Modeling and Coordination in Interorganizational Workflow

Donghui Lin
Abstract

The goal of this thesis is to develop a framework to enable interorganizational workflow collaboration, considering the requirements of workflow design and support of different knowledge backgrounds of organizations.

With the global expansion of Internet and distributed computing environments, computer mediated collaboration has been rapidly increasing among different organizations. In these collaboration environments, organizations cooperate with one another with resources that each owns to achieve some common goals. Collaboration processes that involve multiple organizations can be modeled as interorganizational workflows, with challenges in some aspects. First, different organizations have different cultural and knowledge backgrounds, which might have great impact on the way that the collaboration is conducted. Therefore, when considering interorganizational workflow, many problems occur due to disagreements on views regarding the collaboration processes. Moreover, though cooperation of organizations needs some visible interactions and data sharing, it is also important for organizations to consider the requirements of workflow design as much as possible during the collaboration, such as privacy preservation, flexibility support of local processes, workflow reuse and so on.

In this thesis, with respect to the above points, we address following issues on interorganizational workflow collaboration framework including modeling, coordination, execution and implementation approaches.

1. Interorganizational Workflow Modeling and Coordination

   In interorganizational environments, due to different cultural back-
grounds, participants from different organizations always have different views of modeling workflow process. Moreover, organizations require preserving privacy and autonomy of its own workflow from other organizations. Therefore, workflow inter-visibility is expected to be as little as the collaboration need. Here, the following two sub issues are tackled.

- **Modeling interorganizational workflow based on local process views**
  Interorganizational workflow is modeled for representing and managing collaboration among organizations in this research. Based on the study of real collaboration cases, several problems should be considered in the modeling phase, such as how to represent different process views of different organizations, how to preserve organizational privacy and autonomy and so on. Therefore, in this research, instead of sharing a global pre-defined workflow process view together, each organization has a local process view by its own understanding of the whole collaboration process.

- **Coordinating interorganizational workflow using compatibility analysis mechanisms**
  In the proposed interorganizational workflow model, each organization has a local process view based on its own consideration. Therefore there might be conflicts between local process views of different organizations. In this issue, we propose the coordination approach for different local process views to detect incompatibilities among organizations using the compatibility analysis mechanisms. Further, the proposed compatibility analysis mechanism is illustrated with a case study of collaborative software development.

2. Designing Interorganizational Workflow Execution Mechanisms
Flexibility, adaptation and distribution have been regarded as major challenges of modern interorganizational workflow. To address these challenges, this research proposes an interorganizational workflow execution approach based on process agents and ECA (event-condition-action) rules. In the proposed approach, an interorganizational workflow is modeled as a multiagent system with one process agent for each organization. The whole execution mechanism is divided into two parts: the intra-execution, which refers to execution within the same organizations, and the inter-execution, which represents interaction among organizations. For intra-execution, we use the method of transforming the graph-based local workflow into block-based workflow to design general ECA rules. ECA rules and process agents are used to control state transitions of tasks and process execution in the local workflows. Inter-execution is realized by process agent interaction protocols. The proposed approach can provide flexible execution of interorganizational workflow with distributed organizational autonomy and adaptation. A case study of collaborative software development is illustrated for the proposed approach.

3. Implementation of Interorganizational Workflow Collaboration

Globalization and advanced information technologies cause the increasing development of collaborative business. Flexibility support and privacy preservation are important issues that should be considered in business process management across organizational boundaries. To address these issues, this research proposes an interorganizational workflow collaboration approach to realize collaborative business based on local process views. In the interorganizational workflow model, since each organization has its own local process view, it is necessary to develop the approach of interorganizational workflow collaboration from coordinating local process views of different organizations to implementing the whole model. To realize this process, we divide the collaboration process into two phases: bottom-up coordination and top-down implementation. We illustrate the effec-
tiveness of the proposed approach by a case of collaborative software development. The approach in this research is effective in aspects of workflow flexibility and privacy preservation. Moreover, it can be implemented by extending current commercial business process management systems.
Acknowledgments

Three years of study and research at Kyoto University has been a wonderful experience for me. I would not have been able to complete this journey without the aid and support of countless people, to only some of whom it is possible to give particular mention here.

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Toru Ishida for providing me this valuable opportunity to come to Japan and work in his laboratory. None of the work in this thesis could have been carried out without his remarkable guidance, professional advice and constant encouragement. During the past three years, he has impressed me deeply with his professionalism and enthusiasm as a researcher, professor and organizer. Also, I have learned a lot from him about critical and logical thinking as well as technical writing.

I gratefully appreciate my advisory committee members at Kyoto University, Professor Hajime Kita and Professor Masatoshi Yoshikawa for their constructive suggestions and valuable comments. I am also indebted to my other three vice advisors, Professor Makoto Yokoo at Kyushu University, Professor Huanye Sheng at Shanghai Jiao Tong University and Associate Professor Mizuho Iwaihara at Kyoto University for their insightful advice and helpful discussion on multiagent systems, workflow formalism and application issues.

I wish to express my thanks to faculty members at Ishida & Matsubara Laboratory for their support in numerous ways. Associate Professor Shigeo Matsubara always gave me kind advice and helpful comments in my research. Assistant Professor Hiromitsu Hattori supplied encouragement
whenever I have difficulties in my research. Ms. Yoko Kubota and Ms. Terumi Kosugi were always ready to provide their reliable guidance and kind help on administrative affairs through the academic process.

Professor Yichuan Jiang at Southeast University, China always gave me valuable suggestions on my research and encouraged me continuously by sharing his successful experiences with me. Here I would like to take this opportunity to deliver my thanks to him for his help and wisdom.

I am grateful to Dr. Ahlem Ben Hassine at Language Grid Project, National Institute of Information and Communications Technology (NICT) for her frequent discussion and help on how to develop my research issues and polish journal papers. Also, I wish to thank the software development expert, Mr. Takao Nakaguchi for his amazing technical instructions during my stay at NICT in the summer of 2006.

Sincere gratitude goes to all the members and alumni at Ishida & Matsubara Laboratory for their kindness and help to me, a student from foreign lands. I would like to express my special thanks to my three fellow PhD students, Heeryon Cho, Masahiro Tanaka and Yuu Nakajima, for fighting all the way together with me during the past three years. Their hard-working, kindness and research ability impressed me greatly and I feel so lucky to have them all.

I must thank my good friend Chen Wang at Microsoft Corporation, USA for his continuous emotional support. He also spent much time for proof reading my papers and all the chapters in this thesis.

I would like to express my cordial gratitude to my good friends in China: Liang Chen, Mengyuan Chen, Chaobin Huang, Hua Liu, Zhiqiang Liu, Jun Lu, Jinyuan Pan, Cheng Qian, Quan Wang, Lingsheng Wu, Jinbin Xie, Yan-guang Yang, Yousheng Ye, Dandan Yuan and many others. I could hardly overcome all those difficulties and there would have been less joy and happiness in my life without them. I also want to thank my Chinese friends who are, or have been, in Kyoto University together with me: Yan Li, Yuntian Lu, Yanchen Wu, Yinglan Wu, Qin Zhang and Tiejun Zhao.

Special thanks are due to Ms. Kanae Shibata for giving me constant encouragement and treating me as a family member during my stay in Japan.
Finally, I want to express my gratitude to my parents, Zhuxiang Lin and Meizhen Sun for their enduring understanding, patience and encouragement when it was most required.

This research was partially supported by Grant-in-Aid for Scientific Research (A) (18200009, 2006-2008) from Japan Society for the Promotion of Science (JSPS). My stay in Kyoto University was supported by Japanese Government (Monbukagakusho) Scholarship for Research Student.
# Contents

1 Introduction ........................................... 1
   1.1 Objectives ......................................... 1
   1.2 Approaches and Issues ............................ 3
   1.3 Thesis Outline .................................... 5

2 Background ............................................ 9
   2.1 Workflow Management ............................. 9
   2.2 Interorganizational Workflow ................... 10
      2.2.1 Workflow Interoperability ................... 11
      2.2.2 Requirements .................................. 12
      2.2.3 Interorganizational Workflow Approaches . 13
   2.3 Motivation Example ............................... 18

3 Interorganizational Workflow Based on Local Process Views 21
   3.1 Introduction ....................................... 21
   3.2 Modeling Interorganizational Workflow ........... 23
      3.2.1 Overview of the Model ....................... 23
      3.2.2 Local Process Views ......................... 24
   3.3 Collaborative Software Development Scenario .... 32
   3.4 Discussion ........................................ 34
   3.5 Summary .......................................... 37

4 Coordination of Local Process Views 39
   4.1 Introduction ....................................... 39
6.6 Summary ........................................ 98

7 Conclusion ........................................ 101
   7.1 Contributions .................................. 101
   7.2 Future Directions ................................ 104

Bibliography ....................................... 107

Publications ....................................... 117
## List of Figures

3.1 Interorganizational workflow based on local process views . 24  
3.2 An example of local process view ..................... 26  
3.3 An example of interaction sequence chart .............. 31  
3.4 Local workflow process abstraction using subprocesses . 32  
3.5 Local process views in collaborative software development . 33  

4.1 Phases of compatibility analysis ..................... 42  
4.2 Blocks and subprocesses in workflow process .......... 47  
4.3 Workflow process structure after transformation ........ 48  
4.4 Screenshot of the compatibility analysis tool .......... 57  
4.5 Local workflows and interaction protocol after coordination . 58  
4.6 Find partners using workflow service repository ....... 60  

5.1 Framework of interorganizational workflow execution .. 67  
5.2 ECA rules and workflow execution .................... 69  
5.3 Example of agent interaction protocols: (a) simple protocol; (b) protocols with coordination and negotiation ........... 78  
5.4 Effects of process agent interaction to intra-execution .... 79  

6.1 Global function view of collaborative software development .... 91  
6.2 Design of common interaction part among organizations 93  
6.3 Top level main process of Alpha team .................. 95  
6.4 Parts of a second level process (Development Analysis) . .... 95
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Comparison of models with the perspective of workflow</td>
<td>35</td>
</tr>
<tr>
<td>3.2</td>
<td>Comparison of models with the perspective of organization</td>
<td>37</td>
</tr>
<tr>
<td>5.1</td>
<td>External events and ECA rules</td>
<td>72</td>
</tr>
<tr>
<td>5.2</td>
<td>Internal ECA rules for blocks (1)</td>
<td>74</td>
</tr>
<tr>
<td>5.3</td>
<td>Internal ECA rules for blocks (2)</td>
<td>75</td>
</tr>
<tr>
<td>5.4</td>
<td>Interorganizational workflow execution example</td>
<td>81</td>
</tr>
<tr>
<td>6.1</td>
<td>Example of access rights for users and groups</td>
<td>97</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Objectives

In recent years, many companies have been experiencing great changes in business environments, which have created the necessity of information technology to ensure efficient and effective process management. As a result, there have been numerous attempts to enhance information systems towards providing advanced functions of process management beyond simple manipulation of independent tasks. A workflow management system (WfMS) [Hollingsworth 95] then presents a new solution to the necessity of process management technology and tools.

With the global expansion of the Internet and distributed computing environments, computer-mediated collaborative work has been rapidly increasing within individual organizations, between different organizations, and even across nations. In the collaborative work, organizations cooperate with each other with the resources that each owns to achieve some common goals. The process of the cooperative works can be modeled as an interorganizational workflow, i.e., workflows crossing organizational boundaries inside company or between companies.

