the workflow process. With our research, we feel that this notifying mechanism in the interface between WFMS and the collaborative tool is quite complicated. The notifying mechanism is dependent on the collaborative tools based on the tools' architectures, models and data schemas etc. We can choose one of the two mechanisms to implement it. One is that the interface provides a general module of the notifier. We can think this module is a template having abstract information. For every collaborative tool, we need to customize it within the collaborative tool domain and add some extra information to it. The other is that the interface provides some standard APIs to WFMSs and collaborative tools. Through calling the APIs in the WFMSs and collaborative tools, the interface can connect these two systems to communicate. In this thesis, we choose the first one, module of the notifier, to implement the notifying mechanism. For more information of the implementation, please refer to the implementation in Chapter 6.

CHAPTER 5

ORBWORK ENACTMENT SERVICE FOR METEOR WFMS AND CATCH COLLABORATIVE APPLICATION

METEOR WFMS needs to deal with heterogeneity of platforms within and across cooperating enterprise along with legacy applications and data. Meanwhile, there is increasing demand for advanced feature for supporting mission-critical processes. OrbWork Enactment System for METEOR WFMS, implemented by LSDIS Lab, the University of Georgia, is a CORBA-Based fully distributed, scalable and dynamic workflow runtime system for METEOR WFMS.

CaTCH (Collaboration and Tele-Communication for Healthcare) is an Internet based tele-collaoboration system developed at the LSDIS Lab, the University of Georgia. Its key technical focus is on seamless integration of complementary component technologies and tools to improve the quality of healthcare service. Although the original objective is to improve the quality of healthcare service, CaTCH can be used in other organizations for collaboration. In this thesis, we will integrate CaTCH in OrbWork enactment service for general-purpose METEOR WFMS.

In this chapter, first we provide the overview of OrbWork enactment service for METEOR WFMS. Then we provide the overview of CaTCH system.

5.1 OrbWork Enactment Service for METEOR

The current version of OrbWork, the one of the two implementations of the METEOR workflow management system EAppa➤Enactment service, has been designed to address a variety of shortcomings found in today's workflow systems by supporting the following capabilities:

- provide an enactment system capable of supporting dynamic workflows,
- allow significant scalability of the enactment service,
- integrate disparate distributed and heterogeneous computing environments within and across enterprises,
- utilize open standards, such as CORBA due to its emergence as an infrastructure of choice for developing distributed object-based, interoperable software,
- support workflow interoperability standards, such as JFLOW [OMG98] ,
- utilize Java for portability and network accessibility,
- provide enterprise application and data integration capability in the context of process management,
- provide a user interface via standard Web browsers, for both the workflow end-users and the workflow administrators.

5.1.1 Enactment System of OrbWork

OrbWork provides a fully distributed, scalable enactment system for the METEOR workflow management system. The enactment system has been implemented to support workflows in heterogeneous, autonomous and distributed (HAD) systems. It utilizes the World Wide Web in providing a consistent interface to end-users and workflow administrators from commonly available Web browsers, and also utilizes the HTTP/HTTPS protocol for distribution of task definitions and task routing information.

5.1.1.1 OrbWork Architecture

The architecture of OrbWork enactment system includes the scheduler, workflow specification repository, workflow manager, and the monitor. Figure 5.1 shows an overview of the OrbWork system organization [KSM99].

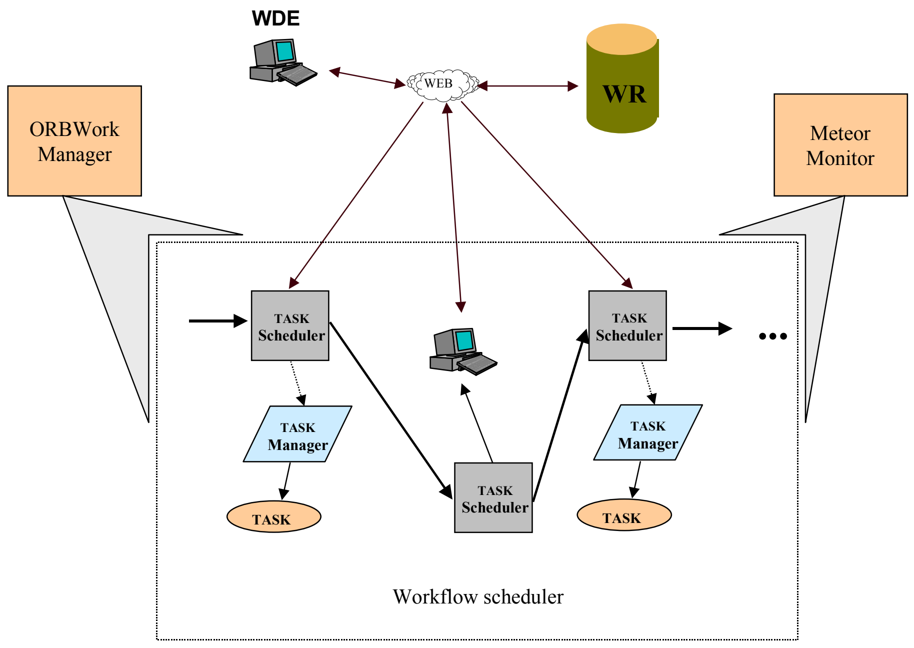


Figure 5.1 OrbWork System Organization

The *scheduler* accesses workflow specifications through the HTTP/HTTPS protocol, directly from the repository. The difference of the scheduler between using HTTP protocol and using HTTPS protocol is the scheduler with HTTPS protocol supports security mechanism in the OrbWork enactment system. The *monitor* records all of the events for all of the workflows being processed by the enactment service. It provides a user interface for the workflow administrator, who can access the information about all of the current workflow instances. The *workflow manager* is used to install new workflow processes (schemas), modify the existing processes, and keep track of the activities of the scheduler. The workflow administrator, using the available interface, controls the existing workflows as well as controls the structure of the scheduler. The structure of the scheduler can be altered by adding more resources, or by migrating fragments of the scheduler to other hosts, for example with lower processing loads. Some schedulers may be replicated, in case the load of workflow instances is too high for a host running just a single scheduler.

OrbWork's scheduler is composed of a number of small schedulers, each of which is responsible for controlling the flow of workflow instances through a single task. The individual schedulers are called *task schedulers*. In this way, OrbWork implements a fully distributed scheduler in that all of the scheduling functions are spread among the participating task schedulers that are responsible for scheduling individual tasks. In this sense, the OrbWork scheduler is composed of a network of cooperating task schedulers. Each task scheduler controls the scheduling of the associated task for all of the workflow instances "flowing" through the task. Each task scheduler maintains the necessary task routing information and task invocation details [KSM99].

As a workflow instance progresses through its execution, individual task schedulers create appropriate task managers that oversee execution of associated tasks. Each workflow instance receives its own task manager, unless the task has been designed to have a worklist, in which case all of the instances are processed by the same task manager.

A workflow is installed by first creating an appropriate workflow context in the Naming Service. (The context is used for storing the object references for all of the participating components.) Then the installation continues by activating and configuring all of the necessary task schedulers and registering them with the Naming Service. All of the component task managers are also registered with the Interface Repository of the underlying ORB.

5.1.1.2 OrbWork Scheduler

OrbWork utilizes a fully distributed scheduler in that the scheduling responsibilities are shared among a number of participating *task schedulers*, according to the designed workflow map. Each task scheduler receives the scheduling specifications at startup from the Workflow Repository (currently, the repository service sends the specifications via the HTTP/HTTPS protocol). Each set of task specifications includes the input dependency (input transitions), output transitions with associated conditions, and data objects sent into and out of the task. In case of the human task (performed directly by end-users), the specifications include HTML templates of the end-user interface page(s). In case of a collaboration task (performed by end-user with collaborative applications), the specifications include HTML templates of the end-user interface page(s) and the detail of its collaboration object structure and the notification template. In case of a non-transactional automatic task (typically performed by a computer program), the specifications also include a task description and the details of its invocation. Finally, in case of a transactional task, the specification includes the details of accessing the desired database and the database query.

When a task is ready to execute, a task scheduler activates an associated task manager. The task manager oversees the execution of the task itself. Figure 5.2 presents a view of the OrbWork's distributed scheduler [KSM99]. Note that scheduling components and the associated tasks and task managers are distributed among four different hosts. The assignment of these components to hosts can be modified at run-time by the workflow administrator.

The partitioning of various components (scheduler's layout), including task schedulers, task managers and tasks, among the participating hosts is flexible. An OrbWork administrator may move any of the components from one host to another. In the fully distributed layout, it is possible to place all of the workflow components on different hosts.

