ACHIEVING DYNAMIC INTER-ORGANIZATIONAL WORKFLOW MANAGEMENT BY INTEGRATING BUSINESS PROCESSES, E-SERVICES, EVENTS, AND RULES

By

JIE MENG

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2002

Dedicated to my parents and husband.

ACKNOWLEDGMENTS

First, I would like to express my gratitude towards Dr. Stanley Y.W. Su, chairman of my supervisory committee, for his continuous guidance, advice, and support throughout the course of my Ph.D. study, and for giving me the opportunity to work in the Database Systems R&D Center. My great appreciation also goes to Dr. Abdelsalam Helal, co-chairman of my supervisory committee, for constantly providing me with valuable comments and suggestions during the dissertation work. I would like to thank my supervisory committee members--Dr. Herman Lam for his constant help, suggestions, and valuable discussions, and Dr. Joachim Hammer and Dr. Sherman Bai for their precious time. Thanks also go to Sharon Grant for making the Database Center such a pleasant place to work.

My wholehearted gratitude goes to my parents, Qingren Meng and Zhenping Ma, for their unconditional love and continuous encouragement throughout my studies, especially during these last two years. I also thank my husband, Chuntao Liao, whose love and support helped me overcome many challenges during my Ph.D. study.

Finally, I thank all the colleagues and friends who helped me with inspiring discussions and also for making my stay at the Database Systems R&D Center so enjoyable. I wish them all the best in their studies and their future endeavors.

This research is supported by the NSF grant #EIA-0075284.

TABLE OF CONTENTS

page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

ACHIEVING DYNAMIC INTER-ORGANIZATIONAL WORKFLOW MANAGEMENT BY INTEGRATING BUSINESS PROCESSES, E-SERVICES, EVENTS, AND RULES

By

Jie Meng

May 2002

Chairman: Dr. Stanley Y.W. Su
Cochairman: Dr. Abdelsalam Helal
Major Department: Computer and Information Science and Engineering

As the global marketplace becomes more and more competitive, business organizations often need to team up and operate as a virtual enterprise in order to utilize the best of their resources for achieving their common business goals. Since the business environment of a virtual enterprise is highly dynamic, it is necessary to develop a workflow management technology that is capable of handling dynamic workflows across enterprise boundaries.

This dissertation describes a dynamic workflow model and a dynamic workflow management system for modeling and controlling the execution of inter-organizational business processes. In this work, all the sharable tasks performed by people or automated systems in a virtual enterprise are defined and published as e-services. The business process models of inter-organizational workflows are defined in terms of, among other things, compositions of e-services provided by the participating organizations. A

dynamic workflow model (DWM) is introduced to enable the specification of dynamic properties associated with a business process model. It extends the underlying model of the Workflow Management Coalition's Workflow Process Definition Language (WPDL) by adding connectors, events, triggers, and rules as its modeling constructs, encapsulating activity definitions and allowing e-service requests to be included as a part of the activity specification. The workflow management system makes use of a business event and rule server to trigger business rules during the enactment of a workflow process model to enforce business constraints and policies and/or to modify the process model at run-time. We also introduce a constraint-based, dynamic service binding mechanism to dynamically bind e-service requests to e-services that satisfy some constraint specifications.

CHAPTER 1
INTRODUCTION

The advent of the Internet, World Wide Web, and distributed computing technologies has enabled business organizations to conduct business electronically: thus, the e-business was born. According to a white paper from Morgan Stanley Dean Witter [Phi00], there have been three major phases in the evolution of e-business technology. E-business first appeared first in the form of EDI (Electronic Data Interchange) [Ada98]. The second phase of e-business was basic E-commerce, where retailers sell their products through their Web sites. Phase three of e-business, currently in progress, is in the form of eMarketPlace "portals," which bring trading partners with related interests together into a common community (i.e., an exchange) to match buyers and sellers and provide other services to serve their interests. The current trend is toward collaborative e-business in which business organizations form virtual alliances under rapidly changing market conditions and make use of the best of their resources for achieving their common business goals. Business processes and application system services of the participating organizations are important resources that need to be used to conduct a joint business. Workflow technology is an enabling technology for managing, coordinating, and controlling the activities of a virtual enterprise.

Workflow technology allows people and companies to model business processes and to control the execution of these processes. Based on the definition given by the Workflow Management Coalition (WfMC) [WfM99b], a business process is a set of

activities that collectively realizes a business goal, normally within the context of an organization. Workflow is the automation of a business process, in whole or in part, during which documents, information, or tasks are passed from one participant to another for actions according to a set of procedural rules. A workflow management system (WfMS) defines, creates, and manages the execution of workflows.

The term workflow originated in the mid-1980s and became popular in the early 1990s. Many vendors have introduced general-purpose workflow management systems. In addition to the general-purpose workflow management systems, some other fields co-opt workflow capabilities [She99]: e.g., Enterprise Resource Planning (ERP) and Enterprise Application Integration (EAI) [Oba01]. With the emergence of e-business and virtual enterprises, interest in workflow technology has escalated. In order to coordinate different organizations in a virtual enterprise in a highly dynamic business environment, a workflow management technology used in e-business environment has to be able to handle workflows that are subject to changes and are distributed across organization boundaries.

To meet the above requirement, we have developed a dynamic workflow model (DWM) to enable the introduction of dynamic properties (i.e., active, customizable, flexible, and adaptive properties) in a business process model. Here, DWM is the mechanism for modeling business processes. It is analogous to the relational data model used to model databases. It is an extension of the underlying model of the Workflow Management Coalition's Workflow Process Definition Language (WPDL [WfM99a]) by adding event, trigger, and rule to WPDL's modeling constructs. By doing so, we integrate business events and business rules with business processes. The enactment of a

business process can post events to trigger the processing of business rules, which may in turn enact other business processes (i.e., the active property). The processing of business rules may also enforce customized business constraints and policies (i.e., customizable property) and/or dynamically alter the process model at run-time (adaptive property). We also separate control information (i.e., split and join) from activity definitions so that each activity definition is encapsulated and reusable. In this work, we treat all the sharable tasks performed by people or automated systems in a virtual enterprise as electronic services (e-services). The e-services are categorized according to different business types, for which standardized e-service templates are defined. Service providers register their e-services with a broker by using the templates. E-service requests are specified in the activity definitions of a process model. Their specifications are also based on the e-service templates. E-service requests are bound to the proper service providers at run-time by using the services of a broker to identify the suitable providers (i.e., the flexible property). By using the above dynamic binding approach, process models are separated from (i.e., not bound to) the specific service providers when they are defined. Changes in the membership of a virtual enterprise (i.e., its service providers and their services) will not affect the specifications of process models because the virtual enterprise has multiple choices in finding suppliers and business partners. The integration of business processes with business events and rules and the dynamic binding of e-services to service providers give the proposed workflow management system its dynamic properties.

We have designed and implemented a dynamic workflow management system called DynaFlow, which is part of an information infrastructure being developed for supporting the Internet-based Scalable E-business Enterprise: the ISEE information

infrastructure [Su01b]. We shall call the virtual enterprise that uses the ISEE information infrastructure an ISEE virtual enterprise, and call the workflow system for an ISEE virtual enterprise an ISEE workflow system. This dissertation presents a dynamic workflow model and the implementation of a dynamic workflow management system. An early version of the system was reported in our previous publication [Men02].

The organization of this dissertation is as follows. In Chapter 2, research related to workflow and virtual enterprise is surveyed. In Chapter 3, the architecture of DynaFlow is introduced. The e-service specification, dynamic workflow model, and the extensions to WPDL are explained in Chapter 4. The design and implementation of the key components of DynaFlow are given in Chapter 5. Chapter 6 summarizes our research contributions and gives some suggestions for future research.

CHAPTER 2
RELATED WORK

We begin in this chapter with the introduction of the WfMC's Workflow Reference Model. We then discuss the achievements and challenges in the workflow area from two aspects: workflow design and workflow enactment. At the end of this chapter, we introduce workflow research that is related to virtual enterprise and e-business and e-service (web service) related technologies.

2.1 WfMC's Workflow Reference Model

The Workflow Management Coalition (WfMC) was founded in August 1993. It is a not-for-profit, international organization of workflow vendors, users, analysts, and university/research groups. The Coalition's mission is to promote and develop the use of workflow through the establishment of standards for software terminology, interoperability, and connectivity between workflow products.

The Workflow Reference Model has been developed from the generic workflow application structure. The model identifies the interfaces within this structure, enabling products to inter-operate at a variety of levels [WfM95]. The major components and interfaces in the Workflow Reference Model can be seen in Figure 2-1.

The Coalition has developed a framework for the establishment of workflow standards based on this reference model. The framework includes five categories of interoperability and communication standards that will allow multiple workflow products to coexist and inter-operate within a user's environment. The five categories of interfaces are the following:

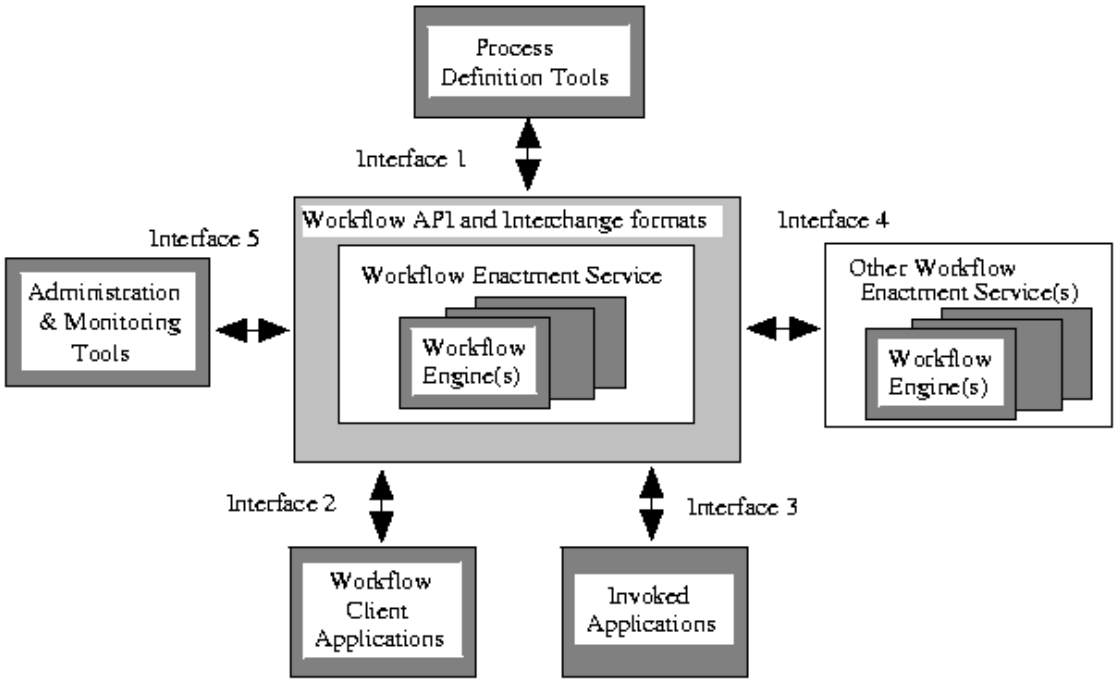


Figure 2-1: Workflow Reference Model--components and interfaces.

Interface 1 includes a common meta-model for describing the process definition and a textual grammar for the interchange of process definitions (the Workflow Process Definition Language or WPDL). It also includes APIs for the manipulation of process definition data.

Interfaces 2 and 3 specify standard workflow management APIs that can be supported by WFM products. These APIs provide a consistent method of access to WFM functions in cross-product WFM Engines.

Interface 4 provides an abstract specification that defines the functionality required to support the interoperability between different workflow engines.

Interface 5 specifies what information needs to be captured and recorded out of the various events that occur during a workflow enactment.

2.2 Workflow Design

Workflow applications begin with workflow designs. Conceptually, a workflow model consists of three different models: a process model, an information model, and an organization model. Among them, the process model is the core. The main task of a workflow design is to define process models.

A process model is an abstract or graphic representation of an organizational process. Process modeling is a procedure that produces a process model, which serves as the basis for a workflow specification. We focus our discussion on business process models since our work focuses on business applications.

2.2.1 Process Modeling Methodologies

There are basically two categories of methodologies to model a business process, namely, communication-based methodologies and activity-based methodologies [Geo95; She96]. Communication-based methodologies stem from the Conversation for Action model proposed by Winograd and Flores [Win87], in which the process of coordination is represented as a finite-state machine. In these methodologies, an action in a workflow is represented by the communication between a customer and a performer. The resulting organizational process reveals a social network in which a group of people, assuming various roles, fulfills an organizational process.

Activity-based methodologies focus on modeling activities instead of modeling the communications among people. In this approach, the overall process is decomposed into tasks that are ordered based on the dependencies among them. An organizational process modeled with this approach is then reduced to a network of activities and subprocesses. Workflow management systems typically adopt activity-based methodologies. For example, in IBM's MQ Workflow [Ley94; IBM00], a process

consists of activities, including program activities, process activities (subprocess), and block activities (i.e., a set of activities that can be repeated until an exit condition is met). The dependencies among these activities are captured by control connectors (control dependencies) and data connectors (data dependencies).

Like most workflow systems, we adopt activity-based methodologies for process modeling in our workflow system.

2.2.2 Process Specification Languages and Definition Tools

Process models are specified in some workflow specification language. A workflow specification language uses rules, constraints, and/or structural constructs to capture the dependencies among the activities, and uses activity attributes to describe the properties of activities. No matter which way a workflow system specifies a process model, the activity structure (control flow) and the information exchange between activities (data flow) are necessary parts of process modeling.

Most workflow management systems use a graphic specification language for process modeling. In a process graph, activities are connected by arrows and routing elements. Examples of such systems include the Build-time of IBM's MQSeries Workflow [IBM00], the process modeling environment of Vitria's Businessware [VIT01], and METEOR Designer of METEOR2 (developed at the University of Georgia) [She97].

Some workflow management systems provide rule-based or constraint-based specification languages. For example, EvE project [Gep98] uses events and Event-Condition-Action (ECA) rules as the fundamental concepts for defining and enforcing workflow logic. Other workflow specification methods include Petri Nets [Mur89], Finite State Automata [Pel94], and Computation Tree Logic [Eme90]. A new kind of

high-level Petri net, XML Net, is proposed by Lenz and Oberweis [Len01] to support inter-organizational process modeling.

In addition to specifying control flows and data flows in process models, a workflow specification language may also support the specification of other properties of business processes. For example, the modeling language in the WIDE project [Cas96] supports specifications of exception handling and transactional properties. With respect to exception support, the model specification allows the definition of ECA rules to support exceptional and asynchronous behavior during workflow enactment. With respect to transactional support, the model specification provides transactional constructs so that the designer can group different activities to achieve atomicity, and also allows the possibility of defining compensation activities for those activities that need to be rolled back. In the CrossFlow project [Gre99; Cro00], in addition to defining the normal workflow structure, execution alternatives called flexible elements, can also be defined in a process model. Different from the OR-split/join structures, the decision of whether or not these flexible elements are executed at run-time is based, not on the transactional conditions, but on the global goal of the workflow.

To capture dynamic properties of business processes, the dynamic workflow model and the dynamic workflow management system presented in this dissertation integrate both business events and business rules with business processes. Synchronous events are open points in a process model where the business rules can be attached and be activated to adapt to a changing business environment. This is a new approach to support dynamic workflow.

2.2.3 WfMC's Workflow Process Definition Language (WPDL)

Although most workflow management systems define their process models using activity-based methodologies, workflow specifications for process models vary greatly in different workflow systems. This can become a hurdle in the integration of different workflow systems. The need for standardization arises. Among current standardization efforts, WfMC's WPDL [WfM99a] is the well-accepted one.

In WfMC's Workflow Reference Model, Interface 1 includes a common meta-model for describing the process definition and a textual grammar for the interchange of process definitions (WPDL) [WfM99a]. Interface 1 focuses on the specification of a process definition meta-model. This meta model identifies the basic set of entities and attributes used in the exchange of process definitions. In this model, a workflow process contains one or more workflow process activities. The activities are associated with workflow applications and workflow participants. Transitions that connect these activities determine the control flow of a workflow process. Transitional conditions can be defined to specify the flow or execution conditions. A variety of attributes describe the characteristics of this limited set of entities. Based on this model, vendor specific tools can transfer models via a common exchange format. The top-level entities contained in the workflow process definition are shown in Figure 2-2.

WPDL is a language for describing the process meta-model Interface 1. In the WfMC's document [WfM99a], the WPDL grammar is given in EBNF (Extended Backus Naur Form).

In our work, we extend the underlying workflow model of WPDL by adding event, rule, trigger and connectors to its set of modeling constructs and allowing e-service requests to be a part of the activity specification. These extensions are made to support

dynamic inter-organizational workflow management. We will describe these extensions in Chapter 4.

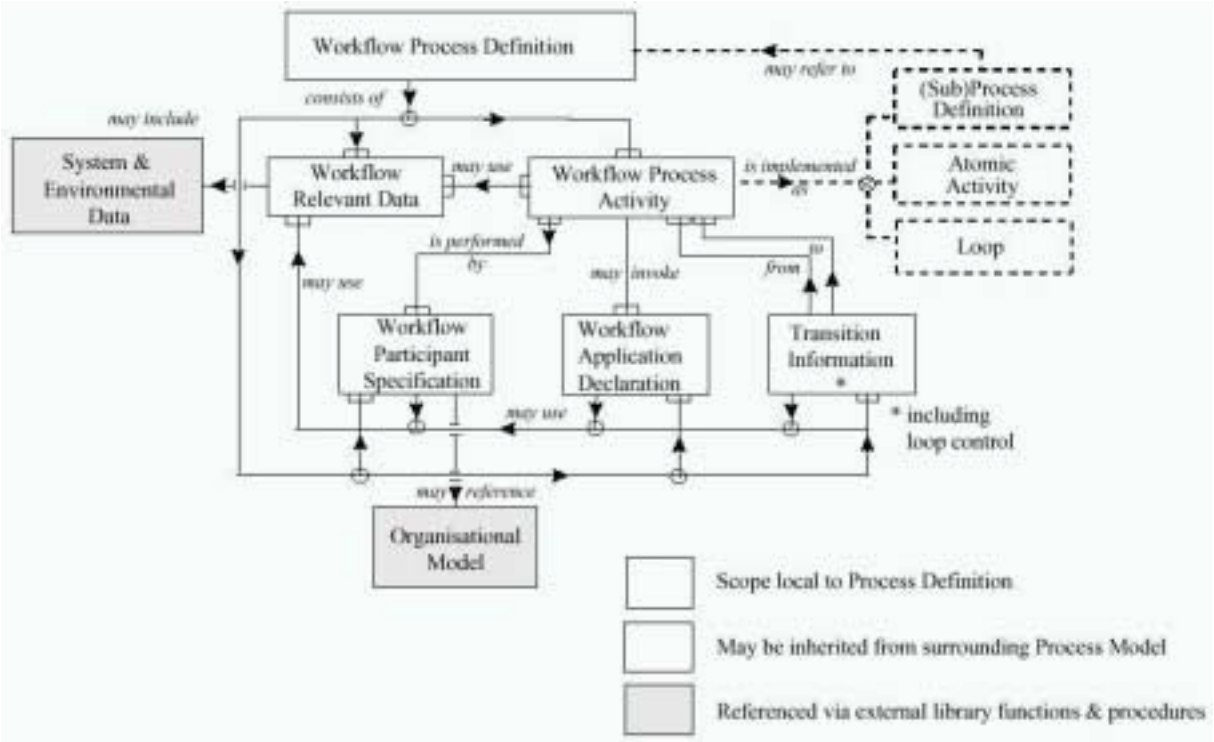


Figure 2-2: Top-level entities in workflow process definition of the Workflow Process Definition Language (WPDL).