An interorganizational workflow is always composed of a set of workflow processes, where n business partners are involved in one global workflow process and each partner has its own local workflow process. Each
local workflow process is controlled by its corresponding business partner. However, these local workflow processes also need interaction because they depend on each other for the correct execution of processes. The global workflow process consists of local workflow processes and an interaction structure [Aalst 99b, Aalst 00].

There are several main challenges in interorganizational workflows. One fact that is always neglected in previous research of interorganizational workflow is that different organizations have different cultural and knowledge backgrounds, which might have great impact on the way that the collaboration is conducted [Hofstede 01]. Different backgrounds may produce conflicts during collaboration among organizations, which causes many problems due to disagreements on views regarding workflow process modeling. Therefore, support of multiple views and coordination efforts among different organizations should be considered when we model interorganizational workflow.

Another challenge for interorganizational workflow is how to satisfy requirements of workflow design, such as workflow flexibility, privacy preservation, workflow reuse, and so on. In collaborative business, organizations always try to preserve the privacy of their internal information as much as possible. Therefore, inter-visibility of local workflow processes should be as little as the asynchronous communication needs when modeling interorganizational workflow. Moreover, flexibility of local workflow and global workflow is extremely important in interorganizational workflow [Aalst 02a, Chebbi 06]. On one hand, interaction among organizations should be flexible so that local workflows can be managed autonomously and rarely affected by dynamic changes of asynchronous communication among organizations. On the other hand, local workflow processes should also be designed in a flexible way so as to adapt connections among organizations. Further, although the importance of interorganizational workflow is well recognized, few significant research outcomes of implementation methodologies have been reported.

Therefore, the objective of this thesis is to develop a framework to enable interorganizational workflow collaboration, considering the perspectives of
both workflow design requirements and organization requirements.

### 1.2 Approaches and Issues

In order to address issues on interorganizational workflow collaboration framework including modeling, coordination, execution and implementation, we take the following steps in this thesis.

First, we develop an interorganizational workflow model to support coordination among organizations. In an interorganizational workflow, participants from different organizations always have different views of modeling workflow process due to their different cultural backgrounds. Moreover, privacy preservation is an important consideration for each organization. Therefore, workflow inter-visibility is expected to be as little as collaborations need. Furthermore, flexibility support and workflow reuse should also be addressed. Here, the following two issues will be discussed.

1. **Modeling interorganizational workflow based on local process views**

   It is important to take different process modeling views of different organizations into consideration when modeling interorganizational workflow. In the real world, organizations seldom share a pre-defined global interorganizational workflow process. Instead, they have their own understanding of how to model the whole workflow process with their unique backgrounds and knowledge, which we call local process views of organizations. Moreover, to provide privacy preservation of organizations, local workflows are not expected to be shared among organizations. Therefore we propose a model, where each organization has a local process view based on its own understanding of the whole collaboration.

2. **Coordinating interorganizational workflow using compatibility analysis mechanisms**

   In the proposed interorganizational workflow model, different local process views should be compatible with each other for execution.
By conducting coordination with compatibility analysis of local process views of different organizations, incompatibilities are expected to be detected and eliminated before workflow execution, and therefore a compatible interorganizational workflow that supports multiple local process views can be established. Coordination is a process for organizations to achieve mutual understandings of interaction protocols and local workflows. While neglected in most of the previous researches, it is actually an extremely important step in interorganizational workflow management. In this issue, we propose an approach of coordinating different local process views, which is divided into two main phases: coordination of interaction protocols, and coordination of local workflows. Further, to validate the proposed model and the coordination approach of local process views, it is necessary to apply in real-world case with implementation. Therefore, in this research issue, we take collaborative software development workflow for instance and extend the research to applications in service-oriented environment.

Second, we design the execution mechanism for interorganizational workflow. There are several major problems in managing interorganizational workflow. For example, flexibility and adaptation are challenging issues, especially when the execution is across organizations. Moreover, existing workflow management systems (WFMS) do not effectively address issues related to environments distributed across organizations. A central and monolithic workflow engine as suggested by Workflow Management Coalition (WFMC) reference model is not sufficient to support the autonomy of enterprises. To address these issues, this research proposes a framework for interorganizational workflow execution based on ECA (event-condition-action) rules and process agents. The whole interorganizational workflow is modeled as a multiagent system with one process agent for each organization. Therefore, process agents of organizations preserve autonomy of organizations with flexibility. We design general ECA rules to make workflow execution automatic with adaptation for different organizations. The
whole execution is divided into two parts: the intra-execution, which means execution within a same organization, and the inter-execution, which represents interaction between organizations. We use a case study of collaborative software development to illustrate the proposed approach.

Finally, we develop an approach of interorganizational workflow collaboration which can be realized in business process management systems [Aalst 03d, Weske 04]. Most of the previous interorganizational workflow models are of limited impact on practical application due to the lack of formal methodological backgrounds and ignorance of proven collaboration frameworks. Moreover, when it comes to the collaborative business processes among organizations, flexible management and privacy preservation might be main issues that should be considered beyond the existing concepts for business process management. In the interorganizational workflow model, since each organization has its own local process view, it is necessary to develop the approach of interorganizational workflow collaboration from coordinating local process views of different organizations to implementing the whole model. To realize this process considering the above issues, we divide the collaboration process into two phases: bottom-up coordination and top-down implementation. Further, it is necessary to illustrate the effectiveness of the proposed interorganizational workflow collaboration approach. We use a case study of collaborative software development to show how the proposed approach can be realized in business process management systems.

1.3 Thesis Outline

This thesis consists of seven chapters, including this introduction chapter. Chapter 2 is dedicated to introduce interorganizational workflow as the background of this thesis. First, we show a summary of workflow management methodologies. Second, we investigate the research of interorganizational workflow: the requirements for interorganizational workflow and current approaches. We also discuss a motivation example that will be re-
Chapter 3 introduces a new modeling method for interorganizational workflow based on local process views. In this model, we take different workflow process modeling views of different organizations into consideration. That is, organizations do not share a pre-defined global interorganizational workflow process, but have their own understandings of how to model the whole workflow process with their unique backgrounds and knowledge. We first explain the overview of the model based on local process views with formal definitions. Then, we demonstrate the model by an example of software development collaboration. Further, in comparison with related work of interorganizational workflow modeling, the following features of our proposed model are shown: privacy preservation, flexibility support, workflow reuse, coordination efforts and multiple view support.

Chapter 4 presents a framework for organizations to coordinate with each other during the process of interorganizational workflow modeling. In the interorganizational workflow model proposed in Chapter 3, since different local process views should be compatible with each other for execution, it is necessary to coordinate different local process views to detect conflicts among organizations. To achieve this, we propose an approach of compatibility analysis for local process views of different organizations. During this process, incompatibilities are expected to be detected and eliminated before workflow execution. Further, we validate the proposed approach by a case study of software development collaboration with tools. We also discuss the extension of this research to applications in the service-oriented environment.

Chapter 5 describes a mechanism for interorganizational workflow execution management using agents and rules. To design the mechanism considering issues of flexibility, adaptation and distribution, an interorganizational workflow is modeled as a multiagent system with a process agent in each organization. We first divide the whole execution framework into intra-execution, the distributive execution within each organization, and inter-execution, the execution of interaction and coordination among organizations. Then, we explains the execution mechanisms in two parts: intra-
execution is controlled and monitored by process agents and ECA rules, where rules control internal state transitions and process agents are used to control the external state transitions of tasks; process agents interact with each other to fulfill the inter-execution by protocols. Further, a case study of execution management is provided to demonstrate the effectiveness of the proposed mechanism.

Chapter 6 shows an approach for interorganizational workflow collaboration which can be realized in business process management systems. In the interorganizational workflow model we propose in Chapter 3, since each organization has its own local process view, the approach of interorganizational workflow collaboration considers coordinating local process views of different organizations and implementing the whole model. To realize this process considering requirements such as workflow flexibility and privacy preservation, we divide the collaboration process into two phases: bottom-up coordination and top-down implementation. Further, we discuss the approach by implementing an example of software development collaboration process in a business process management system to illustrate the effectiveness.

Chapter 7 concludes the thesis by summarizing the results obtained through this research. We also address the prospect of the future research.
Chapter 2

Background

2.1 Workflow Management

The Workflow Management Coalition (WFMC) defines workflow management as follows: automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules. A workflow management system (WfMS) is defined as: a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications [Hollingsworth 95, Davenport 98]. Workflows are commonly understood as having certain key features, especially including the following aspects [Singh 99]. In consisting of a number of tasks, they are composite. They are structurally and semantically complex. Workflows are often cooperative, which involve not only human interaction but also back-and-forth interactions among their constituent activities. Workflows by their very nature arise in heterogeneous environments. The components of workflows may be of autonomous ownership and not fully under the control of the workflow.

Workflow management deals with modeling and controlling the execution of application processes in heterogeneous organizational and techni-
Several perspectives of workflow management have been proposed in previous research [Aalst 03a], including functional perspective, process perspective, organizational perspective, informational perspective, and operational perspective. The functional perspective is used to characterize activities that have to be performed during workflow execution, which also specifies how these activities are decomposed into smaller tasks. The process perspective mainly specifies execution conditions for tasks, including start conditions and execution order conditions. The process perspective and the functional perspective are commonly represented by workflow process definitions. In the organization perspective, typically the structure of an organization is defined by roles, groups and other artifacts clarifying organizational issues, such as responsibility and availability of persons. Information perspective covers data, partitioned in control data and production data. The information perspective assigns input and output parameters to single tasks and, hence, covers data dependencies between them. Operation perspective describes the elementary operations performed by resources and applications. Typically, these operations are used to create, read, or modify control and production data.

Several workflow modeling approaches have been reported in previous research [Geppert 98, Aalst 03a], including unified modeling language, object coordination nets, workflow graphs, workflow nets, workflow evolution and so on.

### 2.2 Interorganizational Workflow

In recent years, computer-mediated collaboration has been rapidly increasing within individual organizations, between different organizations, and even across nations. In such collaboration environment, organizations cooperate with one another with resources that each owns to achieve some common goals. The process of the cooperative works can be modeled as an interorganizational workflow, i.e., workflows crossing organizational boundaries inside an company or between companies. Interorganizational work-
flow has been more and more important in the research area of workflow management.

2.2.1 Workflow Interoperability

WfMC proposes the concept of workflow interoperability in building interorganizational workflow model [Anderson 99]. van der Aalst categorizes several types of workflow interoperability for interorganizational workflow, including capacity sharing, chained execution, subcontracting, case transfer, and loosely coupled workflows [Aalst 99b].

**Capacity sharing** The interoperability of capacity sharing assumes centralized control. The routing of the workflow is under the control of one workflow manager. The execution of tasks is distributed, i.e., resources of several business partners execute tasks. Each organization does not have information about the whole interorganizational workflow, and the workflow manager assigns tasks to the organizations.

**Chained execution** In this form of interoperability, the whole workflow process is split into a number of disjunctive subprocesses which are executed by different organizations in a sequential order. This requires an organization to transfer or initiate the flow for a case after completing all the work. Different from the form of capacity sharing, the control of the workflow in chained execution is distributed over organizations.