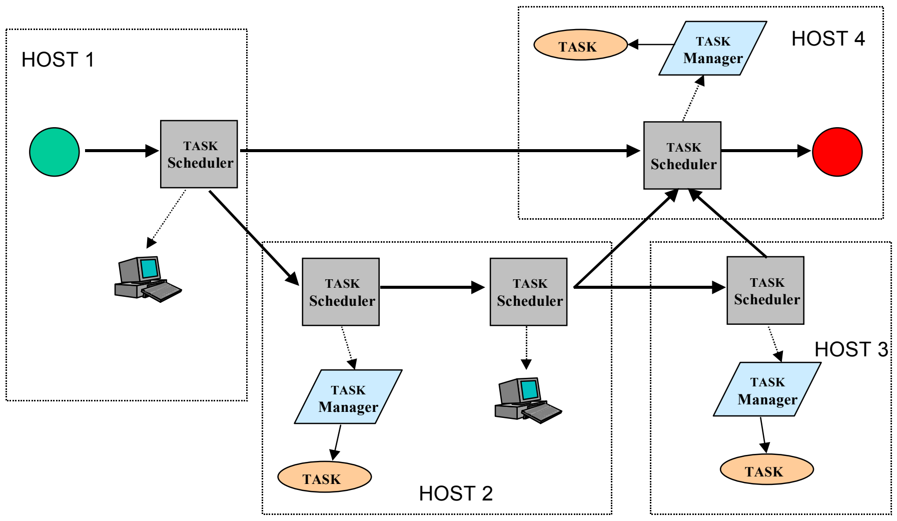


Figure 5.2 OrbWork's Distributed Scheduler

Each task scheduler provides a well-constrained subset of the HTTP/HTTPS protocol, and thus implements a lightweight, local Web server. This enables an OrbWork administrator to interact directly with a selected task scheduler and modify its scheduling specifications from a common Web browser. It also enables the end-user to access workflow instances residing on the task's worklist and collaboration worklist. This set up naturally supports a mobile user.

5.1.2 OrbWork Implementation

One of the most important considerations while designing the OrbWork workflow management system is its flexible and easily modifiable distributed architecture. The current version of the system has been implemented in Java and OrbixWeb 3.2, Iona's CORBA system with Java binding. In addition, Iona's Naming Service has been utilized as a way of providing location transparency for all of the OrbWork components. Using CORBA, and especially Iona's OrbixWeb and Naming Service, as the underlying middleware system offers a number of advantages for implementing a distributed workflow enactment system. In addition to the obvious features provided by CORBA, OrbWork relies on a number of specific services that proved extremely useful in implementing OrbWork. The following table summarizes the features used [KSM99].

Table 5.1 CORBA/Orbix Feature Used in OrbWork

Feature	Application
Dynamic Object Activation	Allows for automatic activation and deactivation of OrbWork components, reducing the load on the host system(s)
Dynamic Invocation Interface (DII)	Only object references are transferred; data object are accessed dynamically, according to their interfaces
Object Loaders	Data objects, task schedulers, and other OrbWork components use loaders to automatically save/restore state
Naming Service	Task schedulers are located with the use of the Name Service; this allows for flexible and transparent placement of the schedulers and their possible migration at runtime

All of the OrbWork components are implemented as CORBA objects. OrbWork relies on the Orbix Activator to start the necessary server when its functions are necessary for the activities of the distributed scheduler and also shutdown the servers once no services have been requested within a specified time interval. In this way, certain portions of a large, distributed workflow (for example those less frequently used) may become inactive, reducing the overhead on the host systems to the necessary minimum.

The implementation detail of collaboration model in OrbWork enactment service will be presented in Chapter 6.

5.1.2.1 Task Schedulers

A task scheduler is implemented as a CORBA object. The IDL interface presented to clients (other task schedulers and other OrbWork components) enables them to invoke various scheduling functions according to the currently loaded specifications. The interface also enables dynamic modifications of the scheduling specifications by reloading from the specification server (repository) or by a direct modification of the specification within the task scheduler.

A task scheduler relies on OrbixWeb Name Service to locate its successors. This enables the OrbWork administrator to dynamically reconfigure the runtime layout of the scheduler by shifting some components between hosts, without introducing any changes to the remaining task schedulers, or workflow instances administered by them.

OrbWork uses the object loader capability supported by OrbixWeb to save/restore the state of a task scheduler. The state includes the necessary information about forthcoming instances (those with still unfulfilled input dependency) and those already on the worklist. As the CORBA object representing a task scheduler is activated (because one of its task predecessors attempts a transfer of the next workflow instance), the necessary scheduling data is automatically reloaded.

5.1.2.2 Task Managers

Task managers control execution of all tasks except human tasks and collaboration tasks (human tasks and collaboration tasks have no associated task managers). Depending on the task type, a task manager is classified as non-transactional or transactional, and is implemented as a CORBA object. A task manager’s IDL interface allows it to be invoked by the corresponding task scheduler. Once activated, the task manager stays active until the task itself completes or generates an exception. Once the task has completed or terminated prematurely with a fault, the task manager notifies its task scheduler. The task scheduler then continues the flow of the workflow instance.

Orbix Activator automatically activates the task manager, only when needed. The communication between the task scheduler and the associated task manager is accomplished by asynchronous (one way) method calls.

A transactional task manager uses JDBC to access the requested data source. Currently, OrbWork provides specific task managers for accessing Oracle and Mini SQL databases. The last of the mentioned task managers allows a uniform access to a wide variety of database management systems (including those on mainframes) from a single task manager.

5.1.2.3 OrbWork Servers

A single OrbWork host runs a number of task schedulers, each of which is implemented as a separate CORBA object. A CORBA object must reside within a CORBA server that typically runs as a single operating system process. In order to save computer resources, a group of OrbWork task schedulers may be placed within a single CORBA server that functions as an OrbWork server. Each OrbWork server is designed to control any number of task schedulers.

A workflow installed on the OrbWork enactment system may utilize any number of heterogeneous hosts (of course, OrbixWeb must be available on each one of them; clients/browsers may be used anywhere). Each of the hosts may have any number of OrbWork servers. However, the most common approach is to keep the number of OrbWork servers close to the number of available processors. Nowadays, some of the available Java virtual machines are able to take advantage of the available processors to run threads. Since the implementation of an OrbWork task scheduler is multithreaded, the question of the number of OrbWork servers may be less critical in that if all of the schedulers are placed within a single server, the schedulers will be able to utilize all of the available processors.

5.1.2.4 OrbWork Manager

The OrbWork Manager is used to install workflows and activate all of the necessary task schedulers. In addition to registering with Orbix Name Service, each task scheduler registers with OrbWork Manager and notifies it of its precise location. In addition, since each task scheduler provides a subset of the HTTP/HTTPS protocol, the scheduler notifies the OrbWork Manager of the precise URL address that the end users and the administrator can use to interact directly with it. The URL address is created when the scheduler is initially installed and it contains the port number that has been assigned to the HTTP/HTTPS server.

The manager is implemented as a CORBA object. It has an IDL interface that allows OrbWork clients to install and administer a workflow as well as create workflow instances. The manager provides an HTTP/HTTPS protocol, so that the same administrative functions can be performed via the Web, from a common browser.

In order to provide an easy access to task schedulers, the OrbWork Manager also functions as a URL redirector, when an end-user wishes to access his task's worklist (for human task or collaboration task). This is necessary since the port number on which the task scheduler's HTTP/HTTPS server is listening is assigned by the system at the time the task scheduler is activated.

It is important to note that the role of the OrbWork Manager is necessary only at the time a new workflow is installed or modified, or when an end-user is connecting for the first time to her designated task. The manager does not participate in any task scheduling activities.

5.1.2.5 Data Object

Data objects are implemented as CORBA objects, providing an IDL interface for accessing all of the defined attributes and methods. As in the case of a task scheduler, the data object implementation uses the object loader to load and save the state of each data object. The CORBA server hosting the data objects is automatically shut down if no data read/write requests arrive within a specified time period, and the dynamic loader saves the state of the object.

As task schedulers implement flow of control within a workflow instance, data objects must be made available at the successor tasks. Instead of the whole object, only its object reference is sent to the task scheduler. When preparing to run the task, the task scheduler accesses the necessary data object(s) (using the Dynamic Invocation Interface) and extracts the relevant attribute values.

5.2 CaTCH Application

The primary technical objective of CaTCH is to investigate the use of rapidly evolving technological development in the fields of desktop video-conference and Internet-based communication and, to provide a reliable, secure, low-cost and low-bandwidth solution enabling tele-consultation in the healthcare industry. But we can also use CaTCH as collaborative application in more industries.

CaTCH provides healthcare information systems solutions involving the integration of technologies such as: Internet/Web and ISDN-based communication environments, desktop video-conferencing and data/application sharing, virtual patient record, intelligent web access and filtering, network computing, and context-supported coordination.

The following highlights some of the key R&D goals, which CaTCH investigated with a view to applications in healthcare [SPP+99].