2.2.4 Organizational Structure

In the workflow model, the organizational structure captures the relations between roles (resource classes based on functional aspects) and groups (resource classes based on organizational aspects) [She99]. Traditional workflow systems are designed for a specific organization, so it is relatively simple to define an organizational structure. For example, MQSeries Workflow provides a complete mechanism to be used to define the organizational model [IBM00]. However, for workflow systems that are used for a virtual enterprise, the workflow processes will not be defined for a single organization. Workflow processes for these workflow systems will cross organizational boundaries. This requires a new paradigm to capture the organizational structure for these workflow systems. Although research is currently being conducted in inter-organizational

workflow, a good solution to this problem has yet to emerge. Some systems have tried to solve it by adopting trading communities or broker mechanisms [Gre98; Alo99; Str00]. A virtual enterprise modeling approach is proposed in [Dav99].

In our work, the services provided by participants in the virtual enterprise are standardized as e-services according to some predefined e-service templates. These e-services are categorized into different business types according to the roles they play in a virtual enterprise. E-service requests in a process model are defined based on some standardized e-service templates. A broker server is used to manage these e-services. The binding of an e-service request to an e-service and its provider is done at run-time.

2.3 Workflow Enactment

Workflow enactment service controls the instantiation of processes and sequencing of activities, according to the process model information. The workflow enactment service may consist of one or more workflow engines, which manage(s) the execution of individual instances of the various processes [WfM95]. Normally, there are two main components in a workflow engine, namely, workflow scheduler and activity handler. The workflow scheduler enforces inter-activity dependencies and coordinates the execution of activities in the workflow. The activity handler handles the execution of an activity instance.

2.3.1 Enactment System Architecture

Most existing commercial workflow systems use centralized architecture to implement the workflow engines. The run-time architecture is centralized in the sense that a single workflow engine handles the execution of a workflow instance. MQ Workflow [MQW] adopts the centralized control of one workflow instance. It employs a three-tiered client-server architecture and incorporates external applications that perform

geographically distributed tasks on different platforms. The advantages of the centralized architecture include lightweight clients, easy monitoring and auditing, simpler synchronous mechanisms, and overall design simplicity. However, there are also many shortcomings, namely, a single point of failure, performance limitations, scalability problems, and so on [Alo97]. To solve these problems, alternative architectures are proposed and implemented. Sheth et al. [She99] summarizes some alternative architectures, including a fully distributed architecture, a client-server or web-enabled distributed architecture, a mobile agent architecture, a distributed object-based architecture, and a lightweight agent-based architecture.

An example of the fully distributed architecture is METEOR2 [She97]. In METEOR2, two fully distributed workflow enactment systems (ORBWork and WEBWork) are implemented. In these systems, there is no single entity responsible for scheduling tasks. The scheduling information is distributed among the individual task managers. Another example of a distributed architecture is WIDE [Cer97]. WIDE provides a distributed workflow enactment system based on an active database-management system. The distributed architecture of the workflow enactment service can enhance reliability and scalability. Other workflow systems with distributed run-time architecture include EvE [Gep98], Mentor [Mut98], and Exotica [Alo95].

Still another example is WW-FLOW [Kim00], which uses a web-enabled distributed architecture to accomplish the workflow enactment service. Through its modular design and run-time encapsulation, a complex process can be executed by multiple workflow engines on the web.

Another interesting approach is mobile agent workflow. An example of this architecture is the Migrating Workflow developed by the University of Houston [Cic98].

In the Migrating Workflow architecture, a migrating workflow instance transfers its code (specification) and its execution state to a site, negotiates a service to be executed on its behalf, receives the result, and moves on. Another example is the ad hoc mobile agent based workflow system introduced by Meng et al. [Men00].

Miller et al. [Mil96] present five run-time architectures for implementing a Workflow Management System. They range from highly centralized to fully distributed architectures. The work also discusses the advantages and disadvantages of these architectures and the suitability of CORBA as a communication infrastructure.

2.3.2 Ad Hoc Modification to Process Model

Most workflow management systems currently on the market are not suitable for dealing with the dynamic nature of present-day business environments. Their use of static process models cannot react to and handle instantaneous changes in policies, constraints, market conditions, and the like. A workflow management system needs to be able to modify a process model at run-time to adapt to dynamic business conditions and exceptional situations [Han98]. While evolutionary changes to process models can be easily handled, ad hoc changes are much more challenging. Ad hoc changes are changes to the execution course of a specific workflow instance at run-time. They require the support of a dynamic workflow engine. Most research efforts in this area deal with the dynamic changes of process models used in transitional workflow systems. Ellis et al. [Ell95] introduce an approach to provide dynamic changes to a process structure. The changes are conducted by replacing a given sub-workflow by another completely specified sub-workflow. Petri-net formalism is used to analyze structural changes. Reichert and Dadam [Rei98] present a formal foundation for supporting dynamic structural changes of running workflow instances. Based upon a formal workflow model

(ADEPT), a complete and minimal set of change operations (ADEPT_flex) is defined to support users in modifying the structure of a running workflow, while maintaining its (structural) correctness and consistency. The modification operations include inserting tasks as well as task blocks into a workflow graph, deleting tasks, skipping tasks, serializing tasks that were previously allowed to run in parallel, and dynamically iterating and dynamically rolling-back a workflow. Muller and Rahm [Mul99] describe a rule-based approach for the detection of semantic exceptions and for dynamic workflow modifications, with a focus on medical workflow scenarios. Rules are used to detect semantic exceptions and to decide which activities have to be dropped or added. Different from the works mentioned above, we address run-time modifications to inter-organizational process models.

2.4 Virtual Enterprise and Workflow Technology

Recently, the use of the workflow technology to manage e-businesses and virtual enterprises has drawn much attention from the academic community. Several research projects attempt to tackle workflow management problems in these new application areas. Here we describe two representative projects.

WISE (Workflow-based Internet SErvices) is a project at the Swiss Federal Institute of Technology [Alo99; Laz01]. It aims to design, build, and test a commercially viable infrastructure for developing distributed applications over the Internet. The infrastructure provides an Internet-based workflow engine acting as the underlying distributed operating system for controlling the execution of distributed applications, and a process modeling tool for defining and monitoring the process. However, the workflow system they provide does not offer the needed facilities to capture and manage the dynamic properties of business processes.

CrossFlow is a European research project for supporting the cross-organizational workflow management of virtual enterprises [Gre99]. Its goal is to develop and implement a mechanism for connecting WfMS and other WfMS-like systems from different organizations in cross-organizational workflows and electronic commerce settings. CrossFlow provides a service-oriented model for cross-organizational workflows. In this service-oriented model, a service specifies which part of the workflow it fulfills and can span multiple activities. The service provider of each service can be either an internal resource (internal service) or an external organization (external service). For an external service, service selection at run-time will be based on the QoS parameters given in service specifications [Kli98]. A flexible change-control mechanism is also introduced in CrossFlow to react to potential problems during a workflow execution [Cro00].

Our inter-organizational workflow system integrates e-services provided by the participating organizations in a virtual enterprise. Unlike the service definition in CrossFlow [Gre99], the e-services in our inter-organizational workflow system are defined and provided independent of business process models. By introducing events, rules, and constraint-based e-service requests, our process model can capture more dynamic properties than that of the other workflow systems described in Chapter 2. The constraint-based dynamic service binding, event and rule mechanism, and run-time modification to process models described in this document provide a comprehensive solution to inter-organizational workflow management.

2.5 Web Service (E-Service) Related Technologies

The World Wide Web (WWW) has achieved a great success as an easy way to access information. Web services, the next step in the evolution of the WWW, allow

programmable elements to be placed on web sites where others can access the distributed behaviors captured by them. As communications protocols (e.g., HTTP) and message formats (e.g., XML) are standardized in the Web community, it becomes increasingly possible and important to have a standardized way to describe, manage, and access Web services.

Simple Object Access Protocol (SOAP) [Box00] is an XML/HTTP-based protocol for accessing services, objects, and servers in a platform-independent manner. It provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment. Web Service Description Language (WSDL) [Chr01] defines an XML grammar for describing network services as a collection of communication endpoints capable of exchanging messages. In WSDL service definitions, the operations and messages are described abstractly, then bound to a concrete network protocol and message format to define an endpoint. Both SOAP and WSDL have been submitted to the Wold Wide Web Consortium as proposals for the W3C XML activity on XML Protocols.

The Universal Description, Discovery and Integration (UDDI) [Ari00] is the first truly cross-industry effort driven by all the major platform and software providers, as well as marketplace operators and e-business leaders. The UDDI project has created a platform-independent, open framework for describing services, discovering businesses, and integrating business services using the Internet. It has also created an operational registry that is available today. The UDDI takes advantage of W3C and Internet Engineering Task Force (IETF) standards such as XML, HTTP and Domain Name System (DNS) protocols. Additionally, cross platform programming features are addressed by adopting early versions of the proposed SOAP. The UDDI protocol is the

building block that will enable businesses to quickly, easily, and dynamically find and transact with one another using their preferred applications.

While web services (also known as e-services) are becoming the programmatic backbone for electronic commerce, an XML language for the description of Web services compositions, the Web Services Flow Language (WSFL [Ley01]), is under development by IBM. WSFL supports two types of composition and choreography. The first type, the flow models, specifies the appropriate usage pattern for a collection of Web services in such a way that the resulting composition describes how to achieve a particular business goal; typically, the result is a description of a business process. The second type, the global models, describes how the parts of a set of Web services interact with each other. Thus, WSFL builds a framework in which service providers and consumers can come together to implement standard business processes.

Since a business process model defined by our DWM can be used by a business organization to define a new e-service in a similar way as IBM's WSFL, our model is also a tool to compose e-services. Moreover, in addition to e-service requests, our model provides additional constructs for capturing the dynamic properties of a business process, i.e. events, rules, and constraints associated with e-services and e-service requests.

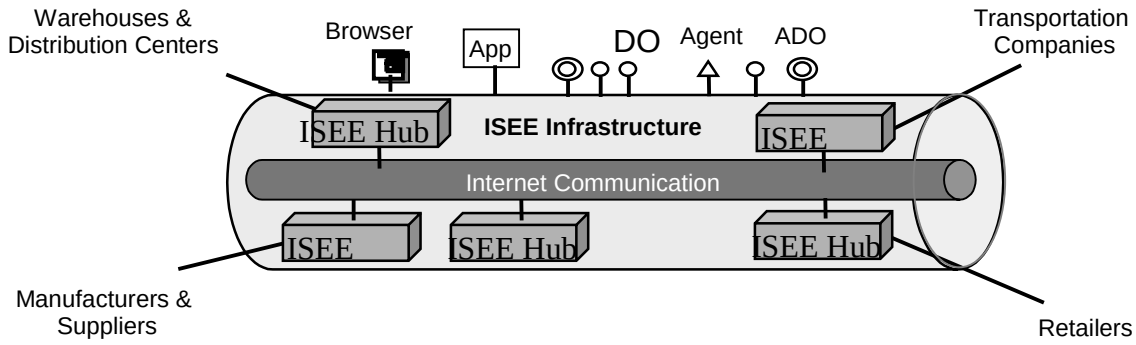
In our Dynamic Workflow Model (DWM), e-service requests specified in an activity are defined according standardized e-service templates, whose format is consistent with the format of web services defined in WSDL. We assume that e-services are maintained by an UDDI-enabled Broker Server, and SOAP messages are used to invoke requested e-services during the execution of workflow instances.

CHAPTER 3
ARCHITECTURE OF DYNAFLOW

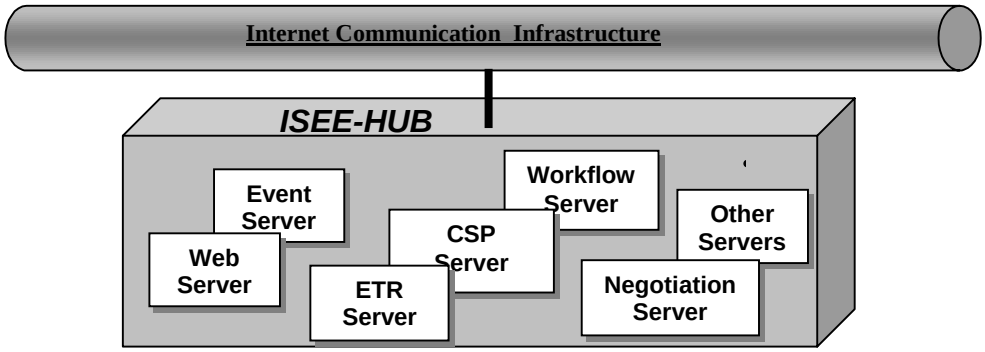
A research project, supported by the National Science Foundation and conducted at the Database Systems Research and Development Center of the University of Florida, is entitled "Research on Advanced Technologies to Support Internet-based Scalable E-business Enterprises (ISEE)." It aims to build an advanced information infrastructure to support collaborative e-business and other distributed applications [Su01b]. The ISEE information infrastructure provides a number of servers that complement the services provided by existing web servers. Our work on dynamic workflow management systems makes use of a number of these servers in its implementation. This chapter begins with an overview of the ISEE information infrastructure. Then the global architecture of the dynamic workflow management system DynaFlow is described and the interaction among the servers that constitute DynaFlow is presented. At the end of this chapter, we give a brief overview on how DynaFlow works.

3.1 Overview of ISEE Infrastructure

The ISEE infrastructure is formed by a network of ISEE Hubs, as shown in Figure 3-1(A), each of which has a number of replicable ISEE servers providing various services to individuals and business companies. Its relationship with the existing application systems, distributed objects, agents, active distributed objects [Lee00, Lee01], and web browsers, is illustrated in Figure 3-1(A).



(A)



(B)

Figure 3-1: ISEE infrastructure. (A) Overall architecture. (B) An ISEE hub.

At present, each ISEE Hub has a Web Server, an Event Server, an Event-Trigger-Rule (ETR) Server, a Workflow Server, an automated Negotiation Server, and a Constraint Satisfaction Processing (CSP) Server as shown in Figure 3-1(B). These servers are middlewares that provide ISEE-services useful for collaborative e-business applications. For example, the Event Server enables flexible and dynamic communication among loosely coupled systems that are distributed across organizations. The Event-Trigger-Rule Server provides timely and automated responses to events by invoking triggers and rules. Another important server provided by the ISEE

infrastructure is the Dynamic Workflow Server. DynaFlow, which makes use of this server to provide dynamic workflow management, will be described in the next section.

ISEE Hubs are installed at the sites of ISEE organizations. Users of an ISEE Hub not only have full access to Internet and Web services, but also to the additional services provided by ISEE servers.

3.2 Global Architecture of DynaFlow

The architecture of DynaFlow is shown in Figure 3-2. The system consists of a Workflow Server, an Event Server, and an ETR Server. A centralized Broker Server is also used in DynaFlow to manage the e-service specifications of an e-service domain that have been registered by participating business organizations and to match e-service requests against these specifications for selecting the suitable e-service provider(s). A Broker Proxy is installed in each ISEE Hub to communicate with the centralized Broker Server. Multiple Broker Servers can be installed to handle multiple e-business domains.

The Workflow Server is the key component of DynaFlow. It is composed of two sub-components: namely, the Process Definition Tool and the Workflow Engine. The Process Definition Tool is used to design inter-organizational process models, which are modeled by a dynamic workflow model (DWM). The Workflow Engine schedules the execution of the workflow instances according to the process model specification. During the execution, the Workflow Engine makes use of the ISEE services provided by other servers, such as the Event Server, the ETR Server, and the Broker Server to achieve the dynamic properties (i.e., the active, flexible, and adaptive properties) of the workflow management system.

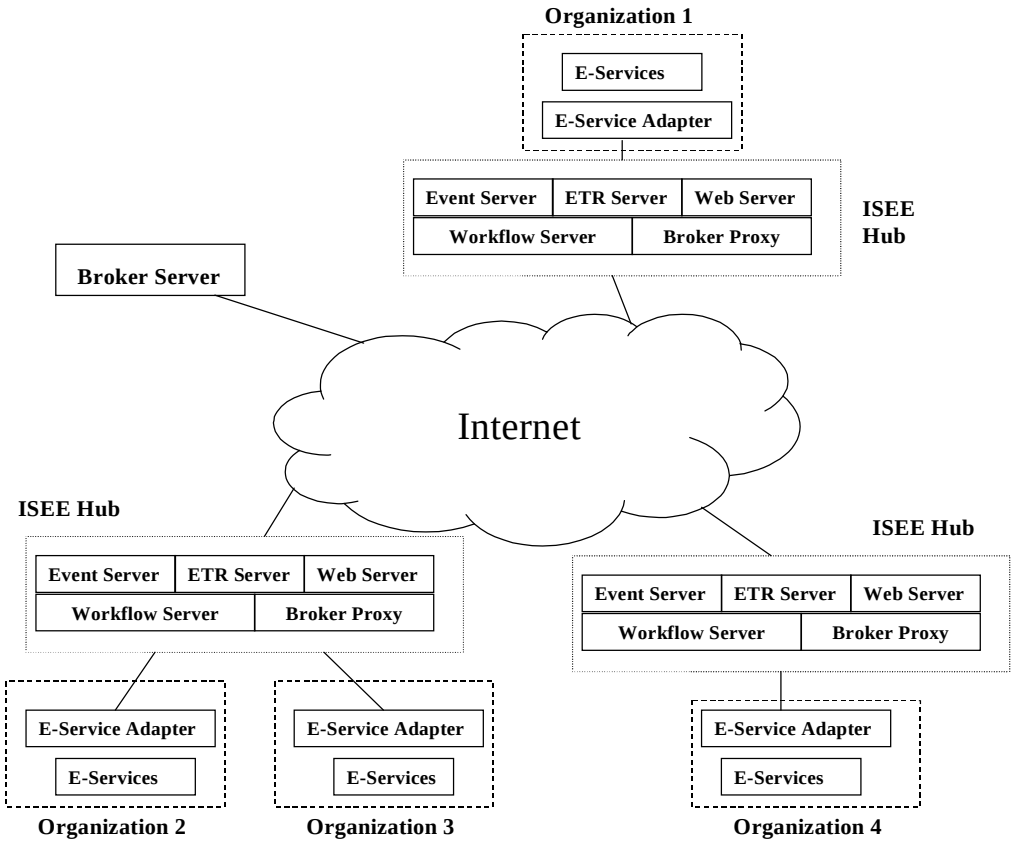


Figure 3-2: Global architecture of DynaFlow.

In an ISEE virtual enterprise, inter-organizational process models are designed to capture the business processes of the entire virtual enterprise. The participating organizations of a virtual enterprise may have their own process models that are processed by their own workflow systems. These “local” process models can be enacted by the dynamic workflow management system as a part of an inter-organizational process model. The inter-organizational process models, once designed, are made accessible to all the participating organizations. The models can be stored in a central repository and can be checked out and customized by participating organizations to meet their local needs. However, for scalability reasons, we replicate these inter-organizational process models, as well as the Workflow Server, at all the ISEE-hub sites. Authorized users of

the virtual enterprise, who may travel from one collaborative site to another, can enact a process model at any site. The workflow instance created by the enactment will then be managed by an instance of the Workflow Server at that site. Thus, concurrent workflow instances initiated at different sites are controlled and managed by multiple instances of the Workflow Servers.

Participating business organizations can perform and contribute different manual or automated tasks, which is useful for the operation of a joint business. These tasks are to be invoked in the execution of an inter-organizational business process. In our work, we uniformly treat all the sharable tasks performed by people or automated systems as electronic services or e-services. An e-service adapter needs to be installed at each organization's site as a wrapper for its e-services so that they can be invoked during the execution of an inter-organizational business process.

3.3 Servers in DynaFlow

The relationship between the servers of DynaFlow is shown in Figure 3-3. In this figure, the dashed lines represent build-time actions, and the solid lines represent run-time actions.