**Subcontracting** In the form of subcontracting, there is one organization which subcontracts subprocesses to other organizations. For the top-level organization, the subprocesses (subcontracted work) appear to be atomic. However, the subprocesses can be very complex for organizations that execute the subcontracted work. The control of the whole workflow is hierarchical. If the top-level organization is regarded as a root, then its subcontracting organizations are like its child nodes.
**Case transfer** In the type of case transfer, the workflow process specification is replicated and each organization has a copy of the workflow process description. However, at any time, each case resides at exactly one location. During execution, process instances can be transferred from one organization to another to balance the workload or simply because tasks are not implemented at all organizations. The interoperability can also be extended to support local variations, e.g., at a specific location the process may be extended with additional tasks.

**Loosely coupled** In loosely coupled interoperability, the whole workflow process is made up of several pieces which may be active in parallel over different organizations, where the definition of each subprocess is local. The local workflow of each organization keeps private to other organizations, and only the interaction protocol for communication among organizations is public.

Among above types of workflow interoperability, Capacity sharing uses centralized control while the others use decentralized control partially or completely. Also, capacity sharing is the only form of interoperability which does not require some partitioning of the workflow [Aalst 99b]. In this thesis, we mainly focus on the loosely coupled workflow interoperability with extension, which can satisfy most of modern collaborative business processes.

### 2.2.2 Requirements

When modeling interorganizational workflow, it is important to consider following requirements from the perspective of workflow management.

**Privacy preservation** The asynchronous communication in loosely coupled interorganizational workflows enables organizations to interact with each other. During the interaction process, organizations frequently share information and exchange data. However, in modern collaborative business, organizations always try to preserve the privacy of their internal information as much as possible. Therefore,
inter-visibility of local workflow processes should be as little as the asynchronous communication needs when modeling interorganizational workflow.

**Flexibility support** In an interorganizational workflow, the flexibility of local workflow and global workflow is an important requirement. On one hand, the interaction part among organizations should be flexible so that the local workflow processes of organizations could be managed autonomously and rarely affected by dynamic changes of asynchronous communication among organizations. On the other hand, local workflow processes should also be designed in a flexible way so as to adapt connections among organizations.

**Workflow reuse** In the service-oriented environment, different organizations always compose their workflows to accomplish some common goals. However, it is costly for organizations to establish complicated workflows. In the context of collaborative business, it is efficient and economical for business partners to find some existing workflows from workflow repositories for coordination. Therefore, workflow reuse has been an important requirement for modern interorganizational workflow.

Besides, from the perspective of organization, support of different views for different organizations, and coordination efforts among organizations are also important issues we should address. In this research, we propose an interorganizational workflow model considering above requirements. Further, we develop mechanisms of how different organizations with different knowledge backgrounds coordinate with each other based on the proposed model.

### 2.2.3 Interorganizational Workflow Approaches

Previous approaches of interorganizational workflow include workflow cooperation and composition [Chebbi 06], workflow inheritance [Aalst 99b,
In the approach of workflow cooperation and composition, each organization always has its own local workflow process. Interorganizational workflow can be considered as a cooperation of several pre-established workflows of several organizations. Chebbi et al. [Chebbi 06] presents an approach to interorganizational workflow cooperation to provide support for organizations which are involved in a shared but not pre-modeled cooperative workflow across organizational boundaries. They introduce cooperative activities, which can be partially visible for different partners. Their approach provides flexibility to workflow configuration. Such kind of approaches usually consists of following steps: (1) workflow advertisement, (2) workflow interconnections, and (3) workflow cooperation.

Workflow inheritance approach is mainly based on workflow interoperability. Van der Aalst [Aalst 02a] proposes a skeletal design of the overall business process as the first step, identifying all key tasks and the control data dependencies between them. In subsequent steps, different partners are assigned responsibility for completing certain parts of the workflow process and a workflow definition is created for each of the partners that include only those parts of original workflow definition that they are responsible for. Partners may then locally make modifications to the created workflow definition to accurately reflect their own business process and capabilities. The modifications made locally by individual business partners are managed using the concept of workflow inheritance. To summarize, the steps in workflow inheritance are as follows: (1) public workflow process design, (2) public workflow process assignment among organizations, (3) distributive private workflow creation in each organization, and (4) private workflow modification using inheritance-preserving transformation.

Contracts among organizations have been reported as a solution to interorganizational workflow in previous research. A typical example is the CrossFlow [Grefen 01], which address issues concerned with business process crossing organizational boundaries. A contract-based approach is used to define the business relationships between the organizations. Within this
contractual basis, interorganizational processes can be defined and performed. Cooperation between organizations is based on dynamic service outsourcing specified in electronic contracts. In their work, service enactment is performed by dynamically linking the workflow management infrastructures of the involved organizations. The lifecycle of a service outsourcing by contracts consists of four phases: (1) contract establishment, (2) dynamic infrastructure configuration, (3) contract enactment, and (4) dynamic infrastructure disposal.

During the past few years, great efforts have been made to specify interorganizational workflow processes for service-oriented architecture. Many workflow specification languages have been proposed, such as WSDL (Web Services Description Language)*, WSFL (Web Services Flow Language)†, XLANG‡, BPML (Business Process Modeling Language)§, ebXML (Electronic Business using eXtensible Markup Language)¶, WSCL (Web Services Conversation Language)‖, WPDL (Workflow Process Definition Language)*, BPEL (Business Process Execution Language)†† and so on. Besides of above languages, researches using specifications have been seen in previous work. Chiu et al. [Chiu 04] use workflow views as a fundamental mechanism for the interoperability of multiple workflows across business organizations. They describe a meta-model of workflow views and their semantics using a cross-organization workflow example. In their work, interoperation model of workflow views and its consistency criteria are provided. The approach provides interorganizational workflow in some formal specification languages.

In the context of workflow execution, dynamics, adaptation, flexibility and distribution have been main requirements [Ellis 95,"http://www.w3.org/TR/wsdl
*http://www.ebpml.org/wsfl.htm
†http://www.ebpml.org/xlang.htm
§http://www.ebpml.org/bpml.htm
¶ http://www.ebxml.org/
‖http://www.w3.org/TR/wsc110/
*http://www.wfmc.org
††http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf
15
Because of some commonly accepted characteristics of an agent (reactivity, autonomy, collaborative behavior, adaptivity, and mobility) [Bradshaw 97, Sycara 98, Wooldridge 02, Viroli 07], agent technology has been regarded as a promising solution to improve current workflow management systems addressing above issues [Chang 96, Judge 98, Singh 99, Huang 00, Jennings 00, O’Brien 01, Ricci 02, Buhler 03, Müller 04, Buhler 05, Ehrler 06].

To solve coordination issues in interorganizational workflow, Andonoff et al. [Andonoff 05a] deal with negotiation between organizations. Agent technology is used in this work for the following reasons: (1) it provides natural abstractions to deal with distribution, heterogeneity and autonomy which are inherent to loose interorganizational workflow (IOW), (2) it introduces powerful concepts such as groups and roles to describe in details the coordination of the different partners involved in the negotiation, and (3) it has investigated the problem of negotiation between agents. The authors specify what kind of negotiation is suitable in loose IOW, and identify what kind of agent behavior (competitive or cooperative) and what kind of negotiation protocol (auction, heuristic, argumentation or Contract-net) better suit loose IOW. They also propose an agent-based architecture, which is compliant with the Workflow Management Coalition reference architecture [Hollingsworth 95], to support negotiation between partners, and an organizational model based on the Agent-Group-Role meta model, which structures the negotiation in terms of Agents, Groups and Roles.

Mobile agents have been reported to be used to support interorganizational workflow management in [Merz 97]. This work argues that the mobile agent approach is well suited for sporadic communication in open distributed systems, especially for rather loose cooperation across local and organizational borders: In an increasing number of cases, management of distributed business procedures reaches beyond such borders. This means that for most existing workflow management systems, cooperating partners are required to give up their local autonomy. However, for cases in which business partners intend to cooperate but still need to preserve their local autonomy, process participation on the basis of mobile agents represents an
attractive and appropriate mechanism. The work shows how such kind of process integration can be achieved. It further demonstrates how the COSM (Common Open Service Market) system software can be extended in order to use petri net based process definitions which realize mobile agents in an integrated distributed system platform.

With recent development of services computing, workflow management has been extended into the Web services area [Greenwood 04, Vidal 04]. Service-oriented computing (SOC) is becoming the prominent paradigm for distributed computing and e-commerce, which makes developers create applications by using services as fundamental elements. Web services, an important SOC example, are defined as self-contained, self-described, active, modular software applications that can be advertised, discovered, and invoked over the Web, e.g., online travel reservation service, online electronic products purchasing service and so on. Services typically require collaborative enactment of workflows across multiple organizations.

Web service composition has been one of the most important research issues recently. The requirements of service composition differ from those of mainstream component-based software development [Milanovic 04]. For example, in place of access to documentation or code, SOC application developers and users only have access to WSDL’s rudimentary functional descriptions. Services are executed in different containers, separated by firewalls and other trustable barriers. In [Milanovic 04], the authors argue that a composition mechanism must therefore satisfy several requirements: connectivity, nonfunctional quality-of-service properties, correctness, and scalability. To meet those requirements, many approaches to service composition has been proposed, including BPEL, semantic Web (OWL-S), Web components, algebraic process composition, Petri nets, model checking and finite-state machines and so on [Korhonen 03, Singh 03, Blake 05, Negri 06].
2.3 Motivation Example

With the development of globalization, the needs of offshore software development have been increasing year by year. Offshore software development is provision of software development services by an external supplier positioned in a country that is geographically remote from the client enterprise, which is a type of offshore outsourcing. The main reasons behind the companies to use offshore software development services include reduced cost, improved performance, and access to a wider labor markets [DiRomualdo 98, Barthelemy 01].

However, cross-cultural issues are becoming an important factor in offshore software development. For example, Indian software companies have found that they need to communicate with U.S. and Japanese clients in very different ways. U.S. client companies normally work with extensive written agreements and explicit documentation, reinforced by frequent and informal telephone and email contact. In contrast, Japanese clients tend to prefer verbal communication, more tacit and continuously negotiated agreements, and less frequent but more formal use of electronic media [Krishna 04]. Such conflicts always cause the failure of the whole work.

Previous researches [Prikladnicki 03] have investigated difficulties in global software development, including software development process, training, planning and engagement, infrastructure, team integration, communication and feedback. Here we take the offshore software development between Chinese companies and Japanese companies for example, and show some typical reasons of failure that are summarized by some related companies.

In offshore software development between Japanese and Chinese companies, there are always conflicts caused by different customs and understandings over software development, which leads to the different modeling of the whole software development process. The following conflicts are always reported in the software development between companies in the two countries. In Japanese companies, software specifications are modified in parallel with development. However, the understanding of Chinese compa-
nies is that specifications are fully designed before development. Therefore, Chinese companies often get flustered by Japanese style of development. Next, Japanese companies always have strict software quality management and therefore they are often dissatisfied with the test process conducted by Chinese companies. Moreover, there always exist communication barriers between Japanese companies and Chinese companies, which lead to misunderstandings of parts of the whole collaboration work. Therefore, it is necessary to establish formal software development process for the collaboration, which can be modeled as an interorganizational workflow. The model should be designed from the perspective of both workflow requirements and organization consideration. We will use an example abstracted from above case throughout this thesis.
Chapter 3

Interorganizational Workflow Based on Local Process Views

In this chapter, we will propose an interorganizational workflow model, considering the issues of how to satisfy the requirements of organization and workflow design. The proposed interorganizational workflow model is illustrated by an example of collaborative software development. Further, we compare our model with previous approaches to show the effectiveness [Lin 07a].