- Multimedia Collaboration supported by ISDN/POTS/LAN/Internet based desktop conferencing. (e.g. physician-specialist consultation)
- Integrated Multimedia Patient Record sharing.
- Intelligent web based healthcare reference resource support.
- Scheduling/workflow
- Remote environment setup and usability

The CaTCH project can be chronologically classified into two distinct phases. The key functionality difference between the two phases was that in Phase 1, the collaboration mechanism was synchronous and live video conferencing was used for tele-consultation. Phase 2 involved an asynchronous collaboration mechanism and tele-consultation used a "store and forward" technique. We won't present the overview of Phase 1. For complete details on Phase 1, please refer to [V97, SPP+99]. Here we present CaTCH design, architecture and sub-component of CaTCH in Phase 2.

5.2.1 CaTCH Design and Architecture

5.2.1.1 CaTCH Design Consideration

Asynchronous Collaboration is a store-and-forward type of a system. It is not real time and there is no "live" collaboration taking place between the parties. The *store-and-forward approach* offers quite a few advantages over the real time collaboration. First of all it is purely based on the Internet and there is no real time communication overhead. Dedicated communication infrastructure could be leveraged to speed up the entire process. Each consultation object can have a variety of digital heterogeneous information encapsulated within it and since it is not real time its bandwidth can virtually be unlimited. In many cases the store-and-forward approach can be faster than its synchronous counterparts as there is no delay involved in scheduling a suitable meeting time between the collaborators.

Various security options can be enforced on the consultation object. It could be encrypted before its transmission and public/private keys would be needed to access them. The communication can be made more reliable by using some kind of client-server handshake protocol and any loss of data can be immediately replenished by retransmission of the consultation object or a segment of the object.

Another interesting fact to be noted is that the state of the consultation at any instance is the collaboration object. In a synchronous system extra controls need to be set up to monitor the collaboration to provide a consultation summary for later use. In an asynchronous collaboration, the consultation object undergoes multiple annotations by various collaborating partners and the collaboration state can be encapsulated captured within the object itself.

Even though there is a lot of discussion going on about use of Internet as an effective collaboration medium there was one issue that arose in our mind. The kind of bandwidth, reliability and security problems, which were encountered in synchronous collaboration, definitely show that the Internet in its current state today cannot handle a real-time collaboration system. Even if it did, the information being exchanged would not be of diagnostic quality. We need to revert back to dedicated secure communication mediums such as ISDN, DSL and Satellite for diagnostic quality solutions. Given the rapid rate of convergence in telecommunications with broadband options expected in very near future, this can change within a year. On the other hand a store-and-forward asynchronous collaboration technology might be able to use the Internet to provide a secure and reliable consultation if the time delay factor is acceptable to the involved parties.

The key research issues in the asynchronous component of CaTCH are as follows:

- *Data Acquisition*: Leveraging existing technologies to capture medical information pertaining to a patient.
- *Creation of Composite Patient Portfolio*: High-level representation of the patient information.
- *Distributed Data Repository*: Store CPP and all accompanying patient artifacts.
- *Agent Based information transfer*: Use of intelligent agents to provide specific functions.

5.2.1.2 CaTCH Architecture

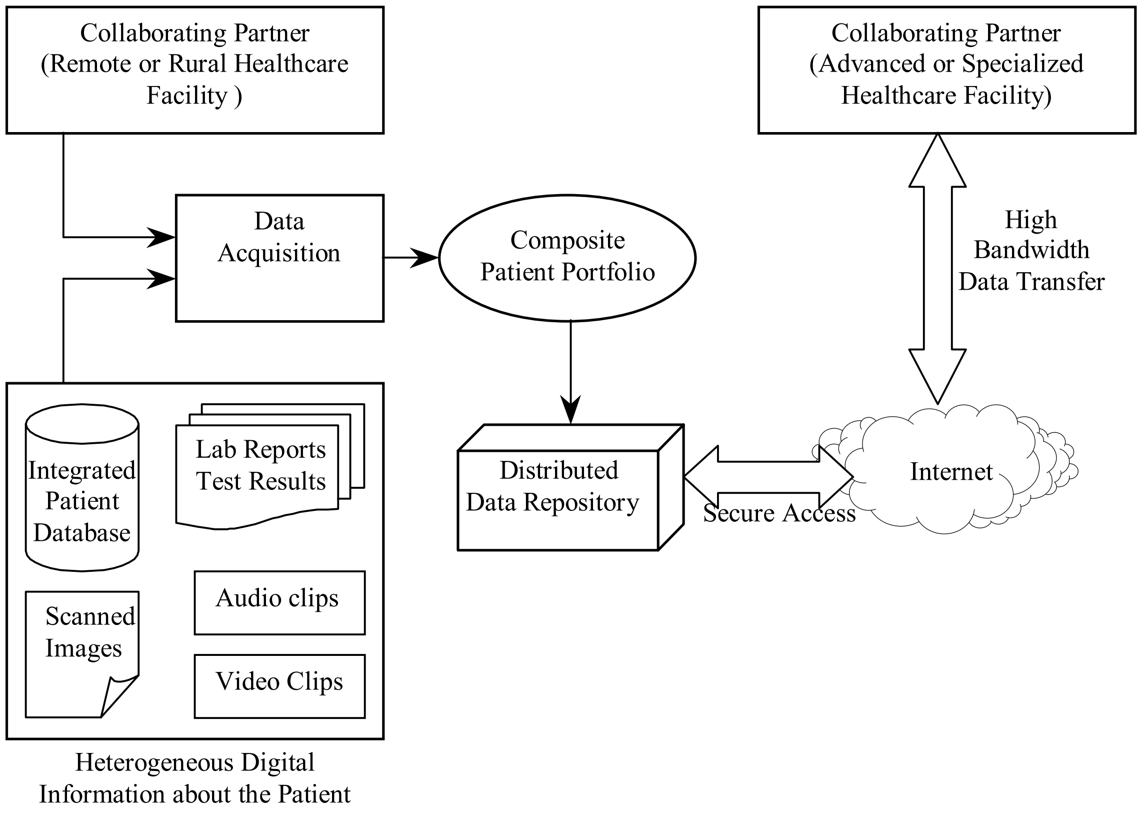

Collaborating Partner (Remote or Rural Healthcare Facility)
Collaborating Partner (Advanced or Specialized Healthcare Facility)
Data Acquisition
Composite Patient Portfolio
High Bandwidth Data Transfer
Integrated Patient Database
Lab Reports Test Results
Scanned Images
Audio clips
Video Clips
Distributed Data Repository
Secure Access
Internet
Heterogeneous Digital Information about the Patient

Figure 5.3 Architecture of Asynchronous component of CaTCH

The CaTCH architecture follows the client-server paradigm. Figure 5.3 illustrates the architecture. The DDR (Distributed Data Repository) represents the server component of the architecture. There are two types of "clients" that interact with the server. One of them is the CaTCH-client and it is used for data acquisition, CPP (Composite Patient Portfolio) creation and transferring data to the DDR. The other client is an Internet browser (Internet Explorer 5.0) and is used for viewing the CPP and other patient multimedia information.

5.2.2 Data Acquisition

Each physician could associate a multitude of heterogeneous multimedia information with a patient. These could be video clips, X-Ray images, sonograms etc. In order to transmit the multimedia information as a part of the consultation, they have to be digitized from their native source. The collaborating physician should have certain basic digitization equipment, such as video capture boards, to capture video clips transmitted from analog tapes (scanners to scan images and X-Rays, microphones to capture audio etc.)

5.2.3 Composite Patient Portfolio (CPP)

Once the data acquisition is completed, it needs to be transmitted over the Internet to a universally accessible repository. The CPP is a representation of the heterogeneous digital multimedia information that needs to be transmitted. It is more or less like a map of the information that describes a particular physician's consultation request. The CPP is encoded using XML. The XML notation was chosen because it is a widely accepted standard for describing syntactic schemas

The patient multimedia information could be of heterogeneous types, file formats and encoding. The CPP represents these. The CPP contains placeholders for the consulting doctors to attach their comments regarding a particular case. The same CPP is exchanged back and forth between the consulting physicians till the consultation is closed. Hence, at any instance of the consultation, the CPP gives us the state of the current collaboration.

The CPP basically contains all the information necessary for the physician to describe a particular patient's condition during the consultation. Metadata information about the various media elements attached by the physician is stored in the CPP. This metadata is used for ensuring for the proper display of the multimedia information. They can also be used for security purposes since the metadata provides encryption information (if any) of the attached data. The CPP and the patient's multimedia information are transmitted to the data repository where it is stored.