3.3.1 Process Definition Tool

The Process Definition Tool is a Graphic User Interface (GUI) used for designing process models. It invokes a GUI that has been implemented for a Knowledge Profile Manager [Lee00, Lee01] to define business events, business rules, and triggers that are related to process models. It also invokes a Constraint Definition Tool (For clarity, it is not shown in the figure) to define constraints that are associated with the e-service requests specified in the process models. The definitions of events generated by the

process models are passed on to both the ETR Server and the Event Server for installations of the events. The run-time workflow structures and activity codes (both will be explained in detail in Chapter 5) are generated from the meta-information of the process models and are used by the Workflow Engine to schedule the execution of workflow instances.

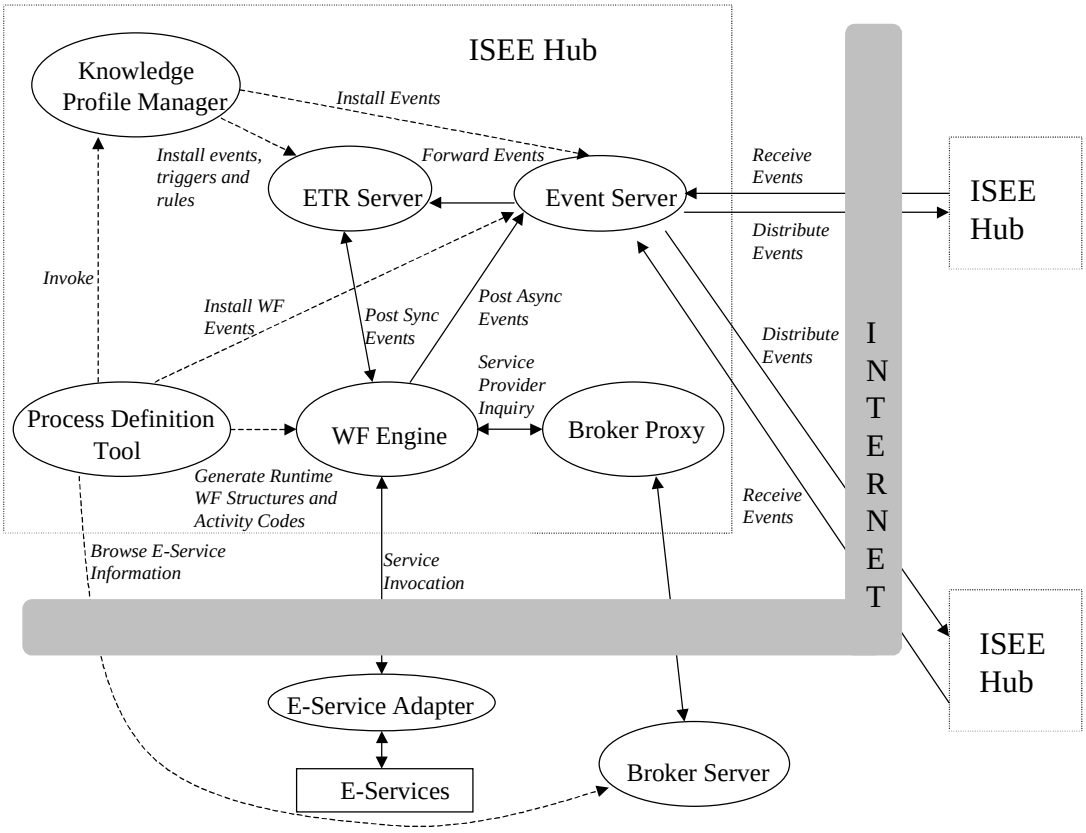


Figure 3-3: Architectural components of DynaFlow.

3.3.2 Knowledge Profile Manager

In DynaFlow, the Knowledge Profile Manager is responsible for defining the events, triggers, and rules related to business process models. The definitions of events are given to both the ETR Server and the Event Server for storage and use. The definitions of rules and triggers are given to and managed by the ETR Server.

3.3.3 Workflow Engine

The Workflow Engine schedules the execution of a workflow instance according to the run-time workflow structures and activity codes generated from the meta-information of the process model. It is connected to the Event Server in order to post asynchronous events to it. The Event Server performs event notifications for the subscribers to these events. The Workflow Engine also posts synchronous events to directly the ETR Server to trigger those rules that are linked to these events. During the execution of the workflow instance, the Workflow Engine would contact the Broker Server to get the service provider information for those e-services that are specified in the activities of a process model. After getting the information about the proper service provider for an e-service request, the workflow engine generates a SOAP message for the e-service request, and sends it to the e-service adapter at the service provider's site to invoke the e-service. A SOAP message containing the e-service result will be returned to the Workflow Engine after the e-service invocation is completed. The Workflow Engine also provides APIs to modify the process model at run-time.

3.3.4 Event Server

The Event Server handles the incoming and outgoing events to and from an ISEE Hub. It is connected to a local copy of the Workflow Engine (to receive the asynchronous events issued by it) and to the remote Event Servers at the other ISEE Hubs (to notify them the occurrences of events). It provides an event registration facility for the participants of the virtual enterprise to register their interest by subscribing to certain events. The ETR Server is a default subscriber to the local Event Server. When the Event Server receives an event from a remote event server, it forwards it to the local ETR

Server to initiate the processing of triggers and rules that are associated with the remote event.

3.3.5 ETR Server

The ETR Server processes the triggers and rules in an ISEE Hub. The trigger and rule definitions that are input through the Knowledge Profile Manager are provided to the ETR server and are transformed to internal data structures used for executing the triggers and rules. The ETR server receives events from the Event Server (for asynchronous workflow events) or directly from the Workflow Engine (for synchronous workflow events), and performs the trigger and rule processing. The relationships between events and rules are specified by triggers. On receiving an event, the ETR Server can immediately identify the trigger related to the event and schedule the structure of rules specified in the trigger for processing.

3.3.6 Broker Server

The Broker Server provides several functions to DynaFlow. It provides facilities for the registration and publication of e-services provided by e-service providers and manages the registered e-service information. It also provides facilities for e-service requesters and process model designers to browse and access the registered e-services information. The constraint-based selection of service providers is another important function of the Broker Server: It makes use of a constraint satisfaction processing (CSP) server to match e-service requests with the e-service specifications of different service providers to identify the proper providers.

3.3.7 Broker Proxy

The Broker Proxy in each ISEE-hub is the local proxy of the remote Broker Server. The Workflow Engine in the ISEE-hub contacts the Broker Server through the

Broker Proxy. The functions that the Broker Server provides to the Workflow Engine include dynamic service binding of e-service requests to e-services and their providers and the processing of inquiries for service provider information and service templates.

3.3.8 E-Service Adapter

E-Service Adapters are installed at participating organizations' sites to wrap services, which could be implemented in different ways. These services can then be invoked using SOAP messages transmitted by the HTTP protocol according to the format of the corresponding e-service templates. An E-Service Adapter thus hides the heterogeneity of service implementations and presents a uniform view of these services as e-services.

3.4 Functions of DynaFlow

Except the Broker Server and the e-service adapters, the architectural components described above are replicated in all the ISEE-hubs over the Internet, making the ISEE infrastructure symmetrical and the architecture a peer-to-peer, multi-server architecture. The execution of a workflow instance is handled by the Workflow Engine of the ISEE Hub at the site where the workflow instance is initiated. Other ISEE Hubs' responsibilities are to receive the asynchronous events posted by a workflow instance and deliver them to their corresponding ETR Servers to trigger the rules defined by the subscribers of these events.

The activities of the participants in DynaFlow are divided into two phases: build-time and run-time.

3.4.1 Build-time Activities

1. Process model designers use the Process Definition Tool to design inter-organizational process models based on DWM.

2. When specifying an e-service request, a designer can invoke Constraint Definition GUI to define constraints for the e-service request.

3. The run-time workflow structures and activity code are generated based on the meta-information of the process models.

4. Organizations that want to initiate workflow instances of a process model can invoke the GUIs of the Knowledge Profile Manager through the Process Definition Tool. These GUIs are used to define events, triggers and rules, which specify some local business events, policies, constraints, and/or regulations.

5. The event definitions are installed in both the ETR Server and the Event Server. The rule and trigger definitions are installed in the ETR Server.

6. Organizations that are interested in the progress of an enactment of a business process can subscribe to asynchronous events posted by the workflow instance, and they can define rules and tie these rules to the events by trigger specifications.

3.4.2 Run-time Activities

1. A participating organization, as the workflow initiator, initiates a workflow instance of a defined process model at an ISEE Hub site.

2. The Workflow Engine schedules the execution of the instantiated workflow instance according to the run-time workflow structure and activity code.

3. During its execution, the Workflow Engine may contact the Broker Server through the Broker Proxy to get the proper service providers for the e-service requests specified in the process model.

4. To invoke an e-service, a SOAP message containing the request data is generated and sent to the e-service adapter at the selected service provider's site to invoke the corresponding e-service. After the invocation, a SOAP message containing the result data is sent back to the Workflow Engine.

5. Synchronous events are posted directly to the local ETR Server to trigger business rules to enforce the local business constraints and policies, and/or to dynamically alter the process model.

6. Asynchronous workflow events are posted to the local Event Server.

7. The local Event Server forwards the asynchronous events to the Event Server of the ISEE Hubs of their subscribers, causing the firing of subscribers' rules. The events are also delivered to the local ETR Server to trigger the local business rules.

CHAPTER 4
DYNAMIC WORKFLOW MODEL

In this chapter, we introduce a dynamic workflow model (DWM) for modeling business processes. Since process models defined in a DWM may invoke manual and automated tasks that can be carried out by different organizations, we shall first present a uniform way of specifying these two general types of tasks uniformly as e-services. We then present DWM as an extension of the underlying model of WfMC's WPDL. After introducing DWM, we shall delineate the dynamic properties that a workflow management system can have if it is built upon DWM and give the specifications of the modeling constructs. We then give a sample process scenario defined in DWM at the end of this chapter.

4.1 E-Services

In our work, we regard e-services as services offered on the Internet that can be accessed programmatically using a standard Internet Protocol and representation formats, such as HTTP and XML. In other words, E-services can be accessed in a uniform way, irrespective to the types of systems and the programming languages used to implement these services [Hel02].

In a virtual enterprise, participating business organizations can perform and contribute different manual or automated tasks that can be specified uniformly as e-services. A participating organization can provide multiple e-services, and an e-service can be provided by multiple organizations. In the Internet environment, providers of e-

services may change frequently; new providers are added and old providers become unavailable. In modeling business processes, it is therefore important to separate e-service requests specified in a process model from their providers. That is, a process model should not statically bind its e-service requests to specific providers at the time a process model is defined, except for rare and special cases. Instead, the binding should occur at run-time when the available providers are known to the workflow management system.

In the following subsections, we will explain how we specify e-services and e-service requests. We also describe how we dynamically bind e-service requests to e-services.

4.1.1 E-Service Template

In order to introduce a standard way for defining e-service in a specific business domain, it is useful to categorize e-services and their providers by the types of business that the providers conduct. The e-service categorization and specification in our work are patterned after the Universal Description, Discovery and Integration (UDDI) specification [Ari00] and the Web Service Description Language (WSDL) [Chr01]. For example, a business organization may be of business type *Distributor* in a supply chain domain. For each business type, a set of useful e-services can be defined. Business organizations that are of the same business type may provide all or some of these e-services. To standardize the specification of an e-service, an *e-service template* can be jointly defined by those business organizations of the same business type. An e-service template consists of one or more operations offered by the e-service. For each operation, there are three general types of attributes:

- Input attributes, which specify the data needed as input to invoke an e-service operation.
- Output attributes, which specify the returned data of an e-service operation.
- Service attributes, which specify some properties of the operation, such as the length of time the operation takes, the side effects of the operation, and the like.

An e-service template can also contain service attributes for the e-service. These attributes specify the properties of the e-service, such as the cost for using the e-service, the quality of the e-service, and so on, which may be useful for service negotiation, contracting, service selection, and service level management. An example of an e-service template for the e-service *OrderProcessing* provided by the business type *Distributor* is shown in Table 4-1. This e-service provides one operation: *Process Order*.

Table 4-1: E-Service template of e-service *OrderProcessing* of *Distributor.*

			Name	Type
Operations	Process Order	Input Attributes	Product_Name Model_Name Quantity User_Info	String String Int UserInfo
		Output Attributes	Status	Status
		Service Attributes	Duration Shipping_Method	Time String
	Service Attributes (e-service)		Cost	Double

All e-service templates are managed by an UDDI-enabled Broker Server [Hel01; Hel02]. They are used by service providers to register their e-services and by process model designers to specify their e-service requests in a process model.

4.1.2 E-Service Constraint

A service provider registers its e-services with the Broker Server by first browsing and selecting the proper e-service templates, which are displayed as a form to be filled by the service provider. During the registration process, the service provider first provides

the broker with its general information, such as its name, URL, telephone, email, etc. It then specifies which e-services it provides. For each e-service, the service provider needs to specify the e-service binding description, which contains the location of the service implementation and details on the protocol and the port to be used to access the server that hosts the e-service. The service provider can also specify some attribute and inter-attribute constraints on the service attributes of the e-service, and the constraints on the input attributes and service attributes of the operations. Attribute constraints are used to specify the values that the individual input and service attributes can have, and inter-attribute constraints are used to specify the interrelationship between the values of these attributes. These constraints restrict the kind of data that the requester of an e-service can provide when the e-service is invoked. By allowing constraints associated with e-service attributes to be explicitly specified, we can extend the Web Service Declaration Language (WSDL) to increase its expressive power. For constraint specifications, we adopt the syntax and semantics of the Constraint-Based Requirement Specification Language used in a previous project [Hua00]. We shall call these constraints *e-service constraints*.

For example, a distributor *Worldwide* which provides the e-service named *OrderProcessing* may specify constraints on the operation Process Order, as given in Figure 4-1. They state that the operation *Process Order* of the e-service *OrderProcessing* can only process the order of computer products with the quantity less than 1000, and if the quantity of the order is larger than 500, this e-service needs to take more than 10 time units. Iac1 is the name of the inter-attribute constraint.

For each e-service, the e-service attributes in the e-service template and the e-service constraints defined by the service provider together form the e-service specification. After registration, the general information of service providers and their e-service specifications are managed by the Broker Server.

ATTRIBUTE_CONSTRAINT:

Product_Name	String	ENUMERATION [“Computer”]	priority[1]
Model_Name	String	ANY	priority[2]
Quantity	Integer	RANGE [1-1000]	priority[3]

INTER_ATTRIBUTE_CONSTRAINT:

Iac1 Quantity > 500 **implies** Duration>10

Figure 4-1: Constraint specification for operation *Process Order*.

4.1.3 E-Service Request Constraint

During process modeling, an e-service request specified in an activity of a process model is defined in terms of the attributes given in the corresponding e-service template. To define an e-service request in the activity, in addition to the variable mappings of the input and output attributes, which map the variables in this activity to the input/output attributes of the requested e-service operation, the constraints on the service attributes can also be specified. We shall call these constraints, e-service request constraints. We will give the full e-service request specification when we define the activities of a process model in Section 4.3.2.

An example of an e-service request constraint for the operation Process Order of the e-service OrderProcessing is shown in Figure 4-2. It states that the requestor expects that the Process Order operation should not take more than 10 time units, the cost of the e-service should not be more than $1,000, and if it takes more than 4 time units, then the cost must be less than $800.

```
ATTRIBUTE_CONSTRAINT:
    Duration      int       RANGE [0 .. 10]          priority[1]
    Cost          float     RANGE [0 .. 1000]        priority[2]

INTER_ATTRIBUTE_CONSTRAINT
    iac1:   Duration >4   implies   Cost < 800
```

Figure 4-2: A sample specification of an e-service request constraint.

In summary, at build-time, the service providers register their e-services with the Broker Server according to these e-services' corresponding templates. The e-service specifications are maintained at the Broker Server site. The process model designers define e-service requests according to the same corresponding e-service templates. The build-time relationship among the Broker Server, the service providers, and the process definition tool is shown in Figure 4-3.

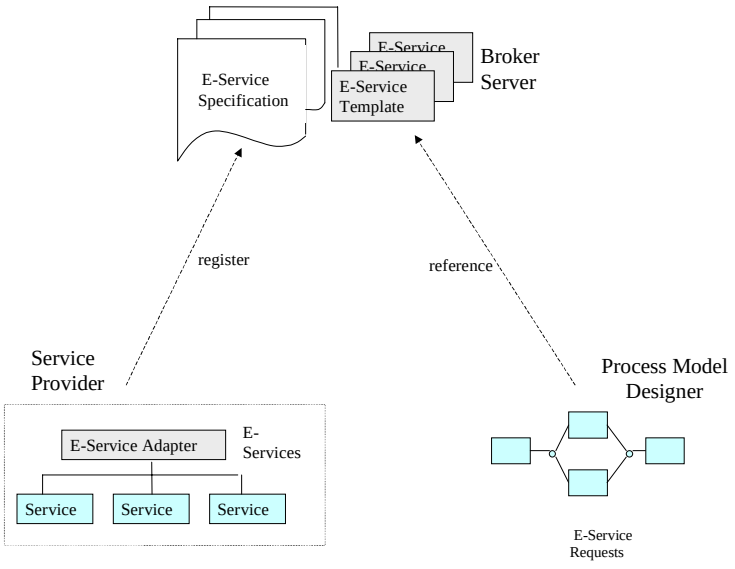


Figure 4-3: Build-time relationship among the Service Provider, the Broker Server, and the Process Definition Tool.

4.1.4 Constraint-based Brokering

An important function of the Broker Server is to do constraint-based brokering and service provider selection. To achieve this, the Broker Server would match e-service

requests with the e-service specifications given by service providers to identify the proper service providers for the request. The data provided for the input attributes of the requested e-services and the constraints specified in these requests will have to be compatible with (i.e., not conflict with) the attribute constraints and inter-attribute constraints specified by a service provider. The constraint matching is accomplished by calling on a Constraint Satisfaction Processor (CSP) [Hua00; Su01a].

In a matching operation, there are three possible results. First, the Broker Server cannot find a service provider that can provide e-services that satisfy the constraints and input data given in the e-service requests. In this case, the matching operation has failed. Second, there is a single service provider, which provides e-services that satisfy all the requirements of the e-service requests or match the requirements within an acceptable threshold. In this case, the matching operation succeeds and the e-service of the provider is used to service the request. In the third case, multiple service providers can satisfy the requests. In this case, a Cost Benefit Evaluation Server [Hua00; Su01a] is used to evaluate and rank the e-services provided by these service providers and the best provider is selected.

The constraint-based service provider selection is an important function that the Broker Server provides to the Workflow Engine. In our dynamic workflow management system, this function is used by the Workflow Engine to do dynamic service binding. We will explain the dynamic service binding technique in Section 4.4, in which the dynamic properties of DWM are discussed.

4.1.5 E-Service Invocation

When a workflow instance needs to access an e-service to submit its e-service request, the Workflow Engine first gets the URL address of a selected service provider

from the Broker Server. It then wraps the e-service request in a SOAP message and sends it through HTTP to the e-service adapter at the selected service provider's site. The E-Service Adapter parses the SOAP message to determine the operation of the e-service to be invoked [Kri01]. The E-service Adapter then determines the corresponding method that implements the service, builds the parameter list as required by the method, and invokes the method. After the invocation is completed, the E-Service Adapter encapsulates the return data into a SOAP message and returns it to the Workflow Engine at the requestor's site. Figure 4-4 shows the run-time e-service invocation mechanism in DynaFlow. Sample SOAP messages that are used to invoke the operation *Process Order* of the e-service *OderProcessing* are shown in Figure 4-5. Figure 4-5(A) shows the SOAP message containing the input data of the operation *Process Order*. The SOAP message containing the result of this e-service invocation is shown in Figure 4-5 (B).