3.1 Introduction

Workflow and business process management have been a research focus for years. Previous workflow process modeling methods include the purely graphical models like flow charts, mathematical models like Petri nets, object oriented models like UML, independent process modeling models like the ARIS methodology and so on. Most of the approaches have also been applied in interorganizational workflow which is modeled for collaboration work across organizational boundaries [Aalst 00, Grefen 01, Chebbi 06, Atluri 07]. With the emergence of service-oriented architecture (SOA), interorganizational workflow has also been much related to research issues like Web service composition [Staab 03, Milanovic 04, Martens 05]. Most
of the existing specifications and approaches in interorganizational workflow concentrate on workflow representation, verification and interaction protocols, with the assumption that organizations accept a common pre-defined model. However, there are some typical challenges with interorganizational workflows for collaboration in the real world.

As has been pointed out in Chapter 1, one challenge is that participants from different organizations have different cultural and knowledge backgrounds, which may produce a great impact on the way that the cooperative work is conducted [Hofstede 01]. Another important aspect is privacy preservation of individual organizations during collaboration [Aalst 02a, Chebbi 06]. Though cooperation of organizations needs some visible interactions and data sharing, it is also important for organizations to preserve their internal experiences and knowledge as much as possible. Therefore, when modeling the interorganizational workflow, it is necessary to consider about how to preserve privacy and autonomy of local workflows of organizations. Moreover, flexibility support and workflow reuse have been attracting attention [Dogac 98, Aalst 00].

When modeling interorganizational workflow, it is important to take different process modeling views of different organizations into consideration. In the real world, organizations always do not share a pre-defined global interorganizational workflow process, but have their own understandings of how to model the whole workflow process with their unique backgrounds and knowledge, which we call local process views of organizations. Moreover, to provide privacy preservation of organizations, the local workflows are not expected to be shared among organizations. Therefore we propose a model, where each organization has a local process view based on its own understanding of the whole collaboration.

The rest of the chapter is organized as follows: Section 3.2 provides an overview of our proposed interorganizational workflow model based on local process views with definitions in details. In Sect.3.3, an example of collaborative software development is introduced. Section 3.4 discusses the comparison of our approach and related work, followed by the summary in Sect.3.5.
3.2 Modeling Interorganizational Workflow

In Chapter 2, we have introduced the background of workflow interoperability in building interorganizational workflow model. Types of workflow interoperability in the interorganizational workflow include capacity sharing, chained execution, subcontracting, case transfer, and loosely coupled workflows [Aalst 99b]. The interorganizational workflow model in this research is mainly based on the loosely coupled interoperability, where the whole workflow is made up of several pieces, which may be active in parallel over different organizations [Aalst 00]. In a loosely coupled interorganizational workflow, only asynchronous communication is considered.

3.2.1 Overview of the Model

In the traditional loosely coupled interorganizational workflow, although each organization has its own local workflow that can be executed independently, there is a common global view of interorganizational workflow. Without considering the potential conflicts caused by different cultural backgrounds and understandings about the collaboration of different organizations, there is an assumption that all organizations accept pre-defined global workflow. However, this assumption is not reasonable because it is almost impossible for all organizations to have the same opinions on creating a global workflow view at the beginning stage, which is more obvious in the service-oriented environment.

In this research, we do not create a single global view of interorganizational workflow. Instead, each organization has its own local process view based on its own consideration of the whole collaboration work. The local process view of each organization represents its unique consideration of the whole workflow process, including the information of its local workflow, interaction part between organizations, and virtual workflow processes of its partners.

Figure 3.1 outlines the architecture for the interorganizational workflow model which supports multiple local process views. In this architecture,
each organization has a local process view. Since there is no global view, they coordinate with each other by compatibility analysis of different local process views of different organizations to detect conflicts and try to solve them. If the local process views of the organizations are mutually compatible, we can say the interorganizational workflow is globally compatible even without a global view.

3.2.2 Local Process Views

Several perspectives have been considered relevant for workflow modeling and workflow execution, including process perspective, organization perspective, data perspective, function perspective and operation perspective. In this research, we mainly focus on the process perspective, where workflow process definitions are defined to specify which tasks need to be executed and in what order [Aalst 00].
As described in Sect. 3.1, there have been various types of modeling methods of workflows in previous work, including the purely graphical models, workflow description languages, mathematical models, object oriented models, and so on. In some previous research, interorganizational workflows are modeled as Petri nets based workflow net (WF-net) [Aalst 02c]. There are several reasons to use Petri nets to model workflow: their formal semantics, graphical nature, expressiveness, analysis techniques, and tools that provide a framework for modeling and analyzing workflow processes [Aalst 02a]. In this research, the local process views might frequently change to eliminate detected incompatibilities. Therefore, flexibility is the most important factor when we choose the workflow model. Workflow graphs are developed with flexibility in mind. Moreover, workflow graphs can be easily transformed into WF-net [Aalst 02b]. For this purpose we use workflow graph to define the local process views.

**Definition 3.1 (Local Process View)** If an organization has $n$ partners in the collaboration environment, then its local process view can be expressed as a tuple $LPV = (I, WF_0, VWF_1, VWF_2, \ldots, VWF_n, ESC)$, where

1. $I$ is a finite set of organizations, including the local organization and its $n$ partners;

2. $WF_0$ is the workflow process of the local organization, and for each $k \in \{1, 2, \ldots, n\}: VWF_k$ is the virtual workflow process of its partner $k$, which is modeled with its own consideration (a workflow process is defined by Definition 3.2); and

3. $ESC$ is the interaction sequence chart which specifies the expected interaction between the local organization and its partners (in Definition 3.3).

Figure 3.2 shows a simple example of local process view of Partner A in an interorganizational workflow which involves two organizations. According to Definition 3.1, we can give the formalism of this local process view of Partner A in the example as $LPV = (I, WF_0, VWF_B)$. In the definition, $I = \{Partner A, Partner B\}$, $WF_0$ is the local workflow process of Partner
A, and $VWF_B$ is the virtual workflow process of Partner B which is designed by the consideration of Partner A.

In this research, workflow process is graph-oriented and defined as a structured process graph [Mendling 05].

**Definition 3.2 (Workflow Process and Virtual Workflow Process)** A workflow process is a tuple $WF = (i, T, F)$ where

1. $i$ is the information about the organization;
2. $T$ is the finite set of all the task nodes, $T = \{t_s, t_e\} \cup T^C$ such that:
   - $t_s$ denotes the start task of the workflow process
   - $t_e$ denotes the end task of the workflow process
   - $T^C$ denotes the finite set of tasks except $t_s$ and $t_e$

For each task $t \in T$, we define $t = (des, prec, eff, c, r)$ such that:

- $des$ is the description of the task
- $prec$ and $eff$ represent the precondition and the effect of the task respectively. Precondition is the condition that is required to be satisfied before the execution of a task, and effect means results produced (data, documents and
so on) by the execution of a task

- $c$ is the intra-connection type of the task such that $c \in C \rightarrow \{\text{direct, AND-split, AND-join, OR-split, OR-join, XOR-split, XOR-join}\}^*$, where split and join should appear in pairs of (AND-split, AND-join), (OR-split, OR-join) and (XOR-split, XOR-join) to ensure that the workflow process is sound [Vanhatalo 07]. Using this element of each task, we can represent the ordering structure between tasks within a local workflow process.

- $r$ is the inter-connection type of the task and $r \in R \rightarrow \{\text{null, in, out}\}^\dagger$.

Inter-connection identifies whether a task is an internal task or an interaction task which has links with other organizations:

1. $F$ is the finite set of flow relations, $F = F^N \cup F^I$ such that:
   - $F^N \subseteq T \times T$ is the finite set of internal flows. Each binary relation in $F^N$ represents an arc between two tasks.
   - $F^I \subseteq (T \times E) \cup (E \times T)$ is the finite set of interaction flows where $E$ is a finite set of events that present interactions between organizations. Each binary relation in $F^I$ represents an arc between a task and an interaction event.

A virtual workflow $VWF = (i, VT, VF)$ has the similar form of a common local workflow process, however, the tasks and flow relations are virtual. Tasks in a virtual workflow always have the abstraction form of tasks in a common local workflow, and the flow relations describe the flow of vir-

---

* According to the Workflow Reference Model of WfMC [Hollingsworth 95], there are following types of ordering structures for tasks. (1) Sequence: a task has a single subsequent task; (2) AND-split: a task splits into multiple parallel tasks that should be all executed; (3) OR-split: a task splits into multiple mutually exclusive alternative tasks, only one of which is followed; (4) AND-join: multiple parallel executing tasks join into a single task; (5) OR-join: multiple mutually exclusive alternative tasks join into a single task; (6) Loop: One or more tasks are repeatedly executed until the exit condition is satisfied. In our definition, we use direct type to represent both Sequence and Loop. For example, if $t_1$ is direct and we have an arc from $t_1$ to $t_2$, then the ordering type is Sequence; if $t_1$ and $t_2$ are both "direct" and we have both arcs from $t_1$ to $t_2$ and $t_2$ to $t_1$, then the ordering type is Loop.

† Here null means that the task is an internal task; in means that the task has interaction with other organizations by a receiving event; out means that the task has interaction with other organizations by a sending event.
Base on Definition 3.2, we can further describe the workflow processes in the example shown in Fig.3.2 formally. Here there are two workflow processes: $WF_0$ (the local workflow process of Partner A), and $VWF_B$ (the virtual workflow process of Partner B).

The local workflow process of Partner A $WF_0$ can be described as $WF_0 = (i, T, F)$ where

1. $i = Partner A$
2. $T = \{A_1, A_2, A_3, A_4\}$, $A_1$ is the start task and $A_5$ is the end task

\[ A_1 = (des, prec, eff, c, r) \]
- Description: $des = development analysis$
- Precondition: $prec = \{business analysis report\}$
- Effect: $eff = \{specification document\}$
- Intra-connection type: $c = direct$
- Inter-connection type: $r = out$

\[ A_2 = (des, prec, eff, c, r) \]
- Description: $des = prototype analysis$
- Precondition: $prec = \{business analysis report, specification document\}$
- Effect: $eff = \{prototype analysis document\}$
- Intra-connection type: $c = direct$
- Inter-connection type: $r = null$

\[ A_3 = (des, prec, eff, c, r) \]
- Description: $des = initial implementation analysis$
- Precondition: $prec = \{initial report document, initial prototype\}$
- Effect: $eff = \{specification analysis result\}$
- Intra-connection type: $c = direct$
- Inter-connection type: $r = in$

\[ A_4 = (des, prec, eff, c, r) \]
- Description: $des = specification modification$
- Precondition: $prec = \{specification analysis result\}$
- Effect: $eff = \{modified specification document\}$
- Intra-connection type: $c = direct$
- Inter-connection type: \( r = \text{out} \)

\[(3) \quad F = F^N \cup F^I\]

\[F^N = \{(A_1, A_2), (A_2, A_3), (A_3, A_4)\}\]

\[F^I = \{(A_1, e_1), (e_2, A_3), (A_4, e_3)\}\]

Similarly, the virtual workflow process of Partner A \( VWF_B \) can be described as \( VWF_B = (i, VT, VF) \) where