5.2.4 Distributed Data Repository (DDR)

The DDR is an Internet accessible repository where all the collaboration related information is stored. The collaborating partners store the CPP and patient's multimedia information (collaboration objects) on the DDR. The DDR has a web server component, which serves these objects over the Internet using the HTTP protocol. These can then be viewed using a web browser and the multimedia information is displayed using appropriate plug-ins. The following outline the primary tasks performed by the DDR:

- *Universal Accessibility*: The DDR is assigned a unique domain name and is universally accessible to all collaborating partners.
- *Security for data being transported*: Using Secure Socket Layer (SSL) and the HTTPS protocol the web server can be configured to deliver encrypted information.

- *User authentication*: Login and password requirements are placed on all users of the system and only valid users are allowed to access the patient information.
- *Reliability*: The DDR should be very reliable with a very high "uptime".

The DDR forms the "server" component in CaTCH's client-server architecture. Availability of XML management and querying tools could facilitate browsing, cataloging and querying of the CPP. In the future a minimal WFMS can be used to maintain *worklists* for individual doctors who participate in the collaboration for scheduling and managing collaboration sessions.

5.2.5 Agent Based Information Transfer

The DRR ensures adequate security for information going out of it. But secure and reliable means have to be established for storing information into the repository. Agents can be used for information transfer between the remote health centers and the DDR's.
An agent at the remote physician's site takes the responsibility of transferring the consultation information and the CPP to the DDR. It establishes contact with the agent at the DDR, authenticates itself and starts sending the information. A *three-way handshake* mechanism is used by the agents for the information transfer. The DDR agent uses the CPP as a guide map for the rest of the patient information. If there was any lost byte of data, it asks for a retransmission. The information could also be sent encrypted and it could be decrypted at run-time at the DDR agent using private key of the accessing user.

CHAPTER 6

IMPLEMENTATION OF COLLABORATION MODEL IN ORBWORK ENACTMENT SYSTEM

One of this thesis's goals is to implement the Collaboration task model in *METEOR* WFMS. A prototype implementation of the Collaboration task in the *METEOR* system has been completed. It has been tested with a workflow application example: Healthcare. First we present the implementation and test environment for the thesis. We use *OrbWork* enactment system for Windows NT 4.0 as the run-time system in METEOR WFMS, *OrbixWeb3.2* for Windows NT 4.0 from Iona, Inc. as the ORB (Object Request Broker) and *CaTCH* as the collaborative tool. Both systems, *OrbWork* and *CaTCH*, were implemented in Java at LSDIS Lab, University of Georgia. We choose *Internet Explorer 5* (*IE5*) from Microsoft as the web browser because *CaTCH* requires the browser to display the XML file directly and the security mechanism of *OrbWork* works only in *IE5*. The database management system is *MINI SQL* (*mSQL2.05*) from Hughes Technologies for both systems. We also use *Apache 1.3.6* server as the web server and *Netforge* from Novocode Software&Networking as the servlet engine with *Apache* server for CaTCH. The web server, servlet engine and DBMS all reside in Solaris UNIX operating system. The Client tool of *CaTCH* is implemented under Windows system. Under the environment, we implement the Collaboration task model in *OrbWork* system. It includes designing the specification file format for the workflow design tool and run-time system, implementing collaboration worklist server, collaboration object structure and collaboration notifier in general interface etc. We will discuss them in detail in next sections.

6.1 The Specification Files for Collaboration Task

From the *METEOR* WFMS model, we know that the workflow specification files are used to transfer workflow process information from a design-time tool to a run-time system. The specification files can be generated by the workflow designer to store all the information of the workflow process. The run-time system loads them to get the information to execute the workflow process. The specification files should have all the information the run-time system needs, such as task information, date object, input parameters, output parameters and routing information, etc. Figure 6.1 shows the functionality of the specification files in the WFMS.

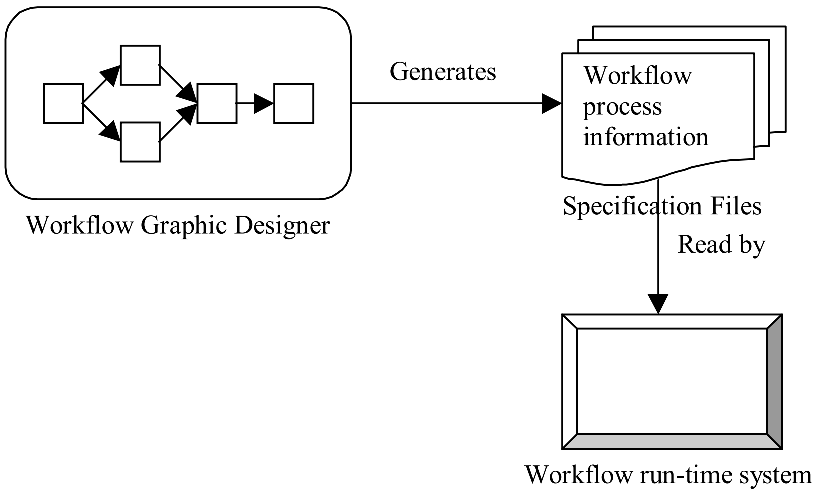


Figure 6.1 Functionality of the specification files in the WFMS

All the specification files of a workflow process will be generated and stored in the specific directory of the *OrbWork* run-time system. In order to present them clearly, we need to define some names for convenience. Let's suppose the *OrbWork* run-time system is in **G:\OrbWork** directory, which can be named **ORBWORK_HOME**. All the specification files of the workflow processes are stored in directory **G:\OrbWork\public_html\wflows**, which can be named **RUNTIME_DESTPATH**. We can also take the Healthcare application as an example, so the **WFLOW_NAME** is Healthcare.

In current *OrbWork* system, there are two types of specification files. One type is for workflow scope and the other is for individual task scope. First we present the files for workflow scope. Then we discuss the files for individual task scope.

6.1.1 Specification Files for Workflow Scope

The workflow designer generates specification files for workflow scope, including **classes**, **hosts**, **start**, **tasks**, and **compile.bat**. They are in the **RUNTIME_DESTPATH\WFLOW_NAME** directory (e.g. G:\OrbWork\public_html\wflows\Healthcare). Information in specification files is defined for the workflow process level. In order to implement the collaboration task model, we need to represent the task type as *COLLABORATION* in **tasks** file. Figure 6.2 shows an example **tasks** file and Figure 6.3 shows the format of **tasks** file.

```
13
Healthcare      NETWORK      pentax    OrbWork
PatientRegister      HUMANCOMPUTER      pentax    Healthcare
RetrievePatientData      TRANSACTIONAL      pentax    Healthcare
MedicalTest      NONTRANSACTIONAL      pentax    Healthcare
CollabDecision_AnalyzeSymptom      HUMANCOMPUTER      pentax    Healthcare
AnalyzeSymptom_Collab      COLLABORATION      pentax    Healthcare
AnalyzeMedicalReport      HUMANCOMPUTER      pentax    Healthcare
CollabDecision_GenerateSolution      HUMANCOMPUTER      pentax    Healthcare
GenerateSolution_Collab      COLLABORATION      pentax    Healthcare
WritePrescription      HUMANCOMPUTER      pentax    Healthcare
Billing      NONTRANSACTIONAL      pentax    Healthcare
StorePatientData      TRANSACTIONAL      pentax    Healthcare
StopProcess      HUMANCOMPUTER      pentax    Healthcare
```

Figure 6.2 Example of **tasks** file

```
N (Number of tasks)
Task1_Name    Task1_Type    Task1_Host    Task1_Parent_Task_Name
……

TaskN_Name    TaskN_Type    TaskN_Host    TaskN_Parent_Task_Name
```

Figure 6.3 Format of **tasks** file

6.1.2 Specification Files for Individual Task Scope

Each task in a workflow process has its own specification files to record the specific task information, such as the input parameters, output parameters, routing information, security information, etc. Some files are defined for all task types, such as **create**, **data**, **foutputs**, **inputs**, **soutputs**, **admin.html** and **index.html**. The information in these files is for input parameters, output parameters, routing information and data object in the task. They are in the **RUNTIME_DESTPATH\WFLOW_NAME\tspecs\TASK_NAME** directory (e.g. G:\OrbWork\public_html\wflows\Healthcare\tspecs\PatientRegister). Other specification files are dependent on the task type. For example, in *Human* task, there are some html template files (**station.html** and **TASK_NAME.html**) to display the data object in the workflow process. In *Transactional* task, there is a **task** file to store the DBMS information, SQL commands and input and output of the SQL commands. These specification files are mainly stored in the direcortory **RUNTIME_DESTPATH\WFLOW_NAME\tspecs\TASK_NAME\task** (e.g. G:\OrbWork\public_html\wflows\Healthcare\tspecs\PatientRegister\task). The workflow graphic designer can generate the general and task type-related specification files for every task according to the task type in the workflow process.

As discussed in chapter 4, we know *Collaboration* task is similar with the *Human* task. We can let the workflow designer generate almost the same specification files as those in *Human* task. We only add some files dependent on the *Collaboration* task type and changed some files to form the specification files of *Collaboration* task.