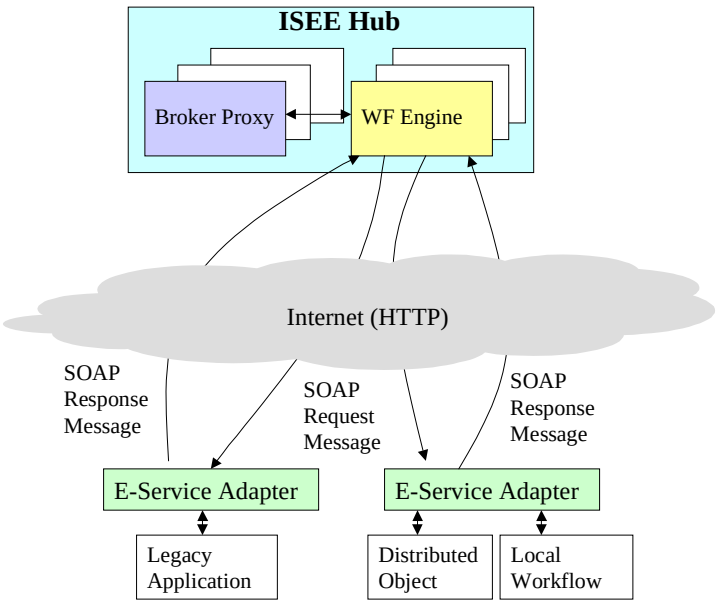


Figure 4-4: Run-time e-service invocation mechanism.

```
<SOAP-ENV:Envelope
      xmlns="www.eservices.ufl.edu/services/OrderProcessing"
      xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
     SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

 <SOAP-ENV:Header>
     <serviceuri>
          www.eservices.ufl.edu/services/Distributor/OrderProcessing
     </serviceuri>
 </SOAP-ENV:Header>

  <SOAP-ENV:Body>
    <!-- element holding remote call param info -->
   <ProcessOrder>
       <productName> Omnibook </productname>
       <modelName>    6000  </model>
       <quantity>    5  </quantity>
       <userInfo>
            <username> DB Center </username>
            <maillingAddr> CSE470, University of Florida, Gainesville, FL
            </maillingAddr>
        </userInfo>
    <ProcessOrder>
   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

(A)

```
<SOAP-ENV:Envelope  xmlns="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

   <SOAP-ENV:Header>
     <serviceuri>
          www.eservices.ufl.edu/services/Distributor/OrderProcessing
     </serviceuri>
   </SOAP-ENV:Header>

   <SOAP-ENV:Body>
       <!-- element holding operation result -->
       <ProcessPO_result>
             <OrderStatus>  order shipped   </OrderStatus>
       </ProcessPO_result>
   </SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```

(B)

Figure 4-5: SOAP messages for invocation of operation *ProcessOrder* in e-service *OrderProcessing*. (A) SOAP request message. (B) SOAP response message.

4.2 Dynamic Workflow Model

As described in Chapter 2, WfMC's WPDL defines a well-accepted standard of workflow meta-model that can be used to enable workflow vendors to exchange workflow models. In WPDL, a process definition is a network of transition edges that connect activity nodes. The edges are optionally labeled by transition conditions. Figure 4-6 shows a business process model that conforms to the WPDL standard.

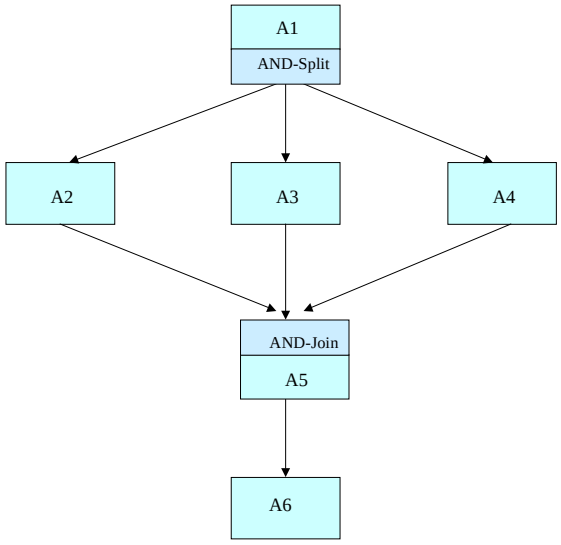


Figure 4-6: WPDL business process model.

However, there are some factors that limit WPDL's ability to support the dynamic inter-organizational workflow required in a virtual enterprise. In WPDL, the specifications of join and split constructs and their constraints (AND, OR, or XOR) are embedded in the specifications of activities. These constructs and their constraints define the structural relationships and constraints among activities. Since they are part of the activity specifications, changes made to them would entail changes to activity specifications. Also in a process model defined in WPDL, all activities can make reference to all the "workflow reference data" like global variables in programming

languages. The data flows among activities are not explicitly defined. This makes the data relationship among activities unclear. Furthermore, a process model defined in WPDL is static. WPDL does not provide any mechanism for run-time modifications of a process model, as discussed before. Finally, since WPDL is defined for traditional workflows, it does not support the dynamic binding of e-service requests, which is required for inter-organization process modeling and execution.

In order to adapt to the changing nature of e-business, we made the following extensions and modifications to the underlying workflow model of WPDL. These extensions and modifications not only make it easier to modify a process model both at definition-time (or build-time) and run-time, but also enable it to support inter-organizational business processes.

(1) Introduction of *Connector*. We extract the specification of Join and Split constructs and their constraints (i.e., OR, AND, and XOR) from the activity definition and introduce a new modeling construct called Connector to specify the above extracted aggregation properties. By separating activity specifications from the specifications of all types of control information, any change made in one will not affect the other. As we shall explain later, the separation also facilitates the run-time modification of the process model itself.

(2) Encapsulation of the activity definition. We extend the WPDL's activity definition by adding the specification of input parameters and output parameters. An activity can only reference the data passed by the input parameters. It exposes the result of operations (or tasks) specified in the activity only through the output parameters. By doing this, an activity definition is encapsulated. The activity definitions and the code

generated for them become reusable. In addition, a modification to the task items inside an activity can be made without affecting other part of the process model.

(3) Inclusion of e-service requests in activity definitions. In a process model defined for a single organization, an activity specification can include manual or automated tasks to be performed within the organization. In a process model defined for an inter-organizational workflow, the activity specification should include e-service request(s) that can be serviced by different business organizations. For this reason, we include e-service requests in the activity definition in addition to using direct invocations of manual or automated tasks.

(4) Inclusion of explicit data flow specification. In a process model defined in WPDL, there is no explicit specification of data flow between activities. Data transfer between activities is through a common data area, which contains the so-called workflow relevant data. Any activity can modify and access the workflow relevant data. In our model, we define data flows explicitly. We use inter-activity parameter mappings to define the data flows between activities.

(5) Introduction of event trigger, and rule. Another important extension we make to WPDL is the introduction of events, rules, and triggers in a process model specification. The activities inside the process model can post events. We distinguish the following three types of events:

- *Before-Activity-Event*: Before an activity is executed, the Workflow Engine that oversees the processing of a workflow instance would post a Before-Activity-Event.
- *After-Activity-Event*: After the processing of an activity, the Workflow Engine would post an After-Activity-Event.

- *External events*: An activity can also explicitly post events to report data conditions or exceptions to the subscribers of the events. Posting an external event is regarded as an operation or task item in an activity.

In the remaining part of this dissertation, we refer to Before-Activity-Event and After-Activity-Event as workflow events because they are treated as an integral part of a process model. Different from external events posted inside activities, workflow events are automatically generated. We will describe the definition of workflow events and their generation in detail in the next section.

By introducing these events, the execution of a workflow instance would post events to automatically trigger the processing of some business rules. These rules have the format of Condition-Action-AlternativeAction. They may simply perform operations in addition to the task items (including e-service requests) specified in activities to enforce some business policies, regulations, or constraints. They may also modify the execution flow of the workflow instance (e.g., skip the next activity or branch to a specific activity in a process model). Different organizations may "attach" different sets of business rules to the events of a process model when they enact the model. Thus, the workflow instances initiated by different organizations will trigger a different set of rules. In this way, a process model can be tailored to fit individual organizations' local needs. Rules can also be dynamically attached to the events posted by a running workflow instance to handle situations that were not foreseen when the instance was initiated. Asynchronous events can be used as notifications of the milestones of the enacted business process.

By extending WPDL in the ways described above, we construct our dynamic workflow model (DWM). In DWM, the modeling constructs used to define a process model include activity, transition, connector, subflow, block, data flow, event, rule, and

trigger. In addition to normal activities, two special purpose activities are introduced: the Begin-Activity and the End-Activity. The Begin-Activity and the End-Activity are used to define the entry and exit points of a process model or a block in a process model, respectively. A block is a connected network of transitions, activities, subflows, and connectors. It is connected to the rest of a process model only via the Begin-Activity and the End-Activity of this block. A block can be a loop-block, which defines a set of activities that can be iterated until an exit condition is met. For each process model or block, only one Begin-Activity and one End-Activity can be defined.

The graphic representation of a business process model is shown in Figure 4-7. The nodes in the graph can be activities, connectors, or subflows. The oval nodes represent the Begin-Activity or the End-Activity, the rectangle nodes represent activities, the hexagram node represents a subflow, the rounded rectangle node represents a block, and the small circle nodes represent connectors. The solid edges represent conditional transitions between activities, connectors, and subflows. The connectors and transitions together specify the control flow. The data dependencies among the activities and subflows are captured by data flows. The data flows can be either implicitly defined together with the transitions, or they can be explicitly defined. A thick solid line between activities/subflows ending in a diamond shape in the graph represents an explicitly defined data flow. The small ovals inside activities represent the events posted by the activities. The three types of events that can be posted by activities are Before-Activity-Event (BE), After-Activity-Event (AE), and External Event (EE). Business rules can be attached to these events by using trigger specifications (represented by dashed lines).

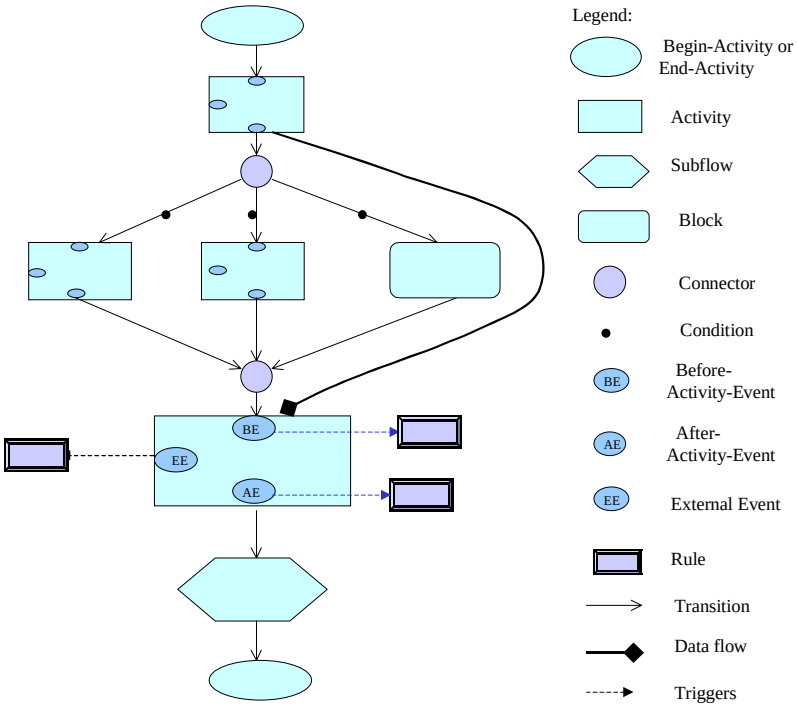


Figure 4-7: Business process model in DWM.

4.3 Workflow Event Definition

In business process modeling, a process model designer can specify whether an activity posts synchronous workflow events, asynchronous workflow events, or both. Event definitions are then automatically generated based on the name of the process model, the name of an activity, and the input and output data of the activity.

The workflow events are named after the corresponding process name, activity name, and workflow type. For example, the name of a synchronous Before-Activity-Event of activity "A1" in process "OrderProcessing" is "OrderProcessing _ Before_A1_SYNC."

The event attributes are used to deliver the data to the event subscribers. For synchronous events, return data from subscribers are expected. Table 4-2 shows the attributes of workflow events and their return data (only for synchronous Workflow

Events). The Before-Activity-Events pass the input data of the activity to the rules linked to them, and the After-Activity-Events deliver the output data of the activity to the subscribers.

In addition to the input parameters or the output parameters of the activity, the attributes of the generated asynchronous workflow events contain the identifier of the service provider, to whose e-services the e-service requests specified in the activity are bound. The identifier can be used to selectively trigger the rules defined by that service provider when it is selected to be the performer of the activity.

Table 4-2: The attributes of workflow events.

Workflow Event Type	Input Attributes	Return Attributes
Async, Before	Service Provider ID, Activity Input Parameters	None
Async, After	Service Provider ID, Activity Output Parameters	None
Sync, Before	Instance ID, Activity Input Parameters	Activity Input Parameters
Sync, After	Instance ID, Activity Output Parameters	Activity Output Parameters

For synchronous workflow events, we add the identifier of the workflow instance, which posts the events, and the identifier of the organization, which initiates the workflow instance, to the input attributes of the events. The instance identifier is used to trigger rules defined only for a specific workflow instance.

4.4 DWM Specification

4.4.1 Process Model Definition

A process model is defined in terms of the modeling constructs provided by the DWM. Descriptive attributes, such as author, creation time, process description, time

estimation, priority, classification, documentation, and so on, can be used to define a process model. The syntax of a process model definition is shown below:

PROCESS <process id>

[CREATED <creation date>]

[DESCRIPTION <description>]

[CLASSIFICATION <classification>]

[PRIORITY <priority>]

[<time estimation>]

[DOCUMENTATION <documentation>]

<activity list>

[<subflow list>]

[<connector list>]

<transition information list>

[<data flow list>]

[<event list>]

[<data class list>]

END_PROCESS

The optional clauses are surrounded by < and >. The clauses such as CREATED, DESCRIPTION, CLASSIFICATION, PRIORITY, and DOCUMENTATION are used to specify the descriptive attributes of a process model. In addition to these attributes, a process model definition also contains the definitions of entities that constitute the process model.

<Activity list> specifies a set of activities that is contained in a process model. Activities are used to define tasks that need to be done for the business process. As we described in the proceeding section, in addition to normal activities, two special-purpose activities, the Begin-Activity and the End-Activity, are introduced to define the entry point and the exit point of a process model or a block in a process model, respectively.

Existing process models can be referenced in a process model definition as sub-flows, which are specified in <subflow list>. The terms <connector list>, <transition information list>, <data flow list>, <event list>, and <data class list> are used to specify connectors, transitions, data flows, events, and data classes that are contained in a process model, respectively. The specifications of these different entities are introduced in subsequent sections.

4.4.2 Activity

We first introduce regular activities that are used to perform tasks in a business process. We then give the details of the two special activities: Begin-Activity and End-Activity.

4.4.2.1 Regular activity

Activities are the basic building blocks of a business process model. Some attributes need to be specified for each activity. Every activity must have an activity identifier. The input parameter and output parameter descriptions also need to be defined to encapsulate the activity. These input and output parameters can be used in a data flow definition. Other attributes may also be defined to specify a performer assignment, an activity description, and other run-time descriptions. The activity body defines the tasks that need to be done by an activity. It is an important part of an activity definition.

The syntax of an activity definition is as follows:

ACTIVITY <activity id>

[DESCRIPTION <description>]

[PERFORMER < participant assignment>]

IN_PARAMETER <input parameter list>

OUT_PARAMETER <output parameter list>

[WF_EVENTS] <workflow event list>

[ACTIVITY_VAR <variable list>]

IMPLEMENTATION <activity body>

END_ACTIVITY

The activity id is a unique identifier for the activity in the context of a process model. The DESCRIPTION clause contains a text string describing what the activity does. Each activity has a list of input parameters and a list of output parameters. They are defined in the IN_PARAMETER clause and the OUT_PARAMETER clause, respectively. The input parameters specify the input data that will be used in the activity body. The output parameters specify the data that will be produced by the activity. The WF_EVENT clause specifies the workflow events that this activity posts. The workflow events can be synchronous Before-Activity-Events, asynchronous Before-Activity-Events, synchronous After-Activity-Events, and/or asynchronous After-Activity-Events. These events are automatically generated.

The PERFORMER clause specifies the business type of the business organization whose e-services are requested in the activity body (e.g., toy retailer as a business type). The process model designer can also specify the performer selection constraint in the PERFORMER clause. The specification of the PERFORMER clause is shown below:

PERFORMER <business type name> (<performer selection constraint>)

There are four types of performer selection constraints that can be specified to restrict the selection of a proper performer (i.e., an e-service provider). They are explained below:

1. CONSTANT: The performer of the activity is a specific organization. For example, in the following PERFOMER clause, *Distributor* represents the business type of organizations that provide services as distributors and *IBM* is specified (i.e., statically bound) as the performer of the activity.

   PERFORMER *Distributor* (CONSTANT *IBM*)

2. ANY: The performer of the activity can be any suitable organization of the specified business type. In this case, the e-service requests in the corresponding activity body can be dynamically bound to a selected organization at run-time by the Workflow Engine. For the example given below, any organization of the *Transportation_Agency* business type can be the performer of this activity.

   PERFORMER *Transportation_Agency* (ANY)

3. SAME_AS: The performer of the activity should be the same as that of another specified activity. An example is given below. Here, *Distributor* represents the business type of organizations that provide services as distributors; and *Activity1* is the name of the other activity. When the performer selection constraint is "SAME_AS *Begin-Activity*," it says that the e-service requests should be bound to the e-services provided by the organization that initiates the workflow instance.

   PERFORMER *Distributor* (SAME_AS *Activity1*)

4. VARIABLE: The performer of the activity is specified by the output parameter of a specified activity. An example is given below. The performer of the current activity is computed by *Activity2* and represented as the output parameter *bestManufacturer*. *Manufacturer* represents the business type of organizations that provide services as manufacturers.

   PERFORMER *Manufacturer* (VARIABLE *Activity2.bestManufacturer*)

The performer selection constraints given in (3) and (4) specify the relationship among the service providers of e-services specified in different activities. We shall call these constraints *inter-activity performer selection constraints*.

The activity body in the IMPLEMENTATION clause specifies a set of task items that need to be done inside the activity. The ACTIVITY_VAR defines the variables that can be used inside the activity body.

There are three basic types of task items inside the activity body, namely, e-service request, inline code, and event posting.

(1) E-service request. In a process model, e-service requests are the main task items in activity definitions. In DWM, we allow an activity to contain a number of e-service requests, if they are closely related and should be serviced by the same organization (i.e., as an atomic unit of work). Otherwise, these requests should be defined in separate activities.

The syntax of the e-service request in an activity-body is shown below:

ESERVICE <e-service name>.<operation name>

INPUT <in_attributes mapping>

OUTPUT <out_attributes mapping>

[CONSTRAINT <constraint definition>]

END_SERVICE

In the above syntax, *e-service name* is the name of the e-service to be requested, and *operation name* is the name of the operation that we wish to invoke. The *in_attributes mapping* in the INPUT clause specifies the mapping between the activity data (namely, input parameters of the activity and activity variables) and the input attributes of the e-service request. The *out_attributes mapping* in the OUTPUT clause specifies the mapping between the activity data (namely, output parameters of the activity and activity variables) and the output attributes of the e-service request. The optional

CONSTRAINT clause specifies e-service request constraints, which have been explained in Section 4.13.

(2) In-line code. Programming code can be included in the activity body to perform some simple computations or make variable assignments. We adopt the syntax of Java programming language for the in-line code. More complicated code can be defined and published as an e-service.

(3) External event posting. An activity can explicitly post events inside its activity body. These events are External events of an activity. The event can be synchronous or asynchronous. The following statements are used to post the event:

return_variables = PostSynch(*event_name, event_variables*);

PostAsych(*event_name, event_variables*);

The *event_name* is the name of the event being posted. The *event_variables* represent the attribute values of this event. The *return_variable* is the return value of a synchronous event.