1. \( i = \text{Partner B} \)
2. \( VT = \{B_1, B_2, B_3\} \), \( B_1 \) is the start task and \( B_3 \) is the end task

\[B_1 = (\text{des}, \text{prec}, \text{eff}, c, r)\]
- Description: \( \text{des} = \text{initial implementation} \)
- Precondition: \( \text{prec} = \{ \text{specification document} \} \)
- Effect: \( \text{eff} = \{ \text{initial prototype} \} \)
- Intra-connection type: \( c = \text{direct} \)
- Inter-connection type: \( r = \text{in} \)

\[B_2 = (\text{des}, \text{prec}, \text{eff}, c, r)\]
- Description: \( \text{des} = \text{initial report} \)
- Precondition: \( \text{prec} = \{ \text{initial prototype} \} \)
- Effect: \( \text{eff} = \{ \text{initial report document} \} \)
- Intra-connection type: \( c = \text{direct} \)
- Inter-connection type: \( r = \text{out} \)

\[B_3 = (\text{des}, \text{prec}, \text{eff}, c, r)\]
- Description: \( \text{des} = \text{main implementation} \)
- Precondition: \( \text{prec} = \{ \text{modified specification document} \} \)
- Effect: \( \text{eff} = \{ \text{system} \} \)
- Intra-connection type: \( c = \text{direct} \)
- Inter-connection type: \( r = \text{in} \)

\[(3) \quad VF = VF^N \cup VF^I\]

\[VF^N = \{(B_1, B_2), (B_2, B_3)\}\]

\[VF^I = \{(e_1, B_1), (B_2, e_2), (e_3, B_3)\}\]

In previous research, [Aalst 99b] specifies interaction protocols in a loosely coupled interorganizational workflow by using Message Sequence
Charts (MSC). Message Sequence Charts are a widespread graphical language to visualize communications between systems/processes [Mauw 94]. The representation of message sequences charts is intuitive and focuses on messages between communication entities. In this research, we focus on the modeling phase of interorganizational workflow. Therefore, we define the interaction events among organizations by using a similar approach of message sequence charts, i.e., an Interaction Sequence Chart.

**Definition 3.3 (Interaction Sequence Chart)** An interaction sequence chart for representing interactions among organizations is defined as a tuple \( ESC = (I, E, \text{from}, \text{to}, \{ \preceq_i \}_{i \in I}) \) where

1. \( I \) is a finite set of organizations;
2. \( E \) is a finite set of interaction points (events);
3. \( \text{from} \) and \( \text{to} \) are functions from \( E \) to \( I \); and
4. For each \( i \in I : \preceq_i \) is a partial order on \( \{ ?e \mid e \in E \text{ and } \text{to}(e) = i \} \cup \{ !e \mid e \in E \text{ and } \text{from}(e) = i \} \) where \( ?e \) represents a receiving event and \( !e \) represents a sending event.

Figure 3.3 shows an example of interaction sequence chart which is based on the local process view in Fig.3.2. Based on Definition 3.3, we can describe it formally.

1. \( I = \{ \text{Partner A, Partner B} \} \)
2. In the local process view of Partner A, there are following four interaction points (events) between the organizations, \( E = \{ e_1, e_2, e_3, e_4 \} \).
   - \( e_1 : \text{send specification} \)
   - \( e_2 : \text{send initial report} \)
   - \( e_3 : \text{send modified specification} \)
   - \( e_4 : \text{send report review} \)
3. Each event has a unique direction from one organization to another. For example, \( \text{from}(e_1) = \text{Partner A} \) and \( \text{to}(e_1) = \text{Partner B} \).
4. The sequence order of events could be a partial order on an organization. For example, on Partner A, \( e_1 \) is always earlier than \( e_2 \), however \( e_3 \) and \( e_4 \) can be any order.
To provide flexibility of local workflow processes of organizations, we adopt top-down modeling methodology by using abstractions of tasks and workflow processes. We therefore induce virtual tasks and subprocesses into the local workflow process for organizations, which also appear in some previous research [Liu 03].

**Definition 3.4 (Subprocess)** A subprocess in a workflow process is defined as a tuple $sp = (des, TS, prec, eff)$ where

1. $des$ denotes the description of the abstract task;
2. $TS$ denotes a finite set of concrete tasks that composes the subprocess;
3. $prec$ and $eff$ denote the precondition and the effect of the subprocess as a whole.

A subprocess has the structure of a workflow process with strongly-
A workflow process is still a complete workflow process after replacing a set of tasks with a subprocess. In this research, we use subprocess to represent abstract-level information of a workflow process. Figure 3.4 is an example of a workflow process with many subprocesses. In the example, $T_1 - T_{29}$ are atomic tasks, and $A - K$ are subprocesses that consist of several atomic tasks. Therefore, each subprocess can be regarded as a virtual task.

3.3 Collaborative Software Development Scenario

To explain the proposed workflow model based on local process views, we demonstrate a case of intercultural software development collaboration between groups from different organizations: Partner A (outsourcer) and Partner B (supplier). This example is abstracted from some problems in real cases of offshore software development between Japanese companies and Chinese companies [Kunihiro 07]. In developing software, they have some rough agreement on the development assignment, e.g. Partner A designs the software specifications and Partner B develops the software based on the
designed specifications. Considering the fact that two groups come from different organizations, it is quite possible that there are conflicts caused by different customs and some misunderstandings over software development, which leads to differences in local process views of the whole workflow process. Such problems have been reported to be major causes for failures in offshore software development across nations [Krishna 04]. When creating the whole workflow process for the collaboration development, each partner wants to keep the detailed local workflow process secret. Therefore, they model the workflow process by their own customs. Figure 3.5 shows an example of their respective local process views based on the definitions.
we give in this research.

In software development collaboration, the following conflicts might occur between the two organizations: (1) In Partner A, software specifications are always modified in parallel with development. However, in Partner B, specifications are almost totally designed before development. Partner B might not be able to understand why the subprocess specification analysis is required. Therefore, when they model their local process view, specification analysis is not considered; (2) In Partner A, the software quality management is more strict. However, in Partner B, software tests are conducted less frequently. In Fig.3.5, Partner B develops the software, but they think that system test should be done by Partner A, which is totally different from the consideration of Partner A.

Due to above mutual misunderstandings, local process views of the two groups differs a lot in the interaction protocols and consideration of workflow processes. If neither organization considers different views before starting the whole software development process, conflicts must occur during the collaboration of the software development. Sometimes the whole collaboration might fail. Therefore, it is necessary to conduct compatibility analysis between different local process views before starting the whole workflow.

3.4 Discussion

As we summarize in Chapter 2, several types of interorganizational workflow approaches have been proposed in previous researches, including workflow cooperation and composition, workflow inheritance, agreement and contracts between organizations, workflow specifications and so on. Here we discuss the interorganizational workflow model by comparing our approach and related work. For this purpose, we choose some typical research for the above types: [Chebbi 06] for workflow cooperation and composition, [Aalst 00] for workflow inheritance, [Grefen 01] for agreement and contracts between organizations, [Chiu 04] for workflow specifications.
Our approach of modeling interorganizational workflow covers both the perspective of workflow and organization requirements, while almost all of previous approaches neglect the latter perspective. Here we give a comparative summary of typical previous efforts in interorganizational workflow in Table 3.1 and 3.2 from both perspectives.

Table 3.1: Comparison of models with the perspective of workflow

<table>
<thead>
<tr>
<th>Approach</th>
<th>Comparison Items</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>privacy</td>
</tr>
<tr>
<td>Chiu et al. [Chiu 04]</td>
<td>✓</td>
</tr>
<tr>
<td>Grefen et al. [Grefen 01]</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Chebbi et al. [Chebbi 06]</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>van der Aalst [Aalst 02a]</td>
<td>✓ ✓</td>
</tr>
<tr>
<td>Our approach</td>
<td>✓ ✓</td>
</tr>
</tbody>
</table>

In the perspective of workflow requirements, the comparison items include privacy preservation, flexibility support, workflow reuse and verification analysis.

(1) Privacy preservation: in modeling interorganizational workflow, most of approaches consider the aspect of privacy preservation of organizations, where local workflow processes of organizations are not entirely open to public. In some approaches, organizations provide part of its workflow as the public part by using workflow abstraction [Grefen 01, Chebbi 06]. In other approaches, organizations design local workflows privately based on a common accepted abstract interorganizational workflow [Aalst 02a]. In our approach, organizations do not open its local workflows, instead they design virtual workflows for other organizations based on their own understandings. The virtual workflows are open to other partners for detecting conflicts.

(2) Flexibility support: flexibility in interorganizational workflow managements covers many aspects, e.g., flexibility of local workflow design and workflow change, flexibility of exception handling, flexibility of workflow
system usage and so on. In our approach, we mainly concentrate on the flexibility of local workflow design of organizations, which is similar to some previous approaches [Aalst 02a, Chebbi 06]. Moreover, since each organization has its own local process view in our approach, it is flexible for organizations to design the whole interorganizational workflow view based on their understanding.

(3) Workflow reuse: the issue of workflow reuse cannot be well addressed in some previous researches that are not service-oriented. For example, in [Aalst 02a], organizations cannot reuse its local workflow for different interorganizational workflow because the local workflows are always designed based on some common interorganizational workflow views. Some approaches enable organizations to preserve their local workflows as services [Chiu 04, Chebbi 06]. However, local workflows need cooperation and composition mechanisms to establish an interorganizational workflow. In our approach, each organization has a local process view including not only local workflows but also interaction information that can be preserved as a workflow service.

(4) Verification analysis: some previous researches use specification languages or mathematical models to represent workflow processes and therefore provide verification functions [Aalst 02a, Chiu 04]. Currently, the issue of verification analysis is not considered in our approach, but it is important to address this issue by extending our current interorganizational workflow model.

In the perspective of organization consideration, the comparison items include coordination efforts and multiple view support.

(1) Coordination efforts: some previous approaches discuss about coordination among organizations. Chebbi et al. [Chebbi 06] propose the idea that organizations coordinate their private workflow through third party. However, it is not clear what is required for organizations to conduct for coordination. In the approach of van der Aalst [Aalst 02a], organizations try to coordinate to get a common part for interorganizational workflow, and design their own private local workflow processes based on the common part. However, organizations might have conflict with each other if there is no
Table 3.2: Comparison of models with the perspective of organization

<table>
<thead>
<tr>
<th>Approach</th>
<th>Comparison Items</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>coordination efforts</td>
</tr>
<tr>
<td>Chiu et al. [Chiu 04]</td>
<td></td>
</tr>
<tr>
<td>Grefen et al. [Grefen 01]</td>
<td></td>
</tr>
<tr>
<td>Chebbi et al. [Chebbi 06]</td>
<td>✓</td>
</tr>
<tr>
<td>van der Aalst [Aalst 02a]</td>
<td>✓</td>
</tr>
<tr>
<td>Our approach</td>
<td>✓</td>
</tr>
</tbody>
</table>

coordination between local workflows. In our approach, the coordination efforts involve both the interaction part and the local workflows.

(2) Multiple view support: almost none of previous approaches of modeling interorganizational workflow support multiple views for different organizations.

3.5 Summary

Modern interorganizational workflow has several main challenges. First, organizations have different views of how to model the workflow process due to their unique cultures and knowledge of background. Second, each organization wants to maintain the privacy of its local workflow from other organizations. Moreover, flexibility support and workflow reuse are also important aspects that should be considered. In this research, we provide an interorganizational workflow approach for organizations to model collaborative business based on their own consideration of the whole process, and coordinate their unique views with partners to reach mutual understanding and compatibility.