The general specification files for an individual task are **create**, **data**, **foutputs**, **inputs**, **soutputs**, **admin.html** and **index.html**. The file formats of them are the same for all task types including Collaboration tasks. We don't need to change them.

The specification files dependent on the *Collaboration* task (suppose the task name is AnalyzeSymptom_Collab) are **acl**, **CollabObjInvocation**, **CollabObjStructure**, **CollabMIMEType**, **detain**, **localhandlers**, **params**, **type**, **TASKNAME.html** (AnalyzeSymptom_Collab.html), **TASKNAME-View.html** (AnalyzeSymptom_Collab-View.html) and **station.html**. They are all in the directory **RUNTIME_DESTPATH\WFLOW_NAME\tspecs\TASK_NAME\task** (e.g. G:\OrbWork\public_html\wflows\Healthcare\tspecs\PatientRegister\task).

Among those files, the format and function of **acl**, **detain**, **localhandlers**, **params**, **station.html** are the same as if the task is *Human* task. The designer can generate these specification files for the *Collaboration* task in the same way as for the *Human* task. On the other hand, the files, **CollabObjInvocation**, **CollabObjStructure**, **CollabMIMEType**, **type**, **TASKNAME.html** (AnalyzeSymptom_Collab.html) and **TASKNAME-View.html** (AnalyzeSymptom_Collab-View.html), are specialized only for the *Collaboration* task. These files represent the special attributes of the Collaboration task model discussed in Chapter 4. We will discuss implementation detail below.

CollabObjStructure file has the data structure of the collaboration object. It's used for workflow run-time system to create the collaboration object. This file is related with the collaborative tool.

Figure 6.4 shows the format of this file and Figure 6.5 shows an example file for *CaTCH* tool.

```
N (number of fields in the object)
Field1_Name    Field1_Type    Field1_Default_Value (Optional)
…
FieldN_Name    FieldN_Type    FieldN_Default_Value (Optional)
```

Figure 6.4 Format of **CollabObjStructure** File

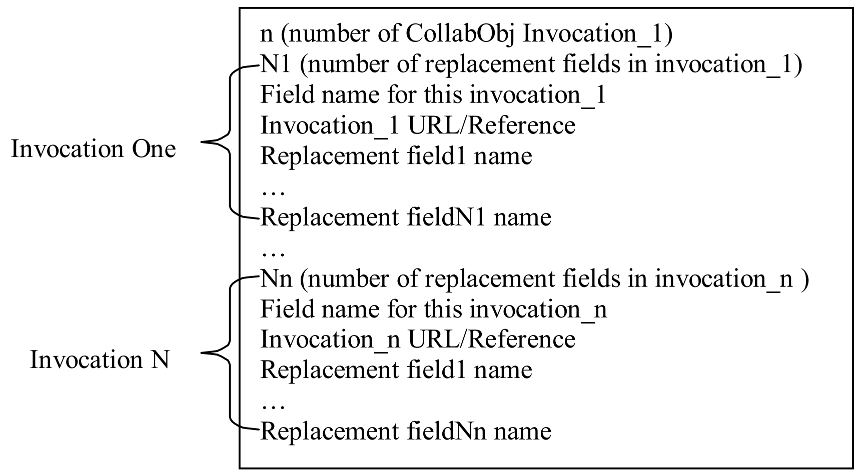
```
6
Sender string
Receiver string
Patient string
Subject string
Day string
File string
```

Figure 6.5 Sample of **CollabObjStructur**e File for *CaTCH* tool

In the format of **CollabObjStructure** file, the first line is the number of the fields in the collaboration object. Next is the field information part. Each line is for one field. Each field may have three parts: field name, field type and field default value. The first two, field name and field type, are required and the last one, field default value is optional. For example, in Figure 6.5, this is the **CollabObjStructure** file for *CaTCH* tool. It has 6 fields in one CaTCH collaboration object. Every field only has two required parts, field name and field type. There is no field default value in the fields.

CollabObjInvocation file is used for workflow users to access the Collaboration object. This file represents the URL/Reference attribute in the Collaboration Object Structure of the Collaboration task model (Please see Figure 4.7). It is related to the collaborative tool. Figure 6.6 shows the format of this file and Figure 6.7 shows an example file for *CaTCH* tool.

In the format of the **CollabObjInvocation** file, it allows users to access the collaboration object through several ways. But generally, there is one way to access the object in collaborative tool for most time. The first line “n” is the number of the collaboration object invocation methods. For every invocation, there are four parts of information. The first part is the number of replacement parameters in the invocation Reference/URL. The second part is the field name for users to access the collaboration object. The third part is the invocation URL/reference. If the invocation has replacement parameter(s), we uses %1%, %2% … to represent the replacement parameter positions in the URL/Reference. The fourth part is the filed(s) name of replacement parameter(s). Each line in this part represents one replacement parameter. In the run-time system, %1%, %2%… will be replaced by the values of these fields in the URL/Reference. There is one point we should note that all of the field names in **CollabObjInvocation** file should appear in the **CollabObjStructure** file also.



```
n (number of CollabObj Invocation_1)
N1 (number of replacement fields in invocation_1)
Field name for this invocation_1
Invocation_1 URL/Reference
Replacement field1 name
…
Replacement fieldN1 name
…
Nn (number of replacement fields in invocation_n )
Field name for this invocation_n
Invocation_n URL/Reference
Replacement field1 name
…
Replacement fieldNn name
```

Figure 6.6 Format of **CollabObjInvocation** file

```
1
1
File
http://orion.cs.uga.edu:5080/~catch/data/%1%/%1%.xml
File
```

Figure 6.7 Sample of **CollabObjInvocation** file for *CaTCH* tool

For example, in Figure 6.7, we have one invocation method for *CaTCH* tool. In the invocation, we have one replacement parameter. The field name for accessing the object is “File” and invocation URL is an HTTP request and there are two positions for the replacement parameter: “File” field. The information in the **CollabObjInvocation** file should be provided when we design the collaboration task of the workflow process.

CollabMIMEType file defines the MIME type of the collaborative tool in the Windows system. OrbWork enactment system will load this file and get the MIME type to invoke the collaborative tool. Because OrbWork has its own HTTP server or HTTPS server, OrbWork will deal with the collaborative tool invocation URL to send back the HTTP response in the corresponding MIME type. The web browser gets the response in the specific MIME type and looks up the Windows system to find out which application can be used to process the response and then invokes this application automatically. This MIME type should be associated with the collaborative tool in the Windows system separately. Figure 6.8 shows the **CollabMIMEType** file for CaTCH tool.

```
application/x-catch
```

Figure 6.8 **CollabMIMEType** file for CaTCH tool

We have already discussed three specification files, **CollabObjStructure** file, **CollabObjInvocation** file and **CollabMIMEType** file that are related to the collaborative tool in the individual task scope. Now we will discuss the specification files related to the WFMS run-time system in the individual task scope. They are **type**, **TASKNAME.html** and **TASKNAME-View.html** files.

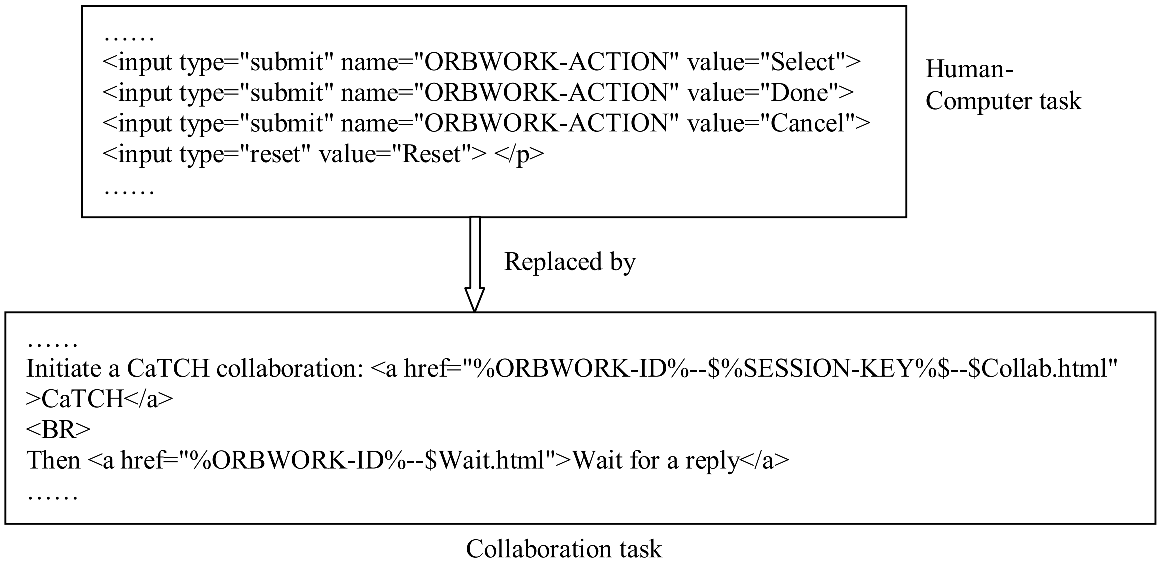
The format of **type** file is very simple. It only has one line: task type. If the task is *Collaboration* task, there is **COLLABORATION** in the first line of the file. Figure 6.9 shows the example **type** file for the *Collaboration* task.

```
COLLABORATION
```