4.4.2.2 Begin-Activity and End-Activity

As mentioned before, there are two special-purpose activities in a process model: the Begin-Activity and the End-Activity. The Begin-Activity is the entry point of a process model. It is used to pass input data to the process. The End-Activity is the exit point of a process. It is used to collect data from other activities of the process and treat the data as the output of the entire process. Different from regular process activities, the Begin-Activity and the End-Activity do not perform any tasks. They are dummy activities.

The Begin-Activity and the End-Activity can also be used respectively to specify the entry point and the exit point of a block in a process model. If a block is a loop block,

which defines a set of activities that can be iterated until an exit condition is met, the loop condition of this loop block should also be specified in the Begin-Activity (for a WHILE loop) or the End-Activity (for a UNTIL loop).

The specification of the Begin-Activity is shown below.

BEGINACTIVITY <process/block id>

[DESCRIPTION <description>]

IN_PARAMETERS <input parameters>

[LOOP <loop condition>]

END_BEGINACTIVITY

The specification of the End-Activity is shown below.

ENDACTIVITY <process/block id>

[DESCRIPTION <description>]

OUT_PARAMETERS <input parameters>

[LOOP <loop condition>]

END_ENDACTIVITY

4.4.2.3 Sample activity definition

A sample activity definition is given in Figure 4-8. The only task item in this activity is an e-service request to the operation *InitiateShipping* of the e-service *ShipOrder*. A constraint is specified for the e-service request, which states that the order should be shipped to the user within 5 days. This e-service request will be dynamically bound to a proper transportation agency during the execution of a workflow instance initiated from this process model. After the activity is finished, an asynchronous workflow event will be posted.

```
ACTIVITY Shipping
        DESCRIPTION  "Ship the order to the customer".
        IN_PARAMETERS   ProductDesc prod_desc, Integer quantity,
                                UserInfo user_info
        OUT_PARAMETERS  Boolean shipping_status
        PERFORMER TransportationAgency (ANY)
        WF_EVENT  async-after
        IMPLEMENTATION

                E-SERVICE ShipOrder.InitiateShipping
                    INPUT prod_desc, quantity, user_info
                    OUTPUT  shipping_status
                    CONSTRAINT
                        ATTRIBUTE_CONSTRAINT:
                            duration     int      [0 .. 5]          priority[1]
                     END_CONSTRAINT
                END_SERVICE

        END_IMPLEMENTATION
END  ACTIVITY
```

Figure 4-8: A sample activity definition.

4.4.3 Subflow

Subflow definition is used to refer to an existing process model in a process model definition. The syntax of the subflow definition is as follows:

SUBFLOW <subflow id> <referenced process id>

[DESCRIPTION <description>]

The *subflow id* represents the subflow identifier used in the business process model being defined. The *referenced process id* is the identifier of a process model, which is referenced in the current process model.

4.4.4 Connector

Connectors are used to specify such aggregation properties as Join and Split constructs and their constraints (OR, AND, and XOR) associated with activities. In WPDL, such information is defined inside activities. We extract all the control

information from the activity definitions and treat them as two separate modeling constructs. Thus, any modification on one will not affect the other.

The specification of the connector is as follows:

CONNECTOR <connector id>

[DESCRIPTION <description>]

AGGREGATION <aggregation information>

END_CONNECTOR

The *connector id* represents the unique identifier of the connector in the extent of the workflow process. The DESCRIPTION clause contains a text string describing the purpose of this connector.

The AGGREGATION clause specifies the aggregation information of the connector, including an aggregation type i.e., SPLIT or JOIN characterization, and an aggregation operator i.e., AND, OR, or XOR. An example of a specification of the AGGREGATION clause is shown below. Here, *transition1*, *transition2*, and *transition3* are three transitions.

AGGREGATION JOIN AND (*transition1*, *transition2*, *transition3*)

This specification defines an AND-JOIN connector, which represents a point in the process enactment where three parallel executing branches converge into a single flow of control. The process enactment continues at that point when all (i.e., AND) transitions leading to the connector have arrived.

4.4.5 Transition Information

The Transition Information specifies the transitions between activities, connectors, and subflows. The conditions associated with transitions enable or disable the transitions during a workflow execution. Transitions and connectors form the control

flow of a business process model. In addition to specifying the control flow, data flow information can also be specified in a transition. When the parameter mapping goes the same way as the transition (i.e., has the same source activity and the same target activity), instead of defining a separated data flow, we include the mapping information in the transition definition.

The specification of Transition Information is shown below:

TRANSITION <transition id>

[DESCRIPTION <description>]

FROM <entity id> TO <entity id>

[CONDITION <transition condition>]

[DATA_MAPPING <inter-activity parameter mapping>]

END_TRANSITION

The FROM clause defines the source entity and the destination entity of the transition. The entities that can be used as the source or the destination include activity, connector, and subflow. The CONDITION clause specifies the condition attached to the transition. The DATA_MAPPING clause specifies the parameter mapping between two activities.

4.4.6 Data Flow

If the data flow between two activities in a process model does not coincide with the control flow specified by a transition between them, a Data Flow specification is used to explicitly define the data flow. For example, data may flow from an activity to another activity that is not directly connected by a transition between them. The DATAFLOW specification is given below:

DATAFLOW <data flow id>

[DESCRIPTION <description>]

FROM <activity id> TO <activity id>

<parameter mapping info>

END_DATAFLOW

The *parameter mapping info* specifies the mapping between one activity's output parameters and another activity's input parameters.

4.4.7 Data Class

The data class definition defines the data types that are used to specify the data flow between activities. The specification is shown below:

DATACLASS <data class id>

[DESCRIPTION <description>]

4.4.8 Event Information

The activities can post external events inside the activity body. These events should be declared first, in order to be used inside the activity body.

EVENT <event id>

[DESCRIPTION <description>]

4.5 Dynamic Properties Provided by DWM

By adding connectors, events, triggers, and rules as modeling constructs, encapsulating activity definitions, and allowing e-service requests as a part of the activity specification, DWM extends WPDL to provide dynamic properties needed to support the requirements of inter-organizational business processes. The dynamic properties of DWM can be classified into four categories: active, flexible, adaptive, and customizable.

4.5.1 Active

The business events and business rules are integrated with business processes. A business process is active in the sense that its enactment may post synchronous and/or asynchronous events to trigger the processing of business rules.

Synchronous workflow events posted may trigger rules that make changes to the business process either by adding additional tasks to the business process or by altering the process model itself at run-time. These rules can be defined either by the process model designer or the organization that initiates a workflow instance. Synchronous workflow events are posted directly to the local ETR Server during the execution of workflow instances.

Asynchronous workflow events are notifications of the processing milestones of an enacted business process. These events are published through the Event Server of an ISEE Hub. Interested organizations can subscribe to these events and define business rules to react to these events. During the execution of workflow instances, asynchronous workflow events are posted to the local Event Server so that these events can be transferred by its notification mechanism to the Event Servers of the ISEE Hubs of the subscribers to trigger their business rules. These business rules may in turn enact other business processes.

4.5.2 Flexible

The e-service requests specified inside a process model are defined according to standardized e-service templates. These e-service requests are bound to suitable service providers in a virtual enterprise during the enactment of the business process through a dynamic service binding mechanism. Thus, a process model defined in this way is flexible in the sense that the actual business organizations that take part in the business

process are determined at run-time. Changes in the membership of the service providers of a particular service will not affect the enactment of a business process.

The Workflow Engine accomplishes the dynamic service binding with the help of the Broker Server, which performs the constraint-based service provider selection. Before the Workflow Engine calls the Broker Server, some preprocessing work needs to be done to determine which e-services requested in the process model need to be provided by the same service provider. For example, the provider that processes a purchase order should also handle the shipping of the product and the issuing of an invoice. This information is given to the Service Broker to select the proper providers. In addition to the reason given before (i.e., allowing a process model to be enacted by a qualified user and controlled by a workflow instance of DynaFlow at any site), the need to do preprocessing of a process model is another reason for replicating the entire model in all collaborative business sites.

4.5.3 Adaptive

Because a process model itself can be modified by a triggered rule to adapt to the changing business environment, the process model defined in DWM is adaptive. Modifications to activity definitions or to the structural relationships and constraints of these activities can be more easily done because their specifications are separated.

We adopt the “code generation” approach to implement the Workflow Engine in DynaFlow. Based on the meta-information of a process model, run-time workflow structures (capturing the structural relationships of activities) and activity codes (capturing the task items in activity definitions) are generated for the Workflow Engine to schedule and execute the instances of this process model. Since the Workflow Engine uses an interpretive approach to schedule the processing of different activities based on

the run-time workflow structures, ad hoc modifications to the process model by modifying the workflow structures can be achieved.

The Workflow Engine provides APIs to do run-time modifications to a process model: e.g., to delete/add a transition, delete/add a data flow, replace an activity, and/or modify a condition of a transition. Please refer to Chapter 5 for detailed information about the code generation approach and ad hoc modification to a process model. Run-time modifications to the process model can be done either by the business rules triggered by synchronous workflow events, or by the user who monitors the processing of a workflow instance.

4.5.4 Customizable

Inter-organizational process models are designed for conducting the business of a virtual enterprise. People who work for a participating organization and have the right of access to a process model can enact the model at its home site. For all the enactments (i.e., all workflow instances) an organization may want to customize the model to suit its local business policies, constraints or regulations. This can be achieved by defining its own set of business rules to perform additional work and/or to modify the process model. These rules are stored in the ETR Server at the site of the organization. During the execution of every workflow instance, the synchronous workflow events are posted to the local ETR Server to trigger these rules. We shall call this type of customization the *organizational customization*.

Also, in different enactments of the same process model, an organization may want different rules to be triggered. This can also be achieved by defining different sets of rules for different enactments. Before a workflow instance is initiated, a unique id is generated for it. This id is incorporated in the rules that should be invoked during the

processing of the workflow instance. When a synchronous event is posted, the instance ID of the workflow instance that posts this event is passed to the rules and is used to identify the proper rules that are associated with the event. We call this type of customization the instance customization.

4.6 Sample Scenario: Order Processing in a Supply Chain Community

We use an order processing in a supply chain as a scenario to illustrate how DWM is used to define the business process model and how the dynamic properties described above can be achieved. Suppose several organizations form a supply chain named *Supply_Chain_Community*. The participating organizations are categorized into four different business types according to the different roles they play: *Retailer*, *Distributor*, *Manufacturer*, and *Transportation Agency*.

A process model, *OrderProcessing*, is defined in DWM and can be used by distributors in *Supply_Chain_Community* to process the orders issued by retailers, as illustrated in Figure 4-7. The Distributor gets an order from a retailer and checks the inventory to make sure that there are enough products in the inventory to satisfy the order. If the order can be satisfied, the Distributor adjusts the inventory accordingly by reducing the quantity of the product. It then acknowledges the order and asks the Transportation Agency to ship the product to the Retailer. If there are not enough items of the product for this order, the order processing fails.

In Figure 4-9, activities are represented by boxes. The descriptions of the activities are inside the corresponding boxes. The business type information of each activity is indicated above the box. The performer selection constraint of each business type is enclosed within parentheses following the business type. For example, “ANY”

following the business type *TransportationAgency* means that the e-services requests in the activity can be serviced by any transportation agency that can be bound to the service requests at run-time. Small circles in the graph represent connectors. For example, *CheckSplit* is a SPLIT connector with XOR property, and *CheckJoin* is a JOIN connector with XOR property. T1 to T12 are transition names.

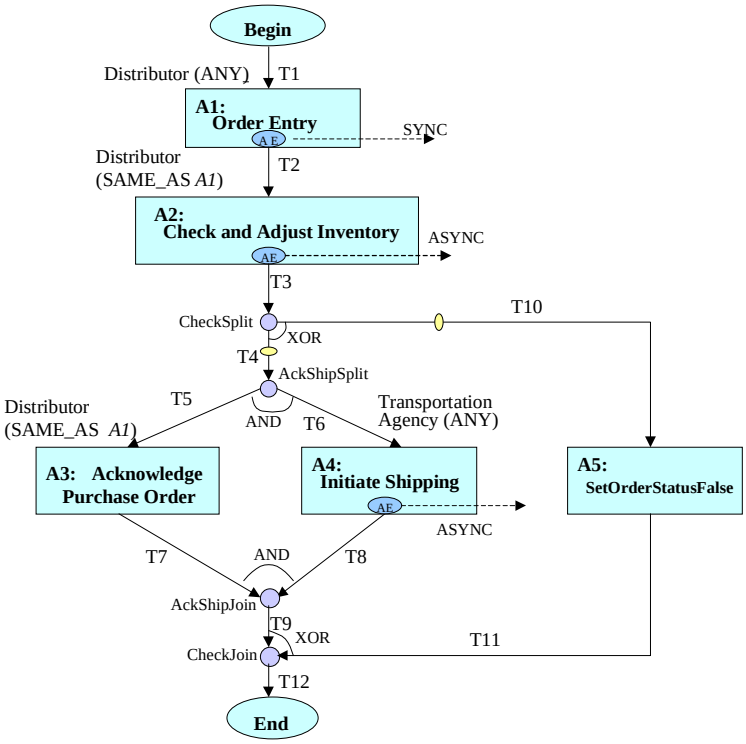


Figure 4-9: *OrderProcessing* model for *Distributors* in the *Supply_Chain_Community*.

The Before-Activity-Events and After-Activity-Events posted by the activities of this process model are also shown in this figure. They are specified by the ovals inside the activity boxes. "SYNC" and "ASYNC" specify the types of the workflow events to be posted. To avoid cluttering the figure, we do not show the data flow information and the detailed specification of the activities.

This model is replicated at the ISEE Hubs of the collaborative organizations. A qualified organization can enact this process model to process orders from retailers. It

can customize the process model by defining business rules that enforce its local business policies, then connecting these rules to the synchronous workflow events of the process model by using trigger specifications. For example, the distributor *WorldWide* has a local business policy, which says that after the order from the retailer is received, the credit history of this retailer needs to be checked before further processing. To enforce this policy, *WorldWide* can define a business rule and attach it to the synchronous event that is posted after activity *A1* (Order Entry) has been processed. This rule is triggered to check the credit history of the retailer who submitted the order. If the credit history of the retailer is good, the processing will continue. Otherwise, the rule can modify the process model at run-time so that after activity *A1* (Order Entry) is finished, activity *A5* (Set Order Status False) is scheduled for processing. To achieve this, transition *T3* needs to be deleted and a new transition, which goes directly from A1 to *A5*, needs to be added.

Business rules attached to workflow events can in turn enact other business process models. For example, if *WorldWide* wants to check if replenishment to the inventory is needed after adjusting the inventory, it can define a business rule and attach it to the asynchronous event posted after activity *A2* (Check and Adjust Inventory). This business rule first checks if the quantity of the product in the inventory has reached the replenishment threshold. If this is the case, a workflow instance of the *InventoryReplenishment* process model, which generates an order of this product to a manufacturer, is created and executed.

We already saw the definition of activity *A4* (Initiate Shipping) in Figure 4-8. Since the performer selection constraint of this activity is “ANY,” the e-service requests

inside this activity will be dynamically bound to a service provider selected through the constraint-based service-provider selection function provided by the Broker Server.

CHAPTER 5
DESIGN AND IMPLEMENTATION

In Chapter 3, we have presented the architecture of DynaFlow, which is composed of a number of servers. Among them, the Workflow Server is the focus of our work. There are two main components in the Workflow Server; namely, the Process Definition Tool and the Workflow Engine. This chapter describes the design and implementation of these two components in detail.

5.1 Design and Implementation of the Process Definition Tool

The Process Definition Tool is a major build-time tool for process modeling. In addition to the Process Definition Tool, there are other servers and components used for build-time activities: a Metadata Manager, a Knowledge Profile Manager, a Workflow Code Generator, and a Constraint Definition GUI.

In this section, we first introduce the build-time environment of DynaFlow. Then we present the design and implementation details of the Process Definition Tool.

5.1.1 Build-time Environment of DynaFlow

The Process Definition Tool (PDT) is a user-friendly graphic editor for creating and customizing process models using DWM as the underlying model. It can be used to construct the diagram of a business process model in a user-friendly and graphic manner. The Process Definition Tool is integrated with the other components of the Workflow Server and other servers used by DynaFlow.

When using the Process Definition Tool to define a process model, the process model designer needs to make use of e-service templates, which are accessible from the Broker Server, to specify e-service requests inside activities. To facilitate the process modeling in our implementation, we store these e-service templates in a Metadata Manager implemented for another project [Lee00]. When the process model designer specifies an e-service request using the Process Definition Tool, the Process Definition Tool retrieves the corresponding e-service template from the Metadata Manager and presents it to the process model designer. For each e-service request, the process model designer can specify e-service constraints by invoking the Constraint Definition Graphical User Interface reported in [Hua00; Su01a] through the Process Definition Tool.

The Process Definition Tool can also invoke a Knowledge Profile Manager (KPM), which provides a number of graphical tools: Data Class Editor, Event Editor, Trigger Editor, and Rule Editor. The Data Class Editor is used to define data classes manipulated or transmitted by the activities. The Event Editor is used to define external events that are posted inside activities of a process model. The Trigger Editor and the Rule Editor are used to define triggers and rules associated with the synchronous and asynchronous workflow events.

The meta-information of process models, e-service request constraints, data classes, events, rules, and triggers, together with the e-service templates, are all stored and maintained by the Metadata Manager.

If some workflow events are defined for a process model, the Process Definition Tool generates the meta-information of these events and stores it in the Metadata Manager, then generates workflow event classes that are used for event posting. It also

makes calls to the Event-Trigger-Rule (ETR) Server to install these events so that these events can trigger the rules attached to them.

Finally, the Process Definition Tool calls the Workflow Code Generator to generate the run-time workflow structures and activity codes based on the meta-information of a process model. The run-time workflow structures and activity codes are used by the Workflow Engine at run-time to schedule and control the execution of workflow instances. We will explain them in detail in the next section.

The build-time environment of DynaFlow is shown in Figure 5-1. The KPM, Event Server, ETR Server, and Matadata Manager were developed for another project called Iknet [Lee00, Su00]. They together with others form the components of DynaFlow.

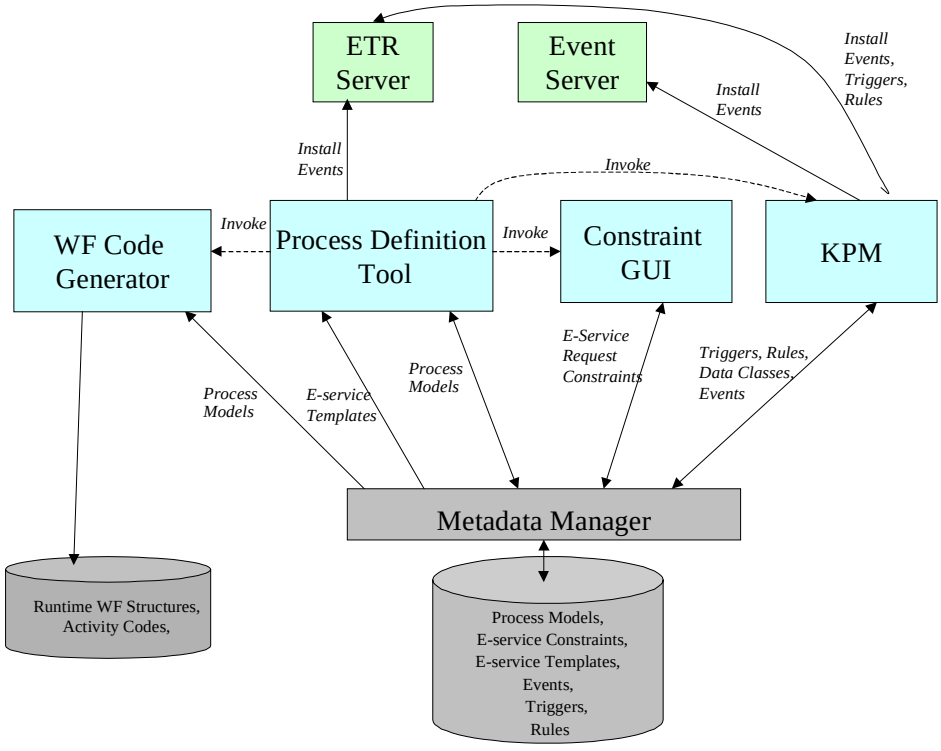


Figure 5-1: Build-time environment of DynaFlow.