We propose an interorganizational workflow model, taking different process modeling views of different organizations into consideration. In this model based on local process views, organizations do not share a pre-
defined global interorganizational workflow process, but have their own understandings of how to model the whole workflow process with their unique backgrounds and knowledge. The local workflows are not designed to be shared among organizations. Organizations can manage their own local workflow processes autonomously. The proposed interorganizational workflow model for collaboration has the following features: consideration of different workflow modeling views of different organizations, privacy preservation of organizations and flexibility of local workflows.
Chapter 4

Coordination of Local Process Views

Based on the model we proposed in Chapter 3, we will further propose a coordination approach by conducting compatibility analysis of different local process views from different organizations to detect conflicts in this chapter. We implement a tool for compatibility analysis and use a case study of collaborative software development to validate our approach [Lin 07a].

4.1 Introduction

Each organization in the interorganizational workflow has its own local process view, including elements of local workflow process, interaction protocols, and virtual workflow processes of other organizations that interact with it. Detailed information of local workflow process is included in the local process view. However, the virtual workflow processes of other organizations are defined based on the requirement and consideration of the local organization. For example, in Fig.3.5, Partner A has its own detailed local workflow process. Partner B also has its own understanding of Partner A’s workflow process, which is modeled as a virtual workflow process of Partner A in the local process view of Partner B. Therefore, conflicts may arise between the virtual workflow process designed by the local organization
and the related detailed workflow processes of other organizations. Furthermore, when defining interaction protocols, there may also be conflicts between different local process views of different organizations. Therefore, coordination is necessary for organizations before executing interorganizational workflow. We consider the following research issues in this chapter.

**Coordinating different local process views** The different local process views should be compatible with each other for execution. By conducting coordination with compatibility analysis of local process views of different organizations, incompatibilities are expected to be detected and eliminated before workflow execution, and therefore a compatible interorganizational workflow that supports multiple local process views can be established. Coordination is a process for organizations to get mutual understandings of interaction protocols and local workflows, which is an extremely important step in interorganizational workflow management while neglected in most of the previous researches. In this issue, we propose an approach of coordinating different local process views, which is divided into two main phases: coordination of interaction protocols, and coordination of local workflows.

**Validating the approach by case and implementation** To validate the proposed interorganizational workflow model and the coordination approach of local process views, it is necessary to apply in real-world case with implementation. Therefore, in this research issue, we take collaborative software development workflow for instance. Further, extension application in service-oriented environment is discussed.

The rest of the chapter is organized as follows: Section 4.2 introduces the coordination issues and approaches. Section 4.3 proposes the compatibility analysis mechanisms for detecting conflicts of different local process views. A tool implementation for compatibility analysis is provided in Sect.4.4. Section 4.5 is a case study of collaborative software development for the compatibility analysis mechanisms. The application of workflow
services based on compatibility analysis is introduced in Sect. 4.6. Section 4.7 is the summary of this chapter.

4.2 Coordination Issues and Approaches

In the process of coordination of different local process views of different organizations, to make the local process views compatible with each other, it is necessary for each organization to interact and coordinate with other organizations by conducting compatibility analysis to eliminate potential conflicts about local process views. During the compatibility analysis between the local process views, the interaction protocols designed by different organizations should first be compared and validated. Only when the two organizations reach agreement on the interaction protocols can they continue the following steps of compatibility analysis.

We describe the coordination in details by focusing on the compatibility analysis of local process views, which is divided into two phases, namely the interaction protocol coordination and local workflow process coordination. This research mainly deals with how to detect conflicts among organizations, while the negotiation process to eliminate conflicts is not the focus of this research. In the following sections, we introduce the two phases of compatibility analysis. This research deals with **compatibility analysis between two partners**. Figure 4.1 shows the whole compatibility mechanisms for different local process views of organizations.

4.3 Compatibility Analysis of Local Process Views

4.3.1 Phase 1: Interaction Protocol Coordination

In the interorganizational collaboration environment, the interaction protocol is the most important issue for all partners. In this research, we use interaction sequence chart (ESC) to define the interaction protocol between
partners. However, since different partners have different considerations for the interaction, incompatibility of interaction protocols might easily cause incompatibility of the workflow processes. Therefore, when conducting compatibility analysis, interaction protocols should be first analyzed, which include two parts, i.e., the content and order of interaction events between two organizations.

**Definition 4.1 (Incompatibility of Interaction Protocols)** If we define the interaction sequence charts of two organizations A and B as
\[ ESC_A = (I, E_A, from_A, to_A, \{ \preceq_i \}_{i \in I}) \]

\[ ESC_B = (I, E_B, from_B, to_B, \{ \preceq_i \}_{i \in I}) \]

respectively, \( I = \{ \text{Org}_A, \text{Org}_B \} \), then the following cases of incompatibility might occur in the interaction protocols of two partners.

1. **Inexistent Event**: there exists an interaction event in one local process view but does not exist in the other, notationally,

\[ (\exists e)(\left( e \in E_A \right) \land \left( e \notin E_B \right)) \lor \left( \left( e \in E_B \right) \land \left( e \notin E_A \right) \right) \]

2. **Inconsistent Direction**: the sender of an interaction event in one local process view is the receiver of the other, notationally,

\[ (\exists e)(\left( from_A(e) = \text{Org}_A \right) \land \left( to_B(e) = \text{Org}_A \right)) \lor \left( \left( from_A(e) = \text{Org}_B \right) \land \left( to_B(e) = \text{Org}_B \right) \right) \]

3. **Inconsistent Condition**: the precondition or effect of a task that is connected with an event is different between the two local process views, notationally,

\[ (\exists e, t_A, t_B)(\left( \text{des}(t_A) = \text{des}(t_B) \right) \land \left( \left( t_A, e \right) \in F^I_A \right) \land \left( \left( t_B, e \right) \in F^I_B \right) \land \left( \text{eff}(t_A) \neq \text{eff}(t_B) \right)) \lor \left( \left( \left( e, t_A \right) \in F^I_A \right) \land \left( \left( e, t_B \right) \in F^I_B \right) \land \left( \text{prec}(t_A) \neq \text{prec}(t_B) \right) \right) \]

4. **Disordered Sequence**: there exists two events that have opposite orders in two local process views, notationally,

\[ (\exists e_i, e_j)(\left( < e_i, e_j > \in \{ \preceq_i \}_A \right) \land \left( < e_j, e_i > \in \{ \preceq_i \}_B \right) \).

The goal of coordinating interaction protocols of two local process views is to detect above incompatibilities and try to eliminate them. If two local process views do not contain any of above cases, then the interaction protocols of two organizations are compatible with each other. The algorithm of compatibility analysis of interaction protocols is shown in Algorithm 1, which can detect all incompatibility cases of interaction protocols of two partners.
procedure Compatibility-Analysis-1(LPV, LPV')
1: IncompatibleFlag ← false
2: for all e ∈ E do
3: if (from(e) = OrgA) ∧ (to(e) = OrgB) then
4: if (e ∉ E') then
5: IncompatibleFlag ← true
6: display Inexistent Event e
7: else if td'(e) = OrgA then
8: IncompatibleFlag ← true
9: display Inconsistent Direction e
10: else if ((des(t) = des(t')) ∧ ((t, e) ∈ F₀) ∧ ((t', e) ∈ Fₐ') ∧ 
       (eff(t) ≠ eff(t'))) then
11: IncompatibleFlag ← true
12: display Inconsistent Condition t
13: end if
14: end if
15: Repeat line 3-14 by reversing the direction of e
16: for all { eᵢ, eⱼ } ⊆ E ∪ E' do
17: if (< eᵢ, eⱼ > ∈ { ≤ i }ₐ) ∧ (< eⱼ, eᵢ > ∈ { ≤ i }ₐ) then
18: IncompatibleFlag ← true
19: display Disordered Sequence eᵢ, eⱼ
20: end if
21: end for
22: end for
23: if IncompatibleFlag = true then
24: return Incompatible
25: else
26: return Compatible
27: end if

Algorithm 1: Compatibility analysis phase 1

4.3.2 Phase 2: Local Workflow Process Coordination

In a local process view of each organization, its local workflow process is represented in details. Moreover, the workflow processes of its partners are
also considered with its own understandings. By this means, the potential conflicts and different views of the same interorganizational workflow are well embedded. In the first phase of compatibility analysis, organizations detect some conflicts about the interaction protocols among organizations. Such conflicts are expected to be eliminated through coordination and negotiation between involved organizations. After that, the compatibility analysis comes to the second phase, coordinating local workflow processes designed by different organizations. For each organization \(i\), it is necessary to compare its local workflow process \(WF_0\) with the virtual workflow processes of \(i\) that are designed in the local process views of its partners.

Here we first define the incompatibility types of workflow process, and then propose an approach of how to detect such incompatibilities.

**Definition 4.2 (Incompatibility of Workflow Processes)** If we define the local process views of two organizations A and B as

\[
LPV_A = (I, WF_0A, VWF_B, ESC_A),
\]

\[
LPV_B = (I, WF_0B, VWF_A, ESC_B)
\]

where \(ESC_A\) and \(ESC_B\) are compatible, then incompatibility of workflow processes include following cases.

1. **Inexistent Task**: there exists a task in the workflow process modeled by other partners but does not exist in the real concrete-level local workflow process, notationally,

\[
(\exists t)((t \in VT_A) \land (t \notin T_0A)) \lor ((t \in VT_B) \land (t \notin T_0B));
\]

2. **Inconsistent Condition**: the preconditions or effects of the same task (e.g., the situation where a task exists in both \(WF_0A\) of \(LPV_A\) and \(VWF_A\) of \(LPV_B\)) are different in two local process views, i.e.,

\[
(\exists t, tt)(((t \in VT_A) \land (tt \in T_0A)) \lor ((t \in VT_B) \land (tt \in T_0B))) \land (des(t) = des(tt)) \land ((prec(t) \neq prec(tt)) \lor (eff(t) \neq eff(tt)));
\]
(3) **Disordered Tasks:** there exist two tasks that have opposite sequential orders in two local process views, notationally,

\[
(\exists t_i, t_j) ((t_i, t_j) \in VT_A) \land (t_i, t_j) \in T_{0A}) \land ((t_i, t_j) \in VF^N_A) \land ((t_j, t_i) \in F^N_{0A}) \lor ((t_i, t_j) \in VT_B) \land (t_i, t_j) \in T_{0B}) \land ((t_i, t_j) \in VF^N_B) \land ((t_j, t_i) \in F^N_{0B})
\]

If two local process views could avoid all the above incompatibilities, we call they are *compatible*.

For each workflow process, the local organization models it as a relatively detailed one but other organizations model it in an abstract level based on their understandings about it. Therefore, there are following difficulties when analyzing the abstract-level virtual workflow process and the detailed-level workflow process of the same organization: (1) the structure of local workflow process modeled by the local organization is always far more complicated and (2) some parts in the abstract-level virtual workflow processes cannot be directly mapped into the detailed-level local workflow processes. For example, a task in the abstract-level virtual workflow process might be equivalent to a group of tasks in the related detailed-level local workflow process. To compare a virtual workflow process with a detailed local workflow process, the abstractions of tasks in the detailed local workflow process are necessary. Therefore virtual tasks and subprocesses are added into the detailed-level local workflow process for each organization when designing the local process view.