Figure 6.9 Example of **type** file for the *Collaboration* task

In the current OrbWork enactment system, the *Human* task has two types of html template files to display the data object. One is like TASKNAME.html (e.g. PatientRegister.html) and the other is like TASKNAME-View.html (e.g. PatientRegister-View.html). TASKNAME-View.html is just used to display the data object, which is in the read-only status. The user can't edit the data fields in this html file. But TASKNAME.html allows the user to edit the data fields of the object. For every *Human* task, the OrbWork system use TASKNAME-View.html first to display the data fields. After the user chooses "Select" button in the html file, OrbWork system uses TASKNAME.html to let user edit the data fields. We borrow this idea and apply it to the *Collaboration* task. We also have two types of html template files for it, TASKNAME.html and TASKNAME-View.html. They are very similar with those in the *Human* task. We only need to modify them a little.

For *Collaboration* task, OrbWork system uses TASKNAME-View.html to display the data fields and allows the user to initiate collaboration (invoking the collaborative tool). After he initiates collaboration, the Collaboration task goes to *Wait for Reply* status. The user can interact with the collaboration partner in this status. Then after the collaboration is finished, OrbWork system uses TASKNAME.html to let the user edit the data object and continue the workflow process. In **TASKNAME-View.html**, we should replace the "Select", "Done", "Reset" and "Cancel" buttons with Collaboration invocation links. Figure 6.10 shows the replacement in the **TASKNAME-View.html**. For *Collaboration* task, we provide the URL link to invoke the collaborative tool. This URL (%ORBWORK-ID%--$%SESSION-KEY%$--$Collab.html) will be handled by OrbWork's HTTP server or HTTPS server. Then an HTTP/HTTPS response is sent back in the MIME type defined by the **CollabMIMEType** file. With the MIME type, the collaborative tool will be invoked automatically. Parameters %ORBWORK-ID% and %SESSION-KEY% will be replaced with detail information at run-time. %SESSION-KEY% is the unique key for the collaboration session for the task in one workflow instance. %ORBWORK-ID% is the workflow instance id in the run-time.



```
……
<input type="submit" name="ORBWORK-ACTION" value="Select">
<input type="submit" name="ORBWORK-ACTION" value="Done">
<input type="submit" name="ORBWORK-ACTION" value="Cancel">
<input type="reset" value="Reset"> </p>
……
```

```
……
Initiate a CaTCH collaboration: <a href="%ORBWORK-ID%--$%SESSION-KEY%$--$Collab.html"
>CaTCH</a>
<BR>
Then <a href="%ORBWORK-ID%--$Wait.html">Wait for a reply</a>
……
```

Figure 6.10 Replacement in **TASKNAME-View.html** between *Human-Computer* task and *Collaboration* task

In current implementation of the Collaboration task model, the **TASKNAME.html** for *Collaboration* task is the same as that for *Human* task. The workflow designer can generate this file in the same way as that for *Human* task.

6.2 The Collaboration Worklist Server

Just like the worklist manager server in current OrbWork enactment system manages the worklist for the *Human* task, we implement *collaboration worklist manager server* to manage the collaboration worklist for the *Collaboration* task. These two worklist manager servers are independent in the OrbWork enactment system. Although their functions and implementation are similar, the *collaboration worklist manager server* has its own features for the *Collaboration* task.

The *collaboration worklist manager server* is implemented as a CORBA object. It's registered in the OrbWork enactment system when OrbWork enactment system is setup (Meteor.OrbWork.SetupOrbWork). Actually it acts as an HTTP server or HTTPS server with security features in the OrbWork (we may define the port number for the HTTP server or HTTPS server in the OrbWork property file and will discuss the OrbWork property file in the later section). The IDL interface allows it to be invoked by OrbWork server, Collaboration task scheduler and other OrbWork components. Once registered in the beginning of OrbWork setup, Collaboration worklist manager server stay active to listen the port to get the HTTP or HTTPS request from users until OrbWork enactment system shuts down. Figure 6.11 shows the IDL interface of *Collaboration worklist manager server*.

```
interface CollabWorklistIf
{
  exception CollabWorklistError{ string msg; };
  //install support for handling a Collaboration task
  void InstallTask( in string    WorkflowName,
                    in string    TaskName,
                    in string    URL );
  // post an collaboration work item to this worklist
  void PostItem( in string    WorkflowId,
                 in string    WorkflowName,
                 in string    TaskName,
                 in idata     Data );
};
```

Figure 6.11 IDL interface of *Collaboration worklist manager server*

In the interface, **InstallTask** function allows the OrbWork server to invoke it to install a Collaboration task scheduler when we install a workflow application in the OrbWork enactment system and the task is a *Collaboration task*. The parameter URL is the root URL for all workflow application, which is defined in the OrbWork property file as OW_WFLOW_URL. **PostItem** function allows the *Collaboration task scheduler* to post a *Collaboration worklist item* to the worklist when the task is executed at run-time. Parameter Data is a two-dimension string array of name-value pairs. We will discuss *Collaboration task scheduler* and *Collaboration worklist item* in detail in the next two sections.

In general, the *Collaboration worklist manager server* works as an HTTP or HTTPS server depending on whether OrbWork enactment system supports security feature. Figure 6.12 shows how it works in the OrbWork enactment system.

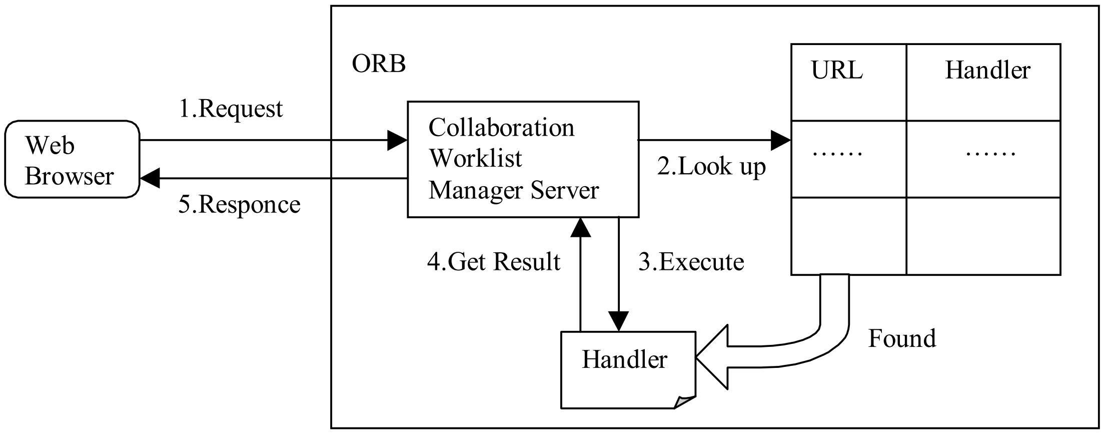


Figure 6.12 Collaboration worklist manager server

When a *Collaboration* task is executed in a workflow application instance, the user (1) sends an HTTP/HTTPS request through the web browser to the *Collaboration worklist manager server*. *Collaboration worklist manager server* will deal with the request. First, it analyzes the requested URL and (2) looks up the corresponding handler in the URL resource table. These handlers are Java methods registered in the URL resource table when the workflow application is installed (these handlers are for all workflow instances) or a workflow instance is executed (these handlers are workflow instance related). After getting the handler, *Collaboration worklist manager server* (3) invokes it and (4) gets the result from it. Then it composes an HTTP/HTTPS response and (5) sends it back to the client.

The handler resources are registered in two ways for two different types. One is for all workflow instances, that means It's workflow type related. **InstallTask** function is responsible for the work when the workflow application is installed in OrbWork. The other is only for one specific workflow instance. **PostItem** function is responsible for the work when the instance is executed. For example, every collaboration session in a workflow instance has a unique session key that is generated when the instance is executed. So the handlers for the URLs with the session key should be registered only in the **PostItem** function.

6.3 The Collaboration Task Scheduler

As described in Chapter 3, each task has its own task scheduler to manage the execution in OrbWork enactment system. Its responsibility includes getting input parameters from previous task(s), checking the invocation conditions, invoking the corresponding task manager, generating output parameters and executing the transition to next task(s).