5.1.2 Design and Implementation of the Process Definition Tool

The Process Definition Tool is implemented using the JavaBeans technology. The screen layout of the Process Definition Tool is composed of three parts: the Main Menu, the Tree Manager, and the Graphic Editor, as shown in Figure 5-2.

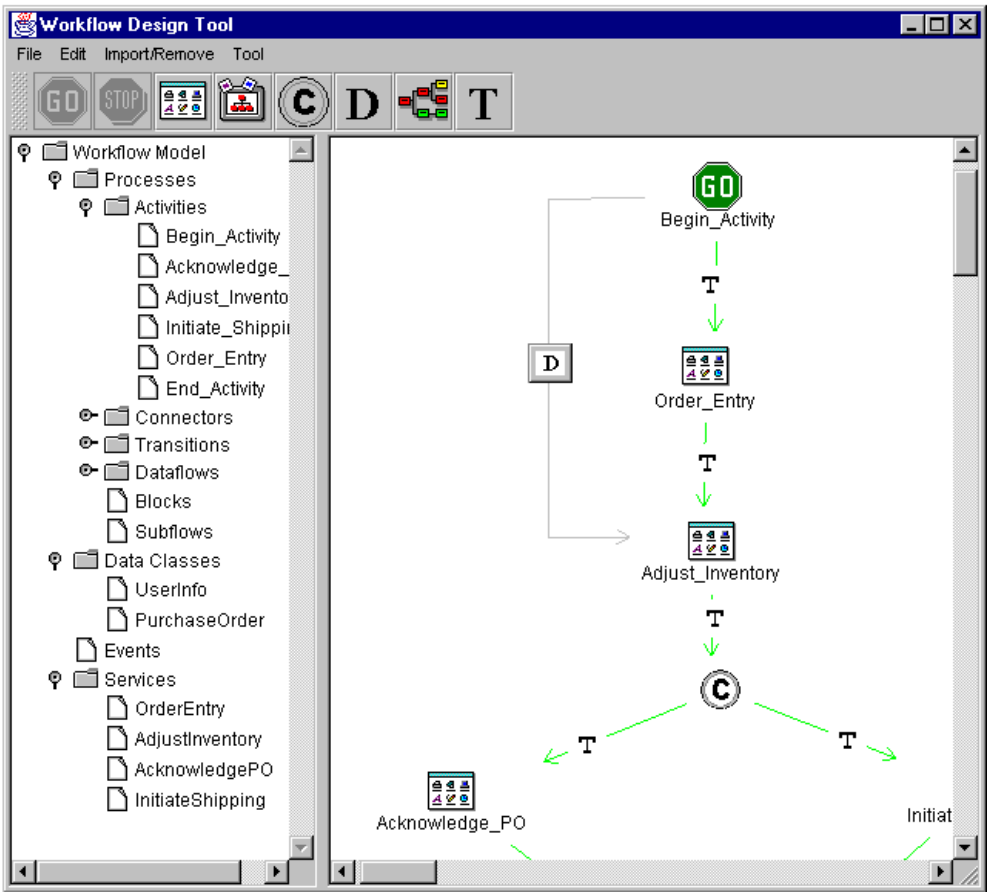


Figure 5-2: The screen layout of the Process Definition Tool.

The Main Menu is shown at the top of the screen. The menu bar provides functions to facilitate process modeling. The functions include creating, loading, and saving process models; importing useful objects into the process model, which include data classes, e-services, and events; invoking the Constraint Definition GUI and KPM; invoking the Workflow Code Generator; and so on.

The Tree Manager is on the left of the window shown in Figure 5-2. It displays all the entities of a process model in a hierarchical manner, including activities,

transitions, connectors, data flows, blocks, and subflows. In addition to these entities, it also displays the imported objects, such as events and e-services.

The Graphic Editor is on the right side of the window. It is used to draw the diagram that represents a process model. The Graphic Editor consists of a drawing tool palette, a drawing area, and customized editors for constructing entities in the diagram. The drawing tool palette displays various icons, each of which corresponds to a modeling construct. The drawing area is provided for the model designer to draw a process model by adding various icons to the work area. Nodes in the diagram represent entities, such as activities, sub-flows, connectors, and blocks. Edges between nodes represent the transitions and data-flows between these entities. When a model designer double-clicks an entity in a program diagram, the corresponding customized editor will show up. It provides a detailed view of the specification of that entity.

The Graphic Editor can be either in a Process Mode or in a Block Mode. The former is used to create a new process model or to edit/delete an existing process model. The latter is used to create/edit a block inside a process model.

Let us take a look of the Activity Editor used for activity definitions. To define an activity, the model designer needs to specify activity information, including general information, parameter information, task items, etc. using the Activity Editor. By double-clicking an activity icon, the corresponding Activity Editor would appear. The Activity Editor has three property pages: a general property page, a parameter property page, and a task property page, as shown in Figure 5-3.

The general property page is for editing the general information of an activity, e.g. the activity name, activity description, and performer and performer selection constraint.

It is also used to specify what kind of workflow events this activity should post. When the OK button is pressed, the event information of the specified workflow events will be stored in the Metadata Manager, and the corresponding event classes are generated.

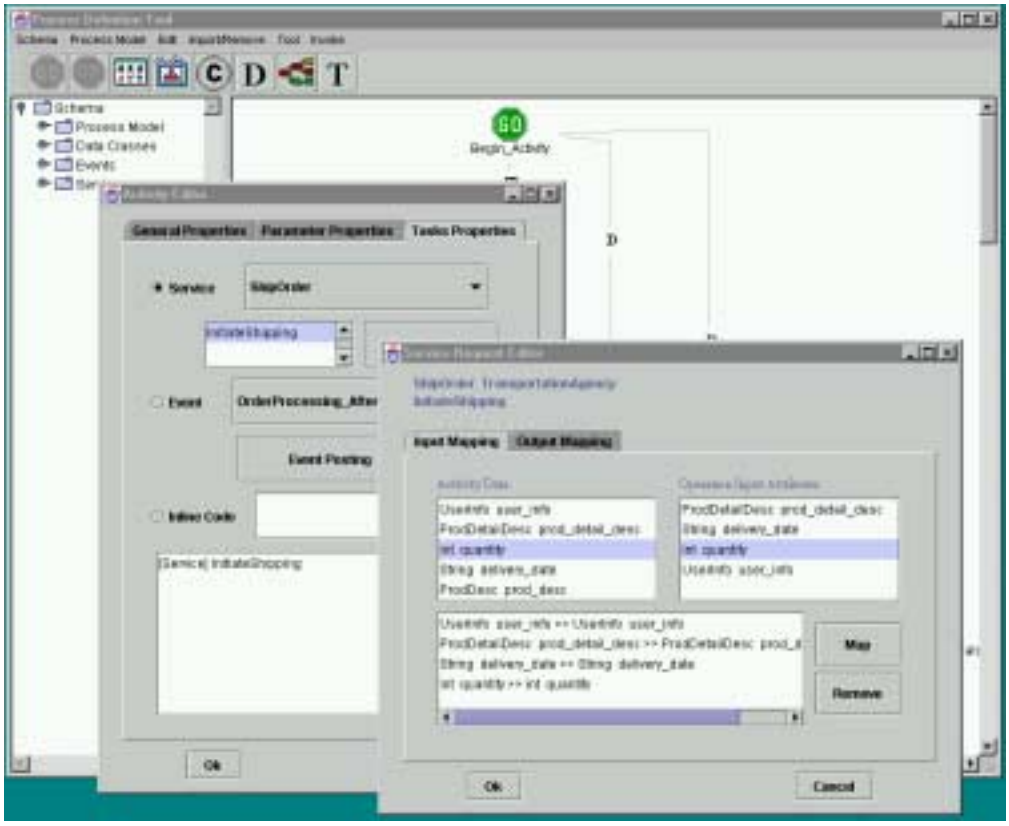


Figure 5-3: Activity editor and e-service request editor.

The parameter property page is for editing the input parameters, the output parameters, and activity variables. Activity variables can be used inside the activity body where task items are defined.

The task property page is for specifying the schedule of a list of task items to be performed in this activity. Three different types of task items can be specified: the e-service request, the event posting, and the inline code. For example, to add an e-service request to the task item list, the process model designer first selects an e-service provided by the business type specified as the performer of this activity. Then the designer needs

to select an operation of this e-service. An e-service request editor will appear after an operation is selected, showing the input mapping and output mapping information for this operation. An e-service request will be added to the task item list of this activity after the process model designer finishes the mapping and clicks the OK button.

The Activity Editor gets the e-service information, i.e., e-service name, operations of an e-service, and the input attribute and output attribute information of the operations from the e-service templates pre-stored in the Metadata Manager.

If the performer selection constraint of an activity is ANY, it means that the e-service requests specified in the activity will be dynamically bound to the qualified e-services of any provider at run-time through the dynamic service binding process. In this case, the process model designer would invoke the Constraint Definition GUI through the Process Definition Tool to define the performer selection constraint. The Constraint Definition GUI is a dynamic HTML Web page, which is generated based on the corresponding e-service template information stored in the Metadata Manager. The Web page can be used to define the attribute constraints and inter-attribute constraints for an e-service request.

5.2 Design and Implementation of the Workflow Engine

In DynaFlow, we use a “Code Generation” approach to implement the Workflow Engine. In this section, we first introduce the “Code Generation” approach. We then describe the run-time workflow structures and activity codes that are used by the Workflow Engine to schedule the execution of workflow instances, and give the design and implementation details of the Workflow Engine. Lastly, we describe the technique used to achieve the run-time modification of a process model.

5.2.1 "Code Generation" Approach for the Workflow Engine

In our system, a Workflow Code Generator is used to process a process model's meta-information and to generate *run-time workflow structures* and *activity codes*. The run-time workflow structures contain the transition information and data flow information of activity interconnections. These structures are used by the Workflow Engine to enforce the inter-activity dependencies of a workflow instance. The specifications of the activity bodies are translated into Java programs and compiled into Java classes, which are called *activity codes*. When the Workflow Engine schedules an activity for execution, the engine simply loads the corresponding activity code from a run-time repository and executes the code directly to perform the specified task items.

This "code generation" approach results in a lightweight, efficient, and adaptive Workflow Engine. It is lightweight because the Workflow Engine does not need logic to interpret the activity specification in order to execute the task items in an activity definition. Taking advantage of the performance of the compiled classes, the Workflow Engine is efficient. Finally, since the Workflow Engine "interprets" the run-time workflow structures to enforce the inter-activity dependencies, it is adaptive in the sense that it is easy to modify the execution course of a workflow instance by modifying the workflow structures at run-time. Dynamically loading activity code at run-time also enables the run-time modification of the task items in an activity. Any changes made to the task items in an activity body will be reflected in the execution of the activity as long as the re-generated code is placed in the run-time repository *before* the execution of the activity. Thus, flexibility is maintained without sacrificing performance.

5.2.2 Run-time Workflow Structures

There are basically three types of run-time workflow structures: entity structures, control flow structures, and data flow structures. We use the sample scenario shown in Figure 4-7 to illustrate the design of these run-time workflow structures.

5.2.2.1 Entity structures

Entity structures provide the Workflow Engine with the necessary information to schedule and execute the following entities in a process model: activities, blocks, and subflows. The contents for the entity structures correspond to the metadata of these entities.

The entity structure for an activity is intended to provide the necessary information to the Workflow Engine for scheduling the activity and executing the generated activity class. Figure 5-4 shows the entity structure for the activity *Shipping*, whose specification was shown in Figure 4-4. The entity structure consists of the following information: activity id, input parameters, output parameters, performer, performer selection constraint, and e-service requests and their constraints. The information can be obtained directly from the specification of the activity, which is stored in the Metadata Manager. The entity structures are stored in hash tables as a part of the run-time workflow structures. The keys to these hash tables are the entity names.

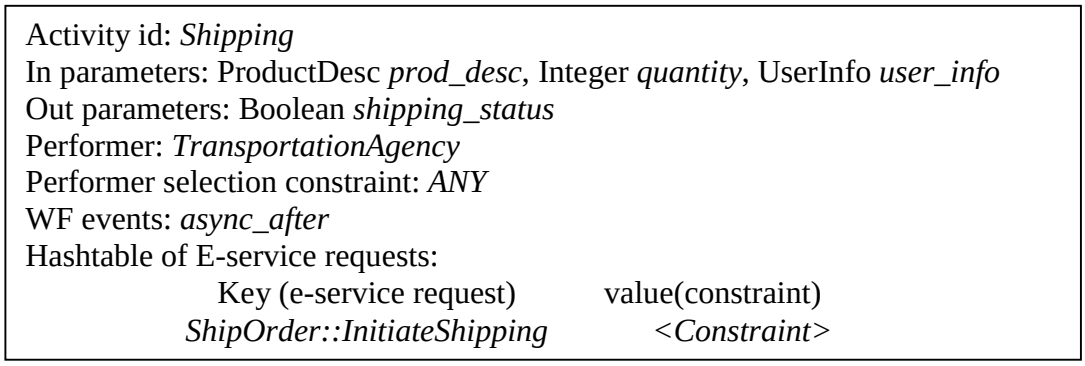


Figure 5-4: Entity structure of an activity.

5.2.2.2 Control flow structures

Control flow structures capture the control dependencies among the entities defined in a process model. A control flow structure is generated for each of the following entities of a process model: activity, subflow, block, connector, and End-Activity. During the execution of a workflow instance, the Workflow Engine determines whether the next entity can be scheduled according to its corresponding control flow structure.

There are three main parts in a control structure: entity name, aggregation property, and transition(s):

- Entity name: The name of the entity corresponding to the control flow structure.
- Aggregation property: If the entity is a JOIN connector, the aggregation property of the control structure is the same as the aggregation property of the JOIN connector (AND, OR, or XOR). Otherwise, the aggregation property is SIMPLE.
- Transition(s): A control flow structure may contain one or more transitions. If the entity of the control structure is a JOIN connector, there are multiple transitions in this control flow structure. Otherwise, there is only one transition in the structure.

The transitions in a control flow structure are stored in a hash table. The key to the hash table is the name of the source entity name of a transition, and the values stored in the hash table entry are the transition name and the condition of the transition.

In the process model *OrderProcessing*, shown in Figure 4-7, the control structure for activity *A2* (Check and Adjust Inventory) is shown in Figure 5-5. In this control flow structure, there is only one transition *T1* coming out of activity *A1* with no condition on it.

Entity name: A2	
Aggregation property: SIMPLE	
Transition:	
key(source entity)	value
A1	T2; NULL

Figure 5-5: Control flow structure of activity *A2*.

The control flow structure for the JOIN connector *CheckJoin* is shown in Figure 5-6. There are two transitions in this control flow structure. They all have no conditions on them.

Entity name: CheckJoin	
Aggregation property: XOR	
Transition:	
key(source entity)	value
A5	T10; NULL
AckShipJoin	T9; NULL

Figure 5-6: Control flow structure of connector *CheckJoin*.

The control flow structures are stored in a hash table with the source entity names of the transitions as the keys, as shown in Figure 5-7. The control structure for a JOIN connector has multiple transitions, and thus multiple source entities. In this case, there are multiple entries in the hash table pointing to the same control flow structure. For example, in the control flow structure for connector *CheckJoin*, there are two entries pointing to it: *A5* and *AckShipJoin*.

5.2.2.3 Data flow structures

A data flow structure captures data dependency, i.e. the parameter mapping information between two entities in a process model. Data flow structures are intended for the Workflow Engine to map the data outputted by entities (such as activities, subflows, and blocks) that have been successfully completed to entities that need these

data to proceed. Data flow structures are generated from the data mapping information in a transition specification or from an explicit data flow specification.

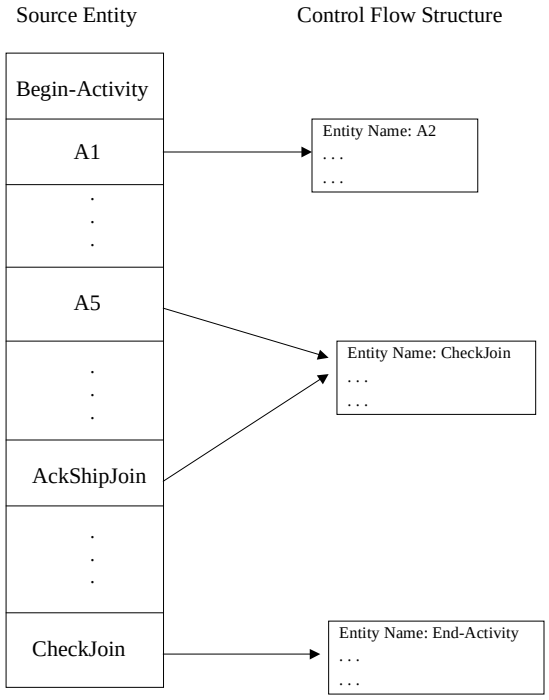


Figure 5-7: Hash table of control flow structures.

There are three kinds of information in a data flow structure, namely, the source entity name, the target entity name, and the parameter mapping information from the source entity to the target entity.

For example, in the process model *OrderProcessing*, a data flow structure that captures the data dependencies between activity *A1* (Order Entry) and activity *A2* (Check and Adjust Inventory), is shown in Figure 5-8.

Source entity name: *A1*
Target entity name: *A2*
Parameter mapping info:
prod_desc >> prod_desc
quantity >> quantity

Figure 5-8: Data flow structure from A1 to A2.

The data flow structures are stored in a hash table with the source entity name as the key. The value stored in a hash entry is another hash table containing data flow structures coming from the source entity. The key to the second hash table is the name of the target entity.

5.2.3 Activity Code

In the "code generation" approach, an activity body, which may contain several task items, is translated into *activity code*. During the execution of a workflow instance, once an activity is scheduled, the corresponding activity code is dynamically loaded and executed.

Activities are the fundamental building blocks of a business process model. In DWM, activity information is encapsulated by specifying an activity's input and output parameters. Task items within the activity body can be specified as e-service requests, event postings, or inline codes. There are several factors making it possible to execute an activity in a "compiled" way: there is no control information in the specification of an activity; task items in the activity body only makes reference to the data passed through the input parameters; and the results of the activity execution are only exposed to any entity outside of the activity through output parameters.

The general structure of the activity code is shown in Figure 5-9. The parenthesized parts are optional depending on whether or not there are corresponding task items in the activity specification. We can see that there are basically three parts in an activity code: member variable declarations, the constructor method, and the activate method.

The activity's input parameters, output parameters, and activity variables are translated into member variables of the activity code. A vector is declared to contain

```
Import statements
Class processName::activityName
{
  declaration of variables for input and output parameters
  declaration of activity variables
  declaration of vector for output parameter values
  declaration of variables related to e-service requests
  declaration of variables related to event postings

  processName::activityName()  //  Constructor
  {
        /* If there are e-service requests in the activity.*/
        [ Get reference of the broker proxy ]

       /* If there are event postings in the activity.*/
        [ Get references of the event server and the ETR server ]
  }

  ActivityResult  activate(Vector inputValues, String businessType,
                           Hashtable serviceConstraints )
  {
          Initialization of input variables
                                .
                                .
                                .
         /* Following statements are for dynamic service binding */
         [ Put input attribute values of an e-service to the e-service request constraint ]
         [ Contact broker proxy to get the service provider's URL (dynamic service binding) ]
                                .
                                .
                                .
         /* Following statements are for an e-service invocation */
         [ Generate SOAP request message ]
         [ Invoke e-service and get SOAP response message ]
         [ Get the return value from the SOAP response message ]
                                .
                                .
                                .
         /* Following statements are for an event posting. */
         [ Construct the event object ]
         [ Post synchronous event to the ETR Server ]
         [ Post asynchronous event to the Event Server ]
                                .
                                .
                                .
         /* For inline code, directly copy it to the activity code. */
         [ inline code ]
                               .
                               .
                                    .
  }
```

Figure 5-9: General structure of an activity code.

values of the output parameters of the activity. Besides these common declarations, other variables necessary to handle e-service requests and event postings inside the activity also need to be declared.