Take the workflow process in Fig.3.4 for instance; a local workflow can be abstracted into different levels of abstract workflows by using subprocess abstraction. Since workflow process is defined as a structured process graph in this research, it can be reduced to a single node [Mendling 05]. Here we can simplify the graph by blocks and subprocesses. A block is a unit of representation that minimally specifies the behavioral pattern of a process flow [Bae 04]. Such workflow structures have also been summarized as different types of workflow patterns [Aalst 03c]. We used the following types of blocks that are discussed in this research, iterative block,
serial block, branch block (including parallel block, conditional selective block and non-conditional selective block). Therefore, we can transform a complicated workflow process with its abstract-level subprocesses into a block-structured hierarchical workflow tree so that the tree contains not only the atomic-level tasks but also abstract-level subprocesses. Then we can compare the virtual workflow process and the related local workflow process. Similar approach of workflow transformation has also been used in previous research for different purposes [Sadiq 00, Bae 04]. Figure 4.2 shows structures of those blocks and the subprocess.

The algorithm of the transforming process is shown in Algorithm 2-7. Algorithm 2 is for iterative block detection. Algorithm 3 is used to compute weights of all the tasks. The start task node has weight 1, which means it is the outermost node in the workflow graph. If there is a split

\[ \text{split} \]

\[ \text{join} \]

Split and join are used in parallel block (\( \text{AND} \rightarrow \text{split, AND} \rightarrow \text{join} \)), conditional selective block (\( \text{OR} \rightarrow \text{split, OR} \rightarrow \text{join} \)) and non-conditional selective block (\( \text{XOR} \rightarrow \text{split, XOR} \rightarrow \text{join} \)). They can be categorized as branch block.
task node, then the weight of its successor node will be larger than it. If there is a join task node, then the weight of the successor node will be less than it. As a result, the most inner node has the largest weight. Algorithm 4-6 are responsible for automatically detecting serial blocks, branch blocks and subprocesses based on Algorithm 3. Algorithm 7 uses the task weight to decide what kind of blocks should be replaced in every step. Besides blocks, the subprocesses are also replaced during this process. The algorithm stops when the workflow process is reduced into one single node, and a workflow tree is finally created.

In Algorithm 7, we use several rules for workflow process transformation. For a workflow process $WF = (i, T, F)$, we have the following rules for the transformation of a workflow process graph structure into hierarchical structure, including iterative block transformation rule, serial block transformation rule, branch block transformation rule and subprocess transformation rule. Rule 1 is for handling the iterative block. In the process of transforming the workflow into hierarchical structure, the iterative arcs are removed for separate analysis. Rule 2 and Rule 3 are for serial block transformation and branch block transformation respectively. While these two types are detected in the workflow process, the involved tasks are replaced
procedure Iterative-Block-Detection(WF)
1:  \( Q \leftarrow \emptyset \)
2:  /*\( color[u] \) identifies whether \( u \) has been checked*/
3:  for all \( u \in T \) do
4:      \( color[u] \leftarrow 0 \)
5:  end for
6:  \( color[t_s] \leftarrow 1 \)
7:  ENQUEUE(\( Q, t_s \))
8:  while \( Q \neq \emptyset \) do
9:      \( u \leftarrow \text{DEQUEUE}(Q) \)
10:     if \( color[u] = 0 \) then
11:        \( color[u] \leftarrow 1 \)
12:     end if
13:     /*\( \text{succ}[u] \) is the successor of \( u \)*/
14:     for all \( v \in \text{succ}[u] \) do
15:        if \( color[v] = \text{GREY} \) then
16:           return \( (u, v) \)
17:        end if
18:     end for
19:     /*\( \text{pred}[v] \) is the predecessor of \( v \)*/
20:     for all \( (v' \in \text{pred}[v]) \land (color[v'] = 1) \) do
21:        ENQUEUE(\( Q, v \))
22:     end for
23:  end while

Algorithm 2: Iterative block detection

by the block, which is also regarded as a special task in the updated workflow process. In the workflow process tree, a new node for the detected block is created at the same time. All the tasks in the block then become a child node of the block in the tree. Rule4 is for transforming subprocesses. The operation is similar to Rule2 and Rule3. Notationally, these rules can be described as follows.

Rule 1: iterative block transformation
\[ \exists (t_1, t_2, \ldots, t_n) \in T \text{ where } \text{ib} = (t_1, t_2, \ldots, t_n) \]
procedure Task-Weight-Computation(WF)
1: split ← \{AND-split, OR-split, XOR-split\}
2: join ← \{AND-join, OR-join, XOR-join\}
3: for all \( t \in T \) do
4: \( \text{weight}(t) \leftarrow 0 \)
5: end for
6: \( \text{weight}[t_s] \leftarrow 1 \ u \leftarrow t_s \ \text{max}_{\text{w}} \leftarrow 0 \)
7: for all \( v \in \text{succ}[u] \) do
8: if \( \text{weight}[v] = 0 \) then
9: \( \text{if} \ c[u] = \text{null} \) then
10: \( \text{weight}(v) \leftarrow \text{weight}(u) \)
11: else if \( c[u] \in \text{split} \) then
12: \( \text{weight}(v) \leftarrow \text{weight}(u) + 1 \)
13: else if \( c[u] \in \text{join} \) then
14: \( \text{weight}(v) \leftarrow \text{weight}(u) - 1 \)
15: end if
16: if \( \text{weight}[v] > \text{max}_{\text{w}} \) then
17: \( \text{max}_{\text{w}} \leftarrow \text{weight}[v] \)
18: end if
19: end if
20: end for
21: return \( \text{max}_{\text{w}} \)

Algorithm 3: Task weight computation

Rule 1: \(((i, T, F), ib) \rightarrow (i, T', F')\)
\(T' = T\)
\(F' = F - \{(t_n, t_1)\}\)

Rule 2: serial block transformation
\(\exists (t_1, t_2, \ldots, t_n) \in T\) where \(sb = (t_1, t_2, \ldots, t_n)\)
Rule 2: \(((i, T, F), sb) \rightarrow (i, T', F')\)
\(T' = T \cup \{sb\} - \{t_1, t_2, \ldots, t_n\}\)
\(F' = F \cup \{(t_{in}, sb)/ (t_{in}, t_1) \in F\} \cup \{(sb, t_{out})/(t_n, t_{out}) \in F\} - (T \times \{t_1\}) \cup \{(t_1, t_2), \ldots (t_{n-1}, t_n)\} \cup \{t_n \times T\}\)

Rule 3: branch block transformation
procedure Serial-Block-Detection(WF, maxw)
1: SeqFlag ← 1
2: sb, Q ← ∅
3: ENQUEUE(Q, ts)
4: while SeqFlag = 1 do
5: if Q = ∅ then
6: return null
7: end if
8: u ← DEQUEUE(Q)
9: /* out[u] is the OutDegree and in[u] is the InDegree of u*/
10: /* succ[u] is the successor of u*/
11: if (weight[u] = maxw) ∧ (out[u] = 1) ∧ (in[succ[u]] = 1) then
12: SeqFlag ← 0
13: sb ← sb ∪ {u}
14: while (out[u] = 1) ∧ (in[succ[u]] = 1) do
15: sb ← sb ∪ {succ[u]}
16: u ← succ[u]
17: end while
18: update related tasks with serial block
19: weight[sb] ← weight[u]
20: else
21: ENQUEUE(Q, succ[u])
22: end if
23: end while

Algorithm 4: Serial block detection

∃(t1, t2, . . . , tn) ∈ T where bb = (t1, t2, . . . , tn)
Rule3 : ((i, T, F), bb) → (i, T', F')
T' = T ∪ {bb} − {t1, t2, . . . , tn}
F' = F ∪ {(t in, bb)/(t in, t1) ∈ F} ∪ {(bb, t out)/(t1, t out) ∈ F} − (T ×
{t1, t2, . . . , tn}) ∪ ({t1, t2, . . . , tn} × T)
Rule 4: subprocess transformation
∃(t1, t2, . . . , tn) ∈ T where sp = (t1, t2, . . . , tn)
Rule4 : ((i, T, F), sp) → (i, T', F')
procedure Branch-Block-Detection(WF, max_w)
1: BraFlag ← 1
2: bb, Q ← Ø
3: ENQUEUE(Q, t_i)
4: while BraFlag = 1 do
5: u ← DEQUEUE(Q)
6: if (weight[u] = max_w) then
7: BraFlag ← 0
8: /*pred[u] is the predecessor of u*/
9: for all v ∈ succ[pred[u]] do
10: bb ← bb ∪ {v}
11: end for
12: update related tasks with branch block
13: weight[bb] ← weight[u] − 1
14: update max_w
15: else
16: ENQUEUE(Q, succ[u])
17: end if
18: end while

Algorithm 5: Branch block detection

\[
T' = T \cup \{sp\} - \{t_1, t_2, \ldots, t_n\} \\
F' = F \cup \{(t_{in}, sp)/ (t_{in}, t_1) \in F\} \cup \{(sp, t_{out})/ (t_{n}, t_{out}) \in F\} - (T \times \{t_1, t_2, \ldots, t_n\}) \cup (\{t_1, t_2, \ldots, t_n\} \times T)
\]

All the rules described above are based on the workflow process in Definition 4.2. Based on workflow process transformation algorithm, the workflow process in Fig.3.4 can be transformed into a process tree as shown in Fig.4.3 (bb=branch block; sb=serial block). After the workflow process is transformed into a hierarchical workflow process, we can conduct the compatibility analysis of workflow processes. Therefore, here we have two steps: (1) Search and map the tasks of abstract-level virtual workflow process in the transformed workflow tree, i.e., for each task \( t \in VT \), check whether there exists an element in the workflow tree so that \( t \) is equivalent to
**procedure**  Subprocess-Detection(WF)

1: /*b is a detected block.*/
2: /*BID is the subprocess ID set of block b.*/
3: /*SPID is the ID set of detected subprocesses.*/
4: BID, SPID ← ∅
5: sp ← ∅
6: for all u ∈ b do
7:   /*ID[u] is the subprocess ID set of task node u.*/
8:   if ID[u] ∉ BID then
9:      BID ← BID ∪ ID[u]
10:   end if
11: end for
12: /*id is a subprocess ID.*/
13: for all id ∈ BID do
14:   Flag ← 0
15: for all u ∈ b do
16:   if id ∉ ID[u] then
17:      Flag ← 1
18:      sp ← b − {u}
19:   end if
20: end for
21: if Flag = 1 then
22:   SPID ← SPID ∪ {id}
23: end if
24: end for
25: BID ← BID − SPID

Algorithm 6: Subprocess detection

**tt** by \( prec(t) = prec(tt) \) and \( eff(t) = eff(tt) \); (2) Compare the order of tasks in the abstract-level virtual workflow process and the related task order in the workflow tree. Then, we can detect cases of incompatibility described in Definition 4.2. The compatibility analysis for workflow processes is implemented by **Algorithm 8**.
procedure Process-Transform($WF, max_w$)
1: detect iterative block
2: \textbf{if} IterativeDetected = true \textbf{then}
3: \hspace{1em} apply Rule1 : $((i, T, F), ib) \rightarrow (i, T', F')$
4: \textbf{end if}
5: \textbf{repeat}
6: detect serial block with $max_w$
7: \textbf{if} SerialDetected = true \textbf{then}
8: \hspace{1em} apply Rule2 : $((i, T, F), sb) \rightarrow (i, T', F')$
9: \hspace{1em} detect subprocess
10: \hspace{1em} \textbf{if} SubprocessDetected = true \textbf{then}
11: \hspace{2em} apply Rule4 : $((i, T, F), sp) \rightarrow (i, T', F')$
12: \hspace{2em} \textbf{end if}
13: update $WF$ and $max_w$
14: \textbf{else}
15: \hspace{1em} detect branch block with $max_w$
16: \hspace{1em} \textbf{if} BranchDetected = true \textbf{then}
17: \hspace{2em} apply Rule3 : $((i, T, F), bb) \rightarrow (i, T', F')$
18: \hspace{2em} detect subprocess
19: \hspace{2em} \textbf{if} SubprocessDetected = true \textbf{then}
20: \hspace{3em} apply Rule4 : $((i, T, F), sp) \rightarrow (i, T', F')$
21: \hspace{3em} \textbf{end if}
22: update $WF$ and $max_w$
23: \textbf{end if}
24: \textbf{end if}
25: \textbf{until} only one node is remained
26: \textbf{return} workflow process tree