For the *Collaboration* task, we have implemented the *collaboration task scheduler* to manage its execution. Besides the general features for all task types whose work is done by Meteor.OrbWork.BaseTaskScheduler, it has its own features for *Collaboration* task. The implementation for *Collaboration* task, Meteor.OrbWork.CollabTaskScheduler, is much similar with the implementation for *Human* task scheduler, Meteor.OrbWork.UserTaskScheduler. It needs to:

- Communicate with the Collaboration worklist manager server
- Get Collaboration worklist item
- Send worklist item
- Get data object value from previous task(s)

- Compose the HTTP/HTTPS response with the data object
- Invoke the execution of the task and schedule the transition.

6.4 The Collaboration Worklist Item

In OrbWork enactment system, every *Human* task in a workflow application has a worklist, which is a list of worklist items. One worklist item represents an instance of the task in the workflow application. Similarly, every *Collaboration* task has a collaboration worklist of the collaboration worklist items. One collaboration worklist item represents an instance of the collaboration task in the workflow application. Collaboration worklist item tells the workflow user what he will do in the workflow process and collaboration, and what is the status of the collaboration. We can use workflow name, task name to identify a collaboration worklist for the task and use instance ID to identify the corresponding collaboration worklist item. Meteor.OrbWork.CollabWorklistItem, the implementation of *Collaboration worklist item*, has the following information:

- Workflow name, task name, instance ID
- Data object and collaboration data (we will discuss the collaboration object in the next section)
- Collaboration session(s): Since a collaboration task in a workflow instance may have several collaboration sessions, the worklist item may have several sessions too. Each session represents one collaboration and has its own object for the collaboration. In current implementation, we support only one collaboration session for the collaboration task in an instance.
- Collaboration status: the status in the collaboration session.
- Collaboration session key: It identifies the collaboration session in the collaboration task. Currently, we have one session key for the worklist item

because every collaboration task in an instance has one collaboration session in our implementation.

- Collaboration notifier: It works as the notification between the OrbWork enactment system and the collaborative tool. We will discuss it in detail in the later section.

6.5 The Collaboration Object

As described in Chapter 4, Collaboration object structure is a principle part of the Collaboration model of METEOR WFMS. In OrbWork enactment system, we implement the collaboration object structure in Meteor.OrbWork.CollabObjItem. The implementation supports one collaboration object item for one collaboration session. In Figure 4.7, we show the collaboration object structure. In the implementation, the Data Type in the structure is a vector of Data fields. Each Data Field is also a vector of four attributes: name, type, value and URL/Reference for invocation. The URL/Reference attribute is optional and doesn't apply to every field. Only the field name appearing in the **CollabObjInvocation** specification file will have the URL/Reference value. The OrbWork enactment system will access the collaboration object through the field's URL/Reference. The collaboration objects will be created in OrbWork enactment system according to the information in the **CollabObjStructure** and **CollabObjInvocation** specification files described in the first section of this Chapter.

6.6 The Collaboration Notifier

In the Collaboration model of METEOR WFMS, there is a collaboration notification mechanism in the general interface described in Chapter 4. We mentioned the notification mechanism in the interface between WFMS and the collaborative tools is complicated because it's dependent on the architecture model and data schema of the collaborative tools. There are two methods to implement the notification mechanism. One is that the interface provides a general module of the collaboration notifier. This module is a template having basic structure and abstract information generated by the workflow designer. For each collaborative tool, we need to customize it according to the collaborative tool's architecture. In this thesis, we choose this method to implement the notification mechanism. For CaTCH tool, we also customize the module with CaTCH's architecture and model.

The other method is that the interface provides APIs to WMFS and collaborative tools. Through the APIs called in the WFMSs and collaborative tools, the interface can connect these two systems to communicate with each other.

In current implementation, the workflow graphic designer may generate the notifier module with the specification files. Figure 6.13 shows the basic structure of the module. It's a Java subclass of thread.

```
package Meteor.OrbWork;
// This class is supposed to complete by the user after generated by designer
public class CollabNotifierThread extends Thread
{
   String wflow_name;
   String task_name;
   String instance_name;
   String sessionKey;
   URL    CollabObjStructureUrl; // url for collab obj structure
   URL    CollabObjInvocationUrl; // url for collab obj invocation
   CollabWorklistImpl collabWorklistMgr;
   CollabWorklistItem collabWorklistItem;
   protected MeteorMonitor mm = null;

   public CollabNotifierThread(CollabWorklistImpl cw, CollabWorklistItem cwi, String wn,
String tn, String in, String k, URL uStructure, URL uInvocation){}

   public void run(){}
}
```

Figure 6.13 Basic structure of the notifier module

The core function of the module is **run()**. Different collaborative tools may have different notifiers according to their implementation. This function should be customized for the collaborative tool after it's generated by the workflow designer. Figure 6.14 shows the basic structure of **run()** function. This thread checks with the collaborative tool every n seconds, which is given in workflow designer (it's 60 seconds in Figure 6.14), and gets the collaboration object(s) from it.

```
public void run()
   {
     while(true)
     {
       //While(has more collaboratin object)
       {
         CollabObjItem coi = new CollabObjItem(CollabObjStructureUrl, CollabObjInvocationUrl);
         for(int i = 0; i < coi.getFieldNo(); i++)
         {
           Vector f = coi.getFieldAt(i);
           /*******
             get v from collaborative tool for the collaboration object, v is a string
          ********/
           f.setElementAt( v, CollabObjItem.FIELD_VALUE );
           objKey += v;
           coi.setFieldAt(i, f);
         }
         coi.setInvocationToField();
         // current has only one session
         collabWorklistItem.AddObjItem(collabWorklistItem.GetSessionKey(), objKey, coi);
       } //END while of more collaboration objects
       try
       {  sleep((int)(60*1000));
       }
       catch (InterruptedException e){}
     }
   }// end run()
```

Figure 6.14 Basic structure of **run()** function

6.7 The Other Changes in OrbWork Enactment System

As we implemented the *Collaboration* model in OrbWork enactment system, we have to add more variables for the C*ollaboration* task in the OrbWork property file, which should in the OrbWork class path directory. These variables are all related with the Collaboration task. Figure 6.15 shows the additional variables in the OrbWork.properties file.

```
OW_COLLAB_WORKLIST_PORT=9004
OW_COLLAB_WORKLIST_SECURE=false
OW_COLLAB_WORKLIST_SECURE_PORT=9006
OW_COLLAB_NOTIFIER_SUFFIX=CollabNotifier
```

Figure 6.15 Additional Variables in OrbWork.properties file

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

Workflow technology cuts across the boundaries of organizations to reach a large number of users, resources and tools, to offer tremendous opportunities for streamlining business processes, and gain unique competitive advantages in software reengineering. Workflow management offers a powerful technique to cooperate, integrate and automate different business tasks for today's organizations, yet preserves the diversity of these tasks for specialized functions used by the organizations. However, most WFMSs (either commercial or research prototype) offer little or no support for collaboration, and this support is critical to most business processes. Such limitations of current workflow products prevent them from communication with outside collaboration partners within their workflow processes. It also prevents them from becoming the backbone of corporate computing, especially for mission critical application approach.

In this thesis, we have designed collaboration model in METEOR WFMS. In the collaboration model, we add collaboration task to the task model and introduce the collaboration object structure to the model. In addition, we design a general interface between METEOR WFMS and collaborative tools. With the collaboration model, we can integrate collaborative applications in the METEOR WFMS. Our effort in this area also gives some valuable experience to those WFMSs that want to support collaboration in their systems.

The other major contribution of this thesis is that we have implemented a practical prototype of the collaboration model in OrbWork enactment system for METEOR. In the implementation, we develop the collaboration worklist server, collaboration task scheduler, the collaboration worklist item, collaboration object and collaboration notifier according to the collaboration model for the METEOR WFMS. In addition, we design the specification files between the workflow enactment system and the workflow designer. This prototype implementation has been achieved successfully and demonstrated with the Healthcare workflow application. This implementation also shows an example of using CORBA technologies.

7.2 Future Work

Finally, in this section, we outline some suggestions to improve the design of the collaboration model and implementation of the prototype as well as future research possibility.

- Current workflow graphic designer for METEOR hasn't supported the collaboration task. We need to add the appropriate GUI to the workflow designer and let it generate the corresponding specification files for the workflow application.
- The current prototype works well with CaTCH collaborative application, but haven't tested it with other collaborative applications, such as Microsoft's NetMeeting. We can apply the prototype to other collaborative applications to test whether the collaboration model and the prototype in METEOR work well.
- The current prototype hasn't been tested with the exception handling mechanism and dynamic feature of OrbWork enactment system. In the future, we can run the workflow application in the OrbWork with whole features.