A constructor method is generated for each activity code. This method is used to contact the broker proxy (if dynamic service binding is needed), the Event Server (if there are any asynchronous event postings in the activity), and the ETR Server (if there are any synchronous event postings in the activity). A service client and a SOAP message generator are also initiated here if there are any e-service requests in the activity. The service client is used to contact the E-Service Adapter at the service provider's site to invoke e-services. The SOAP message generator is used to generate SOAP request messages based on the input values of an e-service and to receive output values from a SOAP response message.

For each activity code, there is one *activate*() method. It performs task items specified in the activity body. According to the type of the activity's performer selection constraint, the generated *activate* method will have different sets of parameters.

If the activity's performer selection constraint is of type ANY, then dynamic service binding needs to be performed inside the *activate* method. In this case, the *activate* method generated has three arguments, namely, a vector containing values for the activity's input parameters, a string containing the business type of the performer, and a hash table containing e-service request constraints specified in the activity. Based on the business type and e-service request constraints, e-service requests are dynamically bound to appropriate e-services of service providers. For each e-service request, a SOAP message containing the values of input attributes of this e-service is generated and sent to

the e-service adapter at the service provider's site through the service client to invoke the e-service. After that, the service provider returns the output values in the form of a SOAP message. The activity code then extracts values from the SOAP message. For event postings inside the activity, corresponding event objects are generated. Synchronous events are posted to the ETR Server, and asynchronous events are posted to the Event Server. Inline codes are directly copied to the activity code. At the end of the activate method, the result, containing the service provider's URL and the values of the activity's output parameters, is returned.

If the activity's performer selection constraint is NOT of type ANY (it will be one of the other three types: CONSTANT, SAME_AS, or VARIABLE), the generate *activate* method has following two arguments; a vector containing values for the activity's input parameters and a string containing the service provider's URL.

Dynamic service binding is not needed for e-service requests specified in such an activity. The service provider's URL can be obtained according to the performer specification of the activity, then passed to the activate method.

5.2.4 Design and Implementation of the Workflow Engine

5.2.4.1 Architecture of the Workflow Engine

The enactment of a business process is performed by the Workflow Engine, which forms the core of the run-time environment. The Workflow Engine uses a multi-thread architecture to manage the execution of workflow instances. The multi-thread architecture allows the concurrent execution of multiple workflow instances.

For one workflow instance, a Workflow Scheduler thread is created. It schedules the transitions and data flows between activities, blocks, and subflows, and thus controls the execution of the workflow instance.

When an activity is scheduled, the Workflow Scheduler creates an Activity Handler thread to load the activity code, and executes it. Thus, the Workflow Scheduler can continue on and the parallel activities in a process model can be executed concurrently by multiple, concurrent Activity Handler threads.

If there are e-service requests specified inside an activity, a service client is created to invoke the corresponding remote e-services. The service client sends a SOAP message containing the request data to a remote E-Service Adapter, and receives a SOAP message containing the retuned data. When a block or a subflow is scheduled, the Workflow Scheduler creates a Block Scheduler thread or a Subflow Scheduler thread, and delegates the scheduling of the block or the subflow to it. The Block Scheduler and the Subflow Scheduler works in a similar way as the Workflow Scheduler: they create Activity Handler threads to handle the execution of the activities contained in the block or subflow. The Workflow Scheduler for a workflow instance is responsible for the synchronization of these concurrent threads. The architecture of the Workflow Engine is shown in Figure 5-10

5.2.4.2 Implementation details

The Workflow Scheduler uses run-time workflow structures to enforce inter-activity dependencies during activity scheduling. There are two kinds of inter-activity dependency: control dependency and data dependency. Similar to run-time workflow structures, control flow structures are used to capture the control dependencies, and data flow structures are used to capture the data dependencies.

During the execution of a workflow instance, the Workflow Scheduler evaluates control flow structures to schedule corresponding entities, and carries out the parameter mapping according to the parameter mapping information contained in the data flow

structures. The evaluation of the control structures and the parameter mapping process are triggered by the completion of a source entity in the control structures and data flow structures.

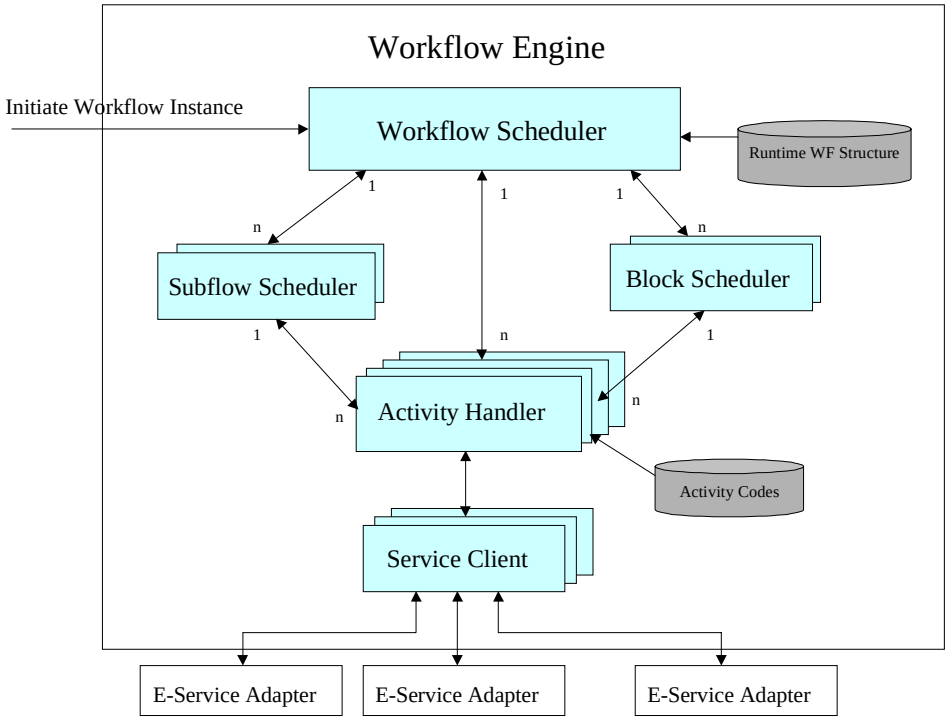


Figure 5-10: Architecture the Workflow Engine.

An activity is completed when all the task items inside the activity are finished. A connector, the Begin-Activity, and the End-Activity are completed as soon as they are scheduled. A block is completed when all the activities in the block are completed (if the block is a loop-block, the loop needs to be finished). A subflow is completed when the enactment of the referred process is finished. The completion of the End-Activity represents the completion of the process enactment.

The components of a Workflow Scheduler are shown in Figure 5-11. When a workflow instance is initiated, a Workflow Scheduler thread is created. It first reads the run-time workflow structures, including control flow structures and data flow structures,

from the run-time repository. The execution of the workflow instance begins from its Begin-Activity. That is, when a workflow instance is initiated, its Begin-Activity is scheduled immediately. The Begin-Activity is completed right after it has been scheduled because it does not contain any task.

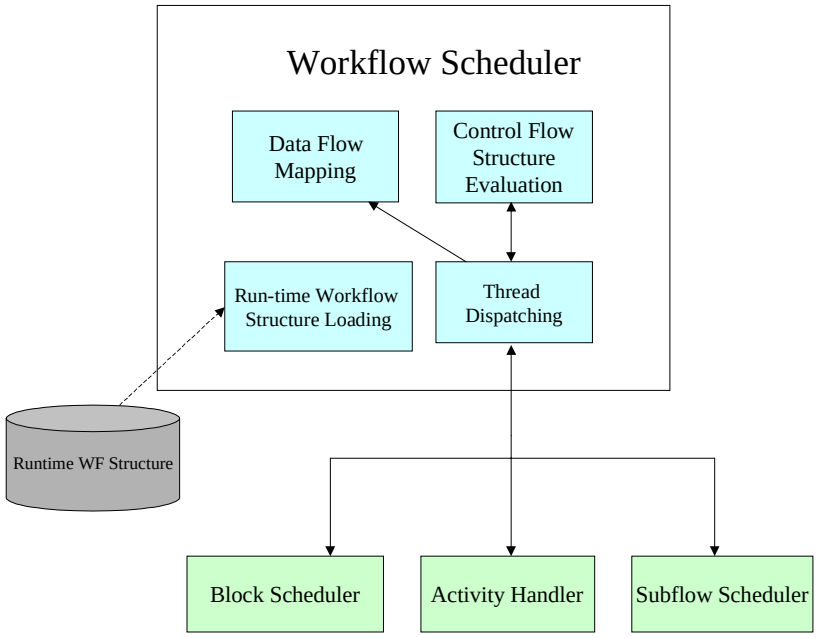


Figure 5-11: Components in a Workflow Scheduler.

After an entity is completed, the Workflow Scheduler needs to take the following steps to schedule the execution of the workflow instance:

(1) Data flow mapping.

If the completed entity is an activity, a subflow, or a block, the Workflow Scheduler maps its output data to the input parameters of the entities, which are the target entities of data flow structures with the completed entity as the source entity.

(2) Control flow structure evaluation.

The Workflow Scheduler retrieves the control flow structures, which contain a transition with the completed entity as the source entity, from the hash table of the control flow structures and evaluates them.

If the aggregation property of a control structure is SIMPLE, there is only one transition in this control flow structure. In this case, the evaluation of the control flow structure is actually the evaluation of the condition on this transition. If there is no condition on the transition, the result of the evaluation is always true. As long as the condition is evaluated to True, the target entity will be scheduled. We use an "interpretive" approach to evaluate a condition.

If the aggregation property is not SIMPLE, it means that the target entity must be a JOIN connector. In this case, after the transition, whose source is the completed entity, is evaluated to True, the Workflow Scheduler also needs to evaluate the aggregation expression, which is formed by the transitions and the aggregation operator (AND, OR, or XOR). For example, for JOIN connector with AND aggregation property *CheckJoin* showing in Figure 5-5, the expression "T9 AND T10" needs to be evaluated. The target entity is scheduled only when the aggregation expression is evaluated to True. Thus, for connector *CheckJoin*, the Workflow Scheduler will wait until both T9 and T10 are evaluated to be true before scheduling the connector *CheckJoin*. In this way, the Workflow Scheduler synchronizes multiple parallel branches.

(3) Entity scheduling and execution

If the scheduled entity is an activity, the Workflow Scheduler creates an Activity Handler thread and delegates the execution of the activity to it. At the end of the execution, the Activity Handler thread passes the output data of the activity to the

Workflow Scheduler to do the data flow mapping, notifies the Workflow Scheduler about the completion of its execution to trigger the control flow structure evaluation, and dies.

If the scheduled entity is a block, the Workflow Scheduler creates a Block Handler thread to handle the execution of the block. The Block Handler schedules the execution of the activities inside the block. It also handles the loop if the block is a loop-block. After the execution of the block, the Block Handler thread notifies the Workflow Scheduler, passes the result to it, and dies.

If the scheduled entity is a subflow, a Subflow Scheduler thread is created to handle the execution of the subflow. The execution of a subflow is actually a workflow instance of the referred model, so the work of the Subflow Scheduler is quite the same as the Workflow Scheduler. After the execution of the subflow, the Subflow Scheduler thread notifies the Workflow Scheduler, passes the result to it, and dies.

If the scheduled entity is a connector, the Workflow Scheduler sets the status of the connector to “complete” and does nothing. If the scheduled entity is the End-Activity of the process model, the workflow instance is completed.

The Workflow Scheduler repeats step (1) to (3) until the End-Activity is reached.

We now take a look into how an Activity Handler handles the execution of an activity. After an Activity Handler has been created, it first posts the Before-Activity-Events if they are specified for the activity. The synchronous event is posted directly to the local ETR Server, to trigger the associated business rules. Asynchronous workflow event is posted to the Event Server so that the Event Server can distribute them to their subscribers. The Activity Handler then loads the activity code and executes it. The execution of the activity code is actually the execution of task items specified inside the

activity definition. The activity code posts external events to the Event Server/ETR Server if they are specified as task items. If there are e-service requests specified inside this activity and the performer selection constraint of the activity is ANY, the activity code contacts the Broker Proxy to bind the e-service requests to the services of some suitable providers in a constraint-based dynamic service binding process. To invoke an e-services request in the activity, the activity code first creates a Service Client. For each e-service request, a SOAP message containing the values of the e-service input attributes is generated and sent to the selected service provider through the Service Client to invoke the e-service. A SOAP message containing the response information is returned to the activity code after the e-service is performed. After the execution of the activity code, the result is returned to the Activity Handler. The Activity handler then posts the After-Activity-Events and passes the output data of the activity to the Workflow Scheduler (or the Block Scheduler if the activity is inside a block, or to the Subflow Scheduler if the activity is in a subflow). Please refer to Figure 5-9 for details about activity codes.

From the above illustration, we can see that to achieve dynamic properties, the Workflow Engine needs to work together with other servers in the ISEE Hub when it executes workflow instances. The run-time interactions between the Workflow Engine and other components of DynaFlow are shown in Figure 5-12.

5.2.4.3 Run-time modification to a business process model

The Workflow Engine provides APIs for run-time modifications of a process model to alter its course of execution. The run-time modification can be accomplished by modifying the run-time workflow structures. The Workflow Engine supports the following modifications.

(1) Delete/add a transition. To modify the control flow of a process model at the run-time, the Workflow Engine provides APIs to delete/add a transition from/to the model during the execution of a workflow instance. The modification is accomplished by deleting/adding the transition information from/to the corresponding control flow structure. If the deleted transition is the only transition in the control flow structure, the control flow structure will then be deleted from the run-time workflow structures of the process model.

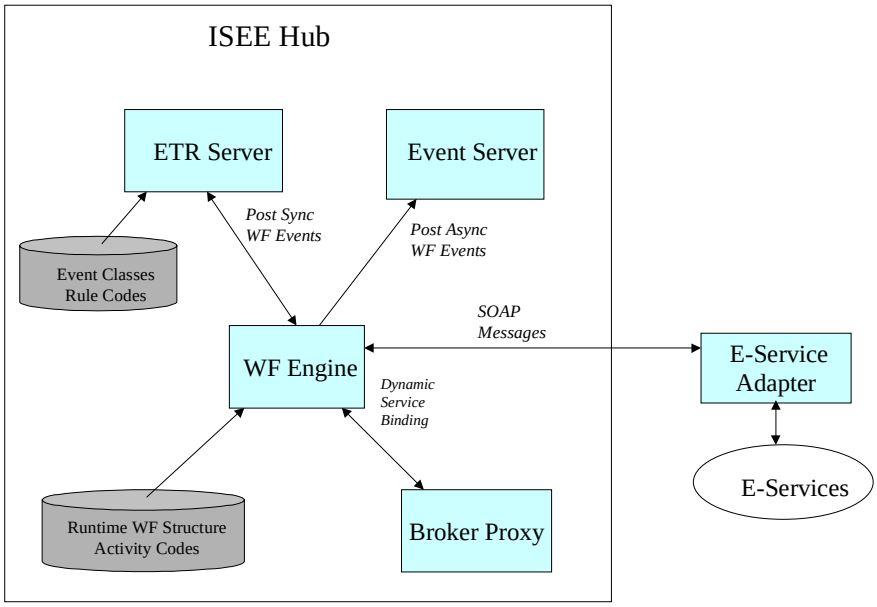


Figure 5-12: Runtime interactions between the Workflow Engine and other servers.

(2) Delete/add a data flow. To do a run-time modification to the data flow of a process model, the Workflow Engine provides APIs to delete/add a data flow from/to the model during the execution of a workflow instance. The modification is accomplished by deleting/adding the corresponding data flow structure from/to the run-time workflow structures of the process model.

(3) Replace an activity. Some business policies are embedded inside an activity body. When there is an unexpected change to these policies during the execution of a workflow instance, the running workflow instance may need to replace the original activity with an alternative one. Because, in our implementation, the activity code is dynamically loaded when an activity is scheduled, we can achieve this by modifying the name of the original activity in the control structure. Thus, the workflow instance will load the alternative activity instead of the original one.

(4) Modify condition. Similar to the modification of activities, the need for modifying conditions also exists. In our implementation, the condition of a transition is evaluated in an interpretive fashion. We can just call the API provided by the Workflow Engine to modify a condition directly.

CHAPTER 6
SUMMARY AND FUTURE WORK

6.1 Summary

In summary, the research presented in this dissertation aims to develop a dynamic workflow model and a workflow management system for modeling and controlling the execution of inter-organizational business processes of a virtual enterprise.

We have designed a Dynamic Workflow Model (DWM) by extending the underlying model of WfMC's WPDL. The extension includes adding the concepts of events and rules into WPDL, introducing the connector construct, encapsulating activity definitions, and supporting e-service requests. The DWM and thus the dynamic workflow management system using DWM as the underlying model are active, flexible, adaptive, and customizable.

A constraint-based, dynamic service binding mechanism has been introduced in our work for the enactment of business processes. The e-service requests in a process model are specified according to the standardized e-service templates. Constraints can be defined for these e-service requests. The e-service requests in the dynamic business process models are bound to the services of specific service providers at run-time by using the services of the Broker Server, which supports constraint-based brokering and service provider selection.

We have also designed and implemented a dynamic workflow management system DynaFlow, which consists of a Workflow Server, an ETR Server, an Event

Server, and a Broker Proxy. The Workflow Engine of the Workflow Server handles the execution of workflow instances. It makes use of the e-services provided by other servers to archive the following dynamic properties: 1) dynamic binding of e-service requests to e-services, 2) notifications of the enactment milestones of a business process through the posting of asynchronous workflow events, 3) customization of inter-organizational process models to meet individual organizations' local needs, 4) customization of workflow instances, and 5) run-time modification of workflow specifications. The code generation approach described in this paper enables the efficient and effective implementation of the above dynamic properties.

6.2 Future Work

There are several issues that can be addressed in the future research. First, there are security issues related to the dynamic workflow management system. The security issues can be seen from three different aspects: process models, workflow events, and e-services. In our system, we assume that process models are designed for a virtual enterprise. A security system needs to be provided for the virtual enterprise to ensure that only qualified organizations and their users can make use of the process models. We also assume general users can register for the asynchronous workflow events and get notified when these events are posted. Since these events carry the process enactment data (input data or output data of an activity), and some data may be sensitive, the system should have a facility to restrict access to the asynchronous workflow events carrying sensitive data. Finally, service providers may have their own security policies to access the e-services they provide. These policies need to be considered during the dynamic service binding process.

Second, in this work, we provide a mechanism to support run-time modifications to a process model. In the current system, we assume that the process model designer takes care of the correctness of a process model. The validation of the "correctness" of a process model can be a challenging research issue.

Third, to achieve dynamic service binding, we assume that standardized e-service templates are jointly defined by business organizations of the same business type. Service providers register their e-services according to e-service templates. We further assume that a process model designer would define e-service requests based on the same standardized e-service templates. The above assumptions may not be realistic because different organizations and users may use different ontologies. They have different interpretations and understandings of concepts (or terms) and their semantic relationships. Also, for the same e-service, they may have different numbers of attributes, and the order of the attributes can also be different. This makes the standardization of e-service templates very difficult; so is the dynamic service binding. Some mechanism for mapping different ontologies will be needed.

Fourth, a business organization can use business process models, which integrate available e-services provided by different organizations, to create new e-services. Thus, from the e-service point of view, constructing a business process can also be seen as the composition of a new e-service. To provide a comprehensive e-service composition infrastructure, more research on e-service representation, e-service registration, e-service discovery, and e-service request binding is essential. In our current implementation, we use a simulated constraint-based broker proxy. The implementation of the constraint-

based broker is an ongoing effort and will enable our dynamic workflow system to perform dynamic e-service compositions.

APPENDIX
BNF FOR EXTENDED WPDL

// Workflow Process Definition

<Workflow Process Definition>::=
WORKFLOW <process id>
    <workflow process definition header>
    [IN_PARAMETERS <parameter list>]
    [OUT_PARAMETERS <parameter list>]
    [<access restriction part>]
    <Begin Activity>
    <Activity List>
    [<Subflow List>]
    [<Connector List>]
    <Transition Information List>
    [<Event List>]
    [<Rule List>]
    [<Data Class List>]
    [<Data Flow List>]
    <End Activity>
    [<Block List>]
END_WORKFLOW

<workflow process definition header>::=
    [CREATED   <creation date>]
    [NAME   <name>]
    [DESCRIPTION   <description>]
    [PRIORITY   <priority>]
    [CLASSIFICATION <business type name>]
    [<time estimation>]
    [DOCUMENTATION <documentation>]

// Begin Activity

BEGINACTIVITY   <process/block id>
    [DESCRIPTION <description>]
    IN_PARAMETERS <input parameters>
    [LOOP   <loop condition>]
END_BEGINACTIVITY

// End Activity

ENDACTIVITY <process/block id>
[DESCRIPTION <description>]
OUT_PARAMETERS <input parameters>
[LOOP <loop condition>]
END_ENDACTIVITY

//Activity List

<Activity List>::=
ACTIVITY <activity id>
[NAME <name>]
[DESCRIPTION <description>]
<activity kind information>
END_ACTIVITY
[<Activity List>]

<activity kind information>::=
[IN_PARAMETERS <parameter list>]
[OUT_PARAMETERS <parameter list>]
[PERFORMER < participant assignment >]
[ACTIVITY_VAR] <variable definition list>
[WF_EVENTS] <workflow event list>
[DOCUMENTATION <documentation>]
IMPLEMENTATION
<task list>
END_IMPLEMENTATION

<task list>::=<task invocation> [< task list>]

<task invocation>::=<generic task>
[DESCRIPTION <description>]

<generic task>::=<e-service request>
| <external event posting>
| <inline code>

<e-service request>::=
ESERVICE <e-service name> <DOT> <operation name>
REQUEST <in_attributes mapping>
RESPONSE <out_attributes mapping>
CONSTRAINT <constraint definition>
END_ESERVICE

<participant assignment>::=<business type name>
<LPR> <performer selection constriant> <RPR>

<performer selection constraint>::=ANY
| CONSTANT <organization name>
| SAME_AS<activity id>
| VARIABLE
<activity id> <DOT> <output parameter>

// Subflow List

<Subflow List>::=
SUBFLOW <subflow id> <referenced process id>
[DESCRIPTION <description>]
[<Subflow List>]

//Connector List

<Connector List> ::=
CONNECTOR <Connector id>
[NAME <name>]
[DESCRIPTION <description>]
<connector kind information>
END_CONNECTOR
[<Connector List>]

<connector kind information> ::=AGGREGATION <aggregation kind>

<aggregation kind>::=JOIN <aggregation operator> <transition list>
| SPLIT <aggregation operator> <transition list>

<aggregation operator>::=AND | OR | XOR

<transition list>::=<transition id> [<transition list>]

// Block List

<Block List>::=<Block> [<Block List>]

<Block>::=
BLOCK <block id>
[NAME <name>]
[DESCRIPTION <description>]
[DOCUMENTATION <documentation>]
<Begin Activity>
<Activity List>
[<Connector List>]
<Transition Information List>

```
            [<Event List>]
            [<Rule List>]
            [<Data Class List>]
            [<Data Flow List>]
            <End Activity>
END_BLOCK

// Transition Information List

<Transition Information List>::=
TRANSITION   <transition id>
            [NAME            <name>]
            [DESCRIPTION   <description>]
            <transition kind description>
            [DATA_MAPPING <data mapping info>]
END_TRANSITION
[<Transition Information List>]

<transition kind description>::=FROM <transition-entity id> TO <transition-entity id>
                                    [CONDITION  <transition condition>]

<transition-entity id>::=<data entity id>
                        | <connector id>
                        | <subflow id>
                        | <block id>

<data mapping info>:: =<LPR> <parameter mapping list> <RPR>

<parameter mapping list>::= <source parameter name>::<target parameter name>
                                [<SCOLON> <parameter mapping list>]

<transition condition>:: = <term> <op> <term>

<term> ::=<data entity id> <DOT> <output parameter name>
            | <value>

<data-entity id>::= <activity id>
                    | <subflow id>
                    | <block id>

<op> ::= EQUAL
            | LESS_THAN
            | GREATER_THAN
            | GREATER_EQUAL
            | LESS_EQUAL
// Data Flow List
```

< Data Flow List> ::=
DATAFLOW <data flow id>
[DESCRIPTION <description>]
FROM <dataflow-entity id> TO <dataflow-entity id>
<data mapping info>
END_DATAFLOW
[<Data Flow List>]

// Event List

<Event List>::=
EVENT <event name>
[DESCRIPTION <description>]
[<Event List>]

// Data Classes

<Data Class List>::=
DATACLASS <data class name>
[DESCRIPTION <description>]
[<Data Class List>]

REFERENCES

[Ada98] Adam, N., Adiwijaya, I., and Atluri, V., “EDI through a Distributed Information Systems Approach,” Proceedings of the 31st Hawaii International Conference on System Sciences, Kohala Coast, Hawaii, USA, January 1998, http://www.computer.org/proceedings/hicss/8233/8233toc.htm, Accessed 12/31/2001.

[Alo95] Alonso, G., Agrawal, D., El Abbadi, A., Mohan, A., Kamath, M., and Guenthoer, R., “Exotica/FMQM: A Persistent Message-Based Architecture for Distributed Workflow Management,” Proceedings of IFIP Working Conference on Information Systems Development for Decentralized Organizations, Trondheim, Norway, August 1995, pp. 1-18.

[Alo99] Alonso, G., Fiedler, U., Hagen, C., Lazcano, A., Schuldt, H., and Weiler, N. “WISE – Business to Business E-Commerce,” Proceedings of the 9th International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises, Sydney, Australia, March 1999, http://www.computer.org/proceedings/ride/0119/0119toc.htm, Accessed 12/31/2001.

[Alo97] Alonso, G. and Mohan, C., “Workflow Management Systems: The Next Generation of Distributed Procesing Tools," *Advanced Transaction Models and Architectures*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1997, pp. 35-62.

[Ari00] Ariba Corporation, IBM Corporation, and Microsoft Corporation, ”UDDI Technical White Paper,” September 2000, http://www.uddi.org, Accessed 12/18/2001.

[Box00] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, Nielsen, H.F., Thatte, S., and Winer, D., “Simple Object Access Protocol (SOAP) 1.1,” May 2000, http://www.w3.org/TR/SOAP, Accessed 12/18/2001.

[Cas96] Casati, F., Grefen, P., Pernici, B., Pozzi, G., and Sanchez, G. “WIDE Workflow Model and Architecture,” Technical Report, University of Twente, 1996, http://citeseer.nj.nec.com/casati96wide.html, Accessed 12/18/2001.

[Cer97] Ceri, S., Grefen, P., and Sanchez, G., "WIDE--A Distributed Architecture for Workflow Management," Proceedings of the 7th International Workshop on Research Issues in Data Engineering, Birmingham, UK, April 1997, http://www.computer.org/proceedings/ride/7849/7849toc.htm, Accessed 12/31/2001.

[Chr01] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1" W3C Note, March 2001, http://www.w3.org/TR/WSDL.html, Accessed 12/18/2001.

[Cic98] Cichocki, A., *Migrating Workflows and Their Transactional Properties*, Doctoral Dissertation, Department of Computer Science, University of Houston, 1998.

[Cro00] CrossFlow Consortium, "Flexible Change Control," CrossFlow Deliverable: D8.a, February 2000, http://www.crossflow.org/, Accessed 12/18/2001.

[Dav99] Davulcu, H., Kifer, M., Pokorny, L.R., Ramakrishnan, C.R., Ramakrishnan, I.V., and Dawson, S., "Modeling and Analysis of Interactions in Virtual Enterprises," Proceedings of the 9th International Workshop on Research Issues on Data Engineering, Information Technology for Virtual Enterprises, Sydney, Australia, March 1999, http://www.computer.org/proceedings/ride-/0119/0119toc.htm, Accessed 12/31/2001.

[Ell95] Ellis, C., Keddara, K., and Rozenberg, G., "Dynamic Change within Workflow Systems," Proceedings of the Conference on Organizational Computing Systems, Milpitas, California, USA, October 1995, pp. 10-21.

[Eme90] Emerson, A., "Temporal and Model Logic", *Handbook of Theoretical Computer Science (Vol. B)*, Amsterdam, The Netherlands, 1990.

[Geo95] Geogakopoulos, D., Hornick, M, and Sheth, A. "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure," Distributed and Parallel Databases, Vol. 3, No. 2, April 1995, pp. 119-153.

[Gep98] Geppert, A. and Tombros, D. "Event-based Distributed Workflow Execution with EVE," IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing (Middleware '98), The Lake District, England, September 1998, pp. 427-444.

[Gre99] Grefen, P. and Hoffner, Y., "CrossFlow - Cross-Organizational Workflow Support for Virtual Organizations," Proceedings of the 9th International Workshop on Research Issues on Data Engineering: Information Technology for Virtual Enterprises, Sydney, Australia, March 1999, http://www.computer-.org/proceedings/ride/0119/0119toc.htm, Accessed 12/31/2001.

[Han98] Han, Y. and Sheth. A., “A Taxinomy of Adaptive Workflow Management,” Proceedings of the Conference on Computer-Supported Cooperative Work, Seattle, Washington, USA, November 1998, http://ccs.mit.edu/klein/cscw98/, Accessed 12/31/2001.

[Hel02] Helal, A., Su, S.Y.W., Meng, J., Krithivasan, R., and Jagatheesam, A., “The Internet Enterprise,” to appear in the Proceedings of the 2nd IEEE/JPSJ Symposium on Applications and the Internet (SAINT02), Nara, Japan, January 2002, http://www.harris.cise.ufl.edu/projects/e-services.htm, Accessed 12/31/2001.

[Hel01] Helal, A., Wang, M., Jagatheesan, A. and Krithivasan, R., "Brokering Based SelfOrganizing E-Service Communities," Proceedings of the Fifth International Symposium on Autonomous Decentralized Systems (ISADS) With an Emphasis on Electronic Commerce, Dallas, Texas, USA, March 2001, pp. 349-356.

[Hua00] Chunbo Huang, *A Replicable Web-Based Automated Negotiation Server for Electronic Commerce Systems*, Doctoral Dissertation, Department of Computer and Information Science and Engineering, University of Florida, 2000. http://www.cise.ufl.edu/tech-reports/tech-reports/tr99-abstracts.shtml, TR 010, Accessed 12/18/2001.

[IBM00] IBM, MQSeries Workflow, August 2000, http://www-4.ibm.com/ software/ts/mqseries/workflow/, Accessed 12/18/2001.

[Kli98] Klingemann, J., Wäsch, J., and Aberer, K., “Adaptive Outsourcing in Cross-Organizational Workflows,” GMD Report, August 1998, http://www.gmd.de/publications/report/0030/, Accessed 12/18/2001.

[Kri01] Krithivasan, R. and Helal, A., "BizBuilder - An e-Services Framework Targeted for Internet Workflow," Proceedings of the 3rd Workshop on Technologies for E-Services, Springer Lecture Notes in Computer Science series, in conjunction with VLDB 2001, Rome, Italy, September 2001, pp. 89-102.

[Kim00] Kim, Y., Kang, S.H., and Kim, D., “WW-FLOW: Web-Based Workflow Management with Runtime Encapsulation,” IEEE Internet Computing, Vol. 4, No. 3, May-June 2000, pp. 55-64.

[Laz01] Lazcano, A., Schuldt, H., Alonso, G., and Schek, H., “WISE: Process based E-Commerce,” IEEE Data Engineering Bulletin, Special Issue on Infrastructure for Advanced E-Services, Vol. 24, No. 1, March 2001, pp. 46-51.

[Lee00] Lee, M., *Event and Rule Services for Achieving a Web-based Knowledge Network*, Doctoral Dissertation, Department of Computer and Information Science and Engineering, University of Florida, 2000. http://www.cise.ufl.edu/tech-reports/tech-reports/tr00-abstracts.shtml, TR 002, Accessed 12/18/2001.

[Lee01] Lee, M., Su, S.Y.W., and Lam, H., "A Web-based Knowledge Network for Supporting Emerging Internet Applications," WWW Journal, Vol. 4, No. 1/2, 2001, pp. 121-140.

[Len01] Lenz, K. and Oberweis, A. “Modeling Interorganizational Workflows with XML Nets,” in Proceedings of 4th Annual Hawaii International Conference on System Sciences, Maui, Hawaii, USA, January 2001, http://www.computer.org/proceedings/hicss/0981/0981toc.htm, Accessed 12/31/2001.

[Ley01] Leymann, F., “Web Services Flow Language,” May 2001, www-4.ibm.com/software/solutions/webservices/pdf/WSFL.pdf, Accessed 12/18/2001.

[Ley94] Leymann, F. and Roller, D., “Business Process Management with Flowmark,” Proceedings of the 39th IEEE Computer Society International Conference, California, USA, February 1994.

[Men00] Meng, J., Helal. A., and Su, S.Y.W., “An Ad-Hoc Workflow System Architecture Based on Mobile Agents and Rule-Based Processing,” Proceedings of The International Conference on Artificial Intelligence, Nevada, USA, June 2000, pp. 245-251.

[Men02] Meng, J., Su, S.Y.W., Lam, H., and Helal, A., “Achieving Dynamic Inter-Organizational Workflow Management by Integrating Business Processes, Events, and Rules,” to appear in the Proceedings of the 35th Hawaii International Conference on System Sciences (HICSS35), Hawaii, USA, January 2002, http://www.harris.cise.ufl.edu/projects/e-services.htm, Accessed 12/31/2001.

[Mil96] Miller, J., Sheth, A., Kochut, K., and Wang, X., "CORBA-Based Run-Time Architectures for Workflow Management Systems," Journal of Database Management, Special Issue on Multidatabases, Vol. 7, No. 1, 1996, pp. 16-27.

[Mul99] Muller, R. and Rahm, E., “Rule-based Dynamic Modification of Workflows in a Medical Domain,” Proceedings of BTW99, Freiburg, Germany, February 1999, pp. 429-448.

[Mur89] Murata, T., “Petri Nets: Properties, Analysis and Applications,” Proceedings of IEEE, Vol.77, No.4, April 1989, pp. 541-580.

[Mut98] Muth, P., Wodtke, D., Weisenfels, J., Dittrich, A. K., and Weikum, G., "From Centralized Workflow Specification to Distributed Workflow Execution," Journal of Intelligent Information Systems, Special Issue on Workflow Management, Vol. 10, No. 2, 1998, pp. 159-184.

[Oba01] Oba, M. and Komoda, N., “Multiple Type Workflow Model for Enterprise Application Integration,” Proceedings of the 34thHawaii International Conference on System Sciences, Hawaii, USA, January 2001, http://www.computer.org/proceedings/hicss/0981/0981toc.htm, Accessed 12/31/2001.

[Pel94] Peluso, E., Goldstine, J., Phoha, S., Sircar, S., Yukish, M., Licari, J.,and Mayk, I., “Hierarchical Supervision for the Command and Control of Interacting Automata,” Proceedings of the Symposium on Command and Control Research. Monterey, CA, USA, 1994, pp. 623-636.

[Phi00] Phillips, C. and Meeker, M., “The B2B Internet Report: Collaborative Commerce,” Morgan Stanley Dean Witter, April 2000, http://www.morganstanley.com/techresearch/b2b/b2bp1a.pdf, Accessed 12/18/2001.

[Rei98] Reichert, M. and Dadam, P. “Adept_flex-Supporting Dynamic Changes of Workflows Without Losing Control,” Journal of Intelligent Information Systems, Special issue on Workflow and Process Management, Vol. 10, No. 2, March 1998, pp. 93-129.

[She99] Sheth, A., Aalst, and W., Arpinar, I., “Process Driving the Networked Economy,” IEEE Concurrency, Vol. 7, No. 3, July-September 1999, pp. 18-31.

[She96] Sheth, A., Georgakopoulos, D., Joosten, S., Rusinkiewicz, M., Scacchi, W., Wileden, J., and Wolf, A., "Report from the NSF Workshop on Workflow and Process Automation in Information Systems," Sigmod Record, Vol. 25, No. 4, December 1996, pp. 55-67.

[She97] Sheth, A. and Kochut, K.J., “Workflow Applications to Research Agenda: Scalable and Dynamic Work Co-ordination and Collaborative Systems,” in Advances in Workflow Management Systems and Interoperability, NATO Advanced Study Institute, Istanbul, Turkey, August 1997, http://citeseer.nj.nec.com/sheth97workflow.html, Accessed 12/31/2001.

[Str00] Stricker, C., Riboni, S., Kradolfer, M., and Taylor, J., “Market-based Workflow Management for Supply Chains of Services,” Proceedings of the 33rd Hawaii International Conference on System Sciences, Maui, Hawaii, January 2000, http://www.computer.org/proceedings/hicss/0493/0493toc.htm, Accessed 12/31/2001.

[Su01a] Su, S. Y.W., Huang, C., Hammer J., Huang, Y., Li, H., Wang, L., Liu Y., Pluempitiwiriyawej, C., Lee, M., and Lam, H., “An Internet-based Negotiation Server for E-commerce,” The VLDB Journal, Vol. 10, No. 1, 2001, pp.72-90.

[Su00] Su, S.Y. W. and Lam, H., “IKnet: Scalable Infrastructure for Achieving Internet-based Knowledge Network,” invited paper, Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet, l’Aquila, Rome, Italy, July 2000, http://www.ssgrr.it/en/ssgrr2000/proceedings.htm, Accessed 12/31/2001, pp. 2-13.

[Su01b] Su, S. Y.W., Lam, H., Lee, M., Bai, S., and Shen, Z., "An Information Infrastructure and E-services for Supporting Internet-based Scalable E-business Enterprises," Proceedings of the 5th International Enterprise Distributed Object Computing Conference, Seattle, Washington, USA, September 2001.

[VIT01] VITRIA Corporation, “BusinessWare Overview,” 2001, http://www.vitria.com/products/businessware/overview.html, Accessed 12/18/2001.

[WfM95] WfMC, “The Workflow Reference Model,” January 1995, http://www.wfmc.org, Accessed 12/18/2001.

[WfM99a] WfMC, “Interface1: Process Definition Interchange V 1.1 Final (WfMC-TC-1016-P)," October 1999, http://www.wfmc.org, Accessed 12/18/2001.

[WfM99b] WfMC, “Terminology and Glossary,” February 1999, http://www.wfmc.org, Accessed 12/18/2001.

[Win87] Winograd, T. and Flores, F., *Understanding Computers and Cognition*, Addison-Wesley, Boston, Massachusetts, USA, 1987.

BIOGRAPHICAL SKETCH

Jie Meng was born in Longyan, Fujian Province, P. R. China, in 1969. She received her Bachelor of Engineering degree in computer science and technology from the Tsinghua University, Beijing, China, in July 1991. She then worked in the Computer Information Center of the Tsinghua University for two years and did research on object-oriented databases. She received her Master of Engineering degree in computer software from the Tsinghua University in July 1995. After that, she worked as a researcher and an instructor for three years in the Computer Science and Technology Department, Tsinghua University. From 1998 to present, Jie Meng has been a Ph.D. Student in the Computer and Information Science and Engineering Department at the University of Florida. She also works as a research assistant in the Database Systems Research and Development Center under the guidance of Professor Stanley Y. W. Su. Jie Meng's primary research interests include workflow systems, e-service composition, object-oriented databases, heterogeneous system integration, and electronic commerce/business systems.