Algorithm 7: Workflow process transformation

4.4 Tool Implementation for Compatibility Analysis

We have implemented a demo tool for compatibility analysis between local process views based on the compatibility analysis mechanism we proposed
procedure Compatibility-Analysis-2($LPV, LPV'$)
1: $IncompatibleFlag \leftarrow false$
2: $TWF_0 \leftarrow$ Process-Transform($WF_0, max_w$)
3: for all $t \in VT'$ do
4: $map[t] \leftarrow null$
5: for all $tt \in TT_0$ do
6: if $des(t) = des(tt)$ then
7: if $(prec(t)=prec(tt)) \land (eff(t)=eff(tt))$ then
8: $map[t] \leftarrow tt$
9: else
10: $IncompatibleFlag \leftarrow true$
11: display Inconsistent Condition $t$
12: end if
13: break
14: end if
15: end for
16: if $map[t] = null$ then
17: $IncompatibleFlag \leftarrow true$
18: display Inexistent Task $t$
19: end if
20: end for
21: for all $(t_i, t_j) \in VF'^N$ do
22: if $(map[t_j], map[t_i]) \in TF'_0$ then
23: $IncompatibleFlag \leftarrow true$
24: display Disordered Tasks $t_i, t_j$
25: end if
26: end for
27: if $IncompatibleFlag = true$ then
28: return Incompatible
29: else
30: return Compatible
31: end if

Algorithm 8: Compatibility analysis phase 2
in this research. In this tool, user can design the local process view, including interaction protocol, local workflow process and virtual workflow process of other organizations. The workflow process can be designed visually and saved as XML data. Compatibility analysis can be conducted between two organizations by exchanging public information including interaction protocols and virtual workflow process as the scenario we show in Fig.4.4. The implemented demo tool includes three modules: (1) workflow process design module; (2) workflow process transformation module; (3) compatibility analysis module.

Figure 7 shows the screenshot of second phase compatibility analysis by Partner B (a part of example in Fig.3.5). The first phase is to analyze different interaction protocols designed by two different organizations. When defining the local workflow process, Partner B adds some subprocesses so that both abstract-level and detailed-level process are involved in the second analysis phase. Further, in order to compare the abstract-level virtual workflow with the detailed-level local workflow, the local workflow with subprocesses is transformed into a workflow process tree. At last, incompatibilities are visualized and informed in details. By two phases’ compatibility analysis, incompatibilities are detected between local process views of two partners, which are expected to be eliminated before the start of collaboration. By this means, the potential conflicts can be reduced in the whole collaboration activity.

4.5 Case Study

In the section, we use a case of collaborative software development workflow (the example in Fig.3.5) to illustrate the approach of coordination we proposed in this research. After modeling the local process views of the two organizations, compatibility analysis should be conducted for coordination.

First, interaction protocols between two partners are compared. In the example shown in Fig.3.5, Partner A’s local process view includes following five interaction events: (1) \( send \ specification_{A \rightarrow B} \);
Figure 4.4: Screenshot of the compatibility analysis tool

(2) send initial report $B \rightarrow A$; (3) send modified specification $A \rightarrow B$; (4) send test report $B \rightarrow A$; and (5) send evaluation report $A \rightarrow B$. However, the interaction events included in Partner B’s local process view are a little different. By using the compatibility analysis phase 1 for interaction protocol coordination described in Algorithm 1 in Sect.4.3.1, the incompatibilities
among interaction protocols can be detected. Then, two partners try to negotiate with each other to reach an agreement on how to modify the interaction protocols for mutual compatibility until there is no incompatibility between the interaction parts of the two organizations.

Second, within each partner, the local workflow and its virtual workflow designed by the other partner are compared. To compare these workflows, we first transform the local detailed workflow process into a block-structured hierarchical workflow process using the approach described in Algorithm 3. The hierarchical workflow process are transformed from the local workflow and combined with subprocesses. For example, in Partner A’s local process view, the hierarchical workflow process combines three subprocesses: requirement design, test preparation and specification analysis. The parameters of precondition and effect for these atomic tasks and subprocesses are defined before the coordination process. Then, the virtual workflow in Partner B’s local process view and the block-structured hierarchical workflow process, which is transformed from the local workflow in Partner A’s local process view, could be compared. To compare these workflows, we search and map the virtual tasks of the virtual workflow in the block-structured hierarchical workflow. Precondition and effect of tasks are compared during this process. By this means, incompatibilities of work-
flow processes designed by two partners can be detected using Algorithm 4. For example, there is a conflict that two partners have different opinions about who should be responsible for the task system test in the two initial local process views. After such compatibility analysis, incompatibilities are detected before executing incompatible workflows. Such incompatibilities are expected to be eliminated by agent negotiation between the two organizations. Therefore, the organizations can execute the overall interorganizational workflow with fewer potential conflicts by the coordination of their local process views, which is a process of mutual understanding.

The compatibility analysis can be conducted using the tool we implemented. Figure 4.5 shows an example of the interaction protocols and local workflow processes of two partners after coordination and negotiation.

### 4.6 Application of Workflow Services

The approach discussed in this research can be extended to the service-oriented environment. In the service-oriented environment, an organization can release or advertise the public information of its local process view (interaction protocol, virtual workflow processes of expected partners) through workflow service platforms. Using the workflow service platforms, organizations can find collaboration partners that meet their requirements.

If there is a workflow service repository which contains a large number of local process views (public information) of organizations, the steps in the example scenario shown in Fig. 4.6 is described as follows:

1. Workflow coordinator A (user) interacts with the workflow service repository, provides requirement for collaboration;
2. The system searches the workflow service repository for potential partners for workflow coordinator A, and finds some possible partners for A based on its local process view description for collaboration requirement;
3. Workflow coordinator A analyzes the compatibility of the local process views of the listed potential partners and its own local process view based on the approach we proposed in Sect. 4.3;
(4) Based on the analysis result and original information of potential partners (failure probability, QoS, interaction cost and so on) provided by the workflow service repository, workflow coordinator A makes a favor ranking list for all potential partners;

(5) Workflow coordinator A chooses the potential partner with the highest favor ranking and coordinates with it. If the coordination fails, it can choose the partner with the second highest ranking and so on;

(6) After successful coordination, workflow coordinator A registers its revised local process view into the workflow service repository for further usage.

![Workflow Service Repository Diagram](image)

Figure 4.6: Find partners using workflow service repository

### 4.7 Summary

In the interorganizational workflow model proposed in Chapter 3, each organization has a local process view based on its own consideration. Therefore
there might be many conflicts between local process views of different organizations. In this research, we propose the coordination mechanism for different local process views to detect incompatibilities among organizations with compatibility analysis. The following two issues are addressed in this chapter.

**Coordinating different local process views** Since different local process views should be compatible with each other for execution, we further propose the coordination mechanism for different local process views to detect conflicts among organizations, which is realized by conducting compatibility analysis of local process views of different organizations. The compatibility analysis is divided into two main phases: coordination of interaction protocols, and coordination of local workflows. During this process, incompatibilities are expected to be detected and eliminated before workflow execution, and therefore a compatible interorganizational workflow that supports multiple local process views can be established.

**Validating the approach by case and implementation** The proposed interorganizational workflow model and coordination approach is demonstrated by a case study of collaborative software development collaboration. We also discuss the extension of this research to applications in the service-oriented environment by reusing workflows of partners.
Chapter 5

Interorganizational Workflow Execution Mechanism

In this chapter, we will discuss how to design the interorganizational workflow execution mechanisms when organizations reach agreement on the workflow for collaboration. This objective is achieved by designing two parts of execution mechanisms: the execution within the same organizations, and the execution among organizations. The approach is based on ECA rules and process agents to provide flexibility, adaptation and distribution [Lin 07b].

5.1 Introduction

Workflow management [Hollingsworth 95] has been widely adopted as an important technology to manage business processes. In recent years, with the global expansion of distributed computing environments, computer mediated collaboration has been increasing among organizations. In such cases, interorganizational workflow is expected to be created to cross organizational boundaries inside an enterprise or between enterprises [Aalst 99b].

There are several major problems in managing interorganizational workflow. For example, flexibility and adaptation are challenging is-
sues [Aalst 03a], especially when the execution is across organizations. Existing workflow management systems (WfMS) do not effectively address issues related to environments distributed across organizations. A central and monolithic workflow engine as suggested by Workflow Management Coalition (WfMC) reference model [Hollingsworth 95] is not sufficient to support the autonomy of enterprises. Multiagent system provides distributed platform, which has been regarded as a promising approach to solve many problems in workflow management [Singh 99]. However, it is rarely discussed in previous research that how to use agent technology to achieve the goal of supporting distribution and flexibility with adaptation and autonomy for interorganizational workflow execution.

To address these issues, this research proposes a framework for interorganizational workflow execution based on ECA (event-condition-action) rules and process agents. The whole interorganizational workflow is modeled as a multiagent system with one process agent for each organization. Therefore, process agents of organizations preserve autonomy of organizations with flexibility. We design general ECA rules to make workflow execution automatic with adaptation for different organizations. The framework will provide effective execution mechanisms for interorganizational workflow.

In the proposed framework, we divide the interorganizational workflow execution process into two parts: the *intra-execution*, which is execution within a same organization, and the *inter-execution*, which represents interaction between organizations. The *intra-execution* is distributed among organizations. Within each organization, there is an engine where a set of general ECA rules is defined for executing the internal workflow process. We use a method of transforming the graph-based workflow model into block-based workflow model [Bae 04, Mendling 05] to derive general rules from blocks. The execution of local workflow processes is controlled and monitored by process agents and rules. Rules are designed to control internal state transitions and process agents are used to control the external state transitions of tasks. Process agent of each organization interacts with that of other organizations to fulfill the *inter-execution*. Protocols are used to han-
dle interactions, which involve all the organizations that have interaction with each other to address specific purposes.

This chapter is organized as follows: Section 5.2 introduces related work. The overview of our approach is proposed in Sect. 5.3. Section 5.4 describes derivation of ECA rules. In Sect. 5.5, we explain process agent for controlling local workflow process and interacting among organizations. Section 5.6 is a case study and discussion of the proposed approach. The summary of this chapter is shown in the last section.

5.2 Related Work

ECA rules have been used for workflow execution in previous research. Casati et al. design various rules for workflow management using patterns and propose a classification of the rules [Casati 00], and present an approach of handling exception using ECA rules [Casati 99]. In [Bae 04], an automatic control mechanism of workflow execution is proposed, which combines traditional workflow process model and ECA rules to derive a general process control method. Those researches have demonstrated that ECA rules can control workflow automatically and deal with internal exceptions as