- In the collaboration model, we propose that there are two ways to implement the collaboration notification mechanism. In current implementation, we use notifier template and let the user customize it with the specific collaborative tool. In the future, we can try the other way in that the interface provides APIs to both enactment system and collaborative tools.
- Besides the run-time collaboration in WFMS, we can develop the design-time collaboration in WFMS build-time. It could be document sharing and collaborative designing.

BIBLIOGRAPHY

[A+97] G. Alonso, D. Agrawal, A. El Abbadi, C. Mohan. *Functionalities and Limitations of Current Workflow Management Systems*, IEEE Expert (Special Issue on Cooperative Information Systems), (Vol.12, No.5), 1997.

[ADT99] S. Arpinar, A. Dogac, and N. Tatbul. *An Open Electronic Marketplace through Agent-based Workflows: MOPPET*, Intl. Journal on Digital Libraries, 1999.

[BJS96] C. Bussler, S. Jablonski, H. Schuster. *A New Generation of Workflow Management Systems: Beyond Taylorism with MOBILE*. ACM SIGOIS Bulletin 17(1) 1996 17-20.

[C99] Collaborative Strategies LLC. *Electronic Collaboration on the Internet and Intranets.* http://www.collaborate.com/publications/intranet.html.

[CO99] CommenceOne. *Enabling the Business-to-Business Trading Web Using marketSite 3.0 Open Market Platform*. White Paper, March 1999.

[ED97] Ahmed Elmagarmid and Weimin Du. *Workflow Management: State of the Art vs. State of the Market*, In Advances in Workflow Management System and Interoperability.

[F95] L. Fischer. *The Workflow Paradigm – The Impact of Information Technology on Business Process Reengineering*. Future Strategies, Inc., Alameda, CA, 2nd edition, 1995

[G99] S. Gallagher. *Person to Person: Collaboration over the wire.* Enterprise Development, October 1999

[GAP97] N. Guimaraes, P. Antunes, A. Pereira. *The Integration of Workflow Systems and Collaboration Tools.* In Advances in Workflow Management System and Interoperability

[GR93] J. Gary and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers, 1993

[GHS95] D. Georgakopoulos, M. Hornick and A.Sheth. *An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure*. Journal of Distributed and Parallel Database Systems, 3(2):119-153, April 1995

[HC99] I. Hilerio and W. Chen. *Herbal-T, Enabling Integration, Interoperability, and Reusability of Internet Components.* Proceeding of the International Joint Conference on Work Activities Coordination and Collaboration, 1999

[IBM98] IBM. *IBM MQSeries Workflow Concepts and Architecture*, Version 3.1 release 1, product number 5697-FM3, 1998.

[ILGP96] Y. Ioannidis, M. Livny, S. Gupta, N. Ponnekanti. *ZOO: A Desktop Experiment Management Environment*. In Proc. 22nd Intl. Conf. On Very Large Database 1996, 274-285.

[J98] S. Jajodia. Interview: Amit Sheth on Workflow Technology, in IEEE Concurrency, April-June, 1998, pp. 21-23.

[K99] K. Kochut. *METEOR Model version 3*, draft LSDIS Lab, the University of Georgia, 1999

[KLB97] A. Khetawat, H. Lavana and F. Brglez. *Collaborative Workflows: A Paradigm For Distributed Benchmarking and Design on the Internet.* NCSU Technical Report, February 1997.

[KLP99] A. Koufman-Frederick, M. Lillie, L. Pattison-Gordon, D. Watt, R. Carter. Electronic Collaboration: A Practical Guide for Educators.

[KS95] N. Krishnakumar and A. Sheth. *Managing Heterogeneous Multi-system Tasks to Support Enterprise-wide Operations.* The Journal on Distributed and Parallel Database Systems, 3 (2), April 1995

[KSM99] K. Kochut, A. Sheth and J. Millor. *ORBWork: "Optimizing Workflow" Using a CORBA based, fully distributed process to create scalable, dynamic systems.* Component Strategies, March 1999, pp. 45-57.

[L99] *Electronic Collaboration: A Practical Guide for Educators*, Northeast and Islands Regional Educational Lab at Brown University.

[Lin97] C. Lin, *A Portable Graphic Workflow Designer*, M.S. Thesis, Department of Computer Science, University of Georgia, May 1997.

[LW99] H. Ludwig and K. Whittingham. *Virtual Enterprise Co-ordinator—Agreement-Driven Gateways for Cross Organisational Workflow Management.* Proceeding of the International Joint Conference on Work Activities Coordination and Collaboration, 1999

[M+96] E. Mesrobian, R. Muntz, E. Shek, S. Nittel, M. LaRouche, M. Kriguer. *OASIS: An Open Architecture Scientific Information System*. In Proc. 6th Intl. Workshop on Research Issues in Data Engineering 1996, 107-116.

[MLK96] R. McClatchey, J. Le Goff, Z. Kovacs. *An Application of Workflow Management and Product Data Management Conventions in a Distributed Scientific Environment*. Manuscript, 1996

[N+91] J. Nunamaker, A. Dennis, J. Valacich, D. Vogel and J. George. Electronic meeting systems to support group work, Communications of the ACM, 34(7),

[O98] Ontology Org. *Reference Architecture for iMarkets*, Part 1 Overview. 1998

[OMG98] Object Management Group. *Workflow Management Facility*, OMG document bom/98-06-07, 1998

[PEL97] H. Pozewaunig, J. Eder, W. Leibhart. *ePERT: Extending PERT for Workflow Management System.* Advances in Databases and Information Systems (ADBIS) 1997: 217-224.

[S97] Amit Sheth. *From Contemporary Workflow Process Automation to Adaptive and Dynamic Work Activity Coordination and Collaboration.* Proceeding of the Workshop on Workflows in Scientific and Engineering Applications [keynote talk/invited paper], Toulouse, France, September 1997.

[S98] B. Schlicher. *Applying CORBA in the Enterprise.*

[SGJ$^+$96] A. Sheth, D.Georgakopoulos, S. Joosten, M. Rusinkiewicz, W. Scacchi, J. Wileden and A. Wolf. Report from the NSF workshop on workflow and process automation in information system. *SIGMOD Record*, 25(4):55-67, December 1996.

[She95] A. Sheth. *Tutorial notes on workflow automation: Apllication, technology and research*. ACM SIGMOD Conference, May 1995

[SJo96] A. Sheth and S. Joosten. *Workshop on Workflow Management: Research, Technology, Products, Applications and Experiences*, August 1996.

[SK97] A. Sheth and K. Kochut. *Workflow Applications to Research Agenda: Scalable and Dynamic Work Coordination and Collaboration System.* Proceedings of the NATO Advanced Study Institute on Workflow Management Systemsand Interoperability, August 1997.

[SKM+96] A. Sheth, K. Kochut, J. Miller, D. Worah, S. Das, C. Lin, D. Palaniswami, J. Lynch and I. Shevchenko. *Supporting State-Wide Immunization Tracking Using Multi-Paradigm Workflow Technology*. In Proc. Of the 22nd Intl. Conf. On Very Large Database (VLDB96), September 1996.

[SPP+99] A. Sheth, K. Parasuraman, S. Poreddy, S. Velmurugan, W. Karp and S. Crane. CaTCH: Collaboration and Tele-Consulting in Healthcare. UGA Technical Report, 1999

[SRG94] L. Stein, S. Rozen, N. Goodman. *Managing Laboratory Workflow with LabBase*. In Proc. 1994 Conf. On Computers in Medicine.

[SVA99] A. Sheth, W.M.P. van der Aalst, I.B. Arpinar. *Processes Driving the Networked Economy: Process Portals, Process Vortexes, and Dynamically Trading Processes*, IEEE Concurrency, July-September 1999

[VW97] Gottfried Vossen and Mathias Weske. *The WASA Approch to Workflow Management for Scientific Application*. In *Advances in Workflow Management System and Interoperability.*

[WfMC94] Workflow Management Coalition: Workflow Reference Model, Workflow Management Coalition Standard, WfMC-TC-1003, 1994

[Wod96] D. Wodtke, J. Weissenfels, G. Weikum, A. Kotz Dittrich. *The Mentor Project: Steps Towards Enterprise-Wide Workflow Management*. In Proc. 12th IEEE Intl. Conf. On Data Engineering 1996 556-565.

[WS97] D. Worah and A. Sheth. *Transactions in Transactional Workflows*. In Advanced Transaction Models and Architectures, S. Jajodia and L. Kerschberg, Eds., Kluwer Academic Publishers, 1997.

[X97] Weixiong Xu. *NEOWORK: A Reliable, CORBA-Based Workflow Enactment System For METEOR$_2$*, Master thesis of the University of Georgia, 1997.

[Zheng97] *K. Zheng, Designing Workflow Processes in METEOR Workflow Management System.* M.S. Thesis, LSDIS Lab, Computer Science Department, University of Georgia, June 1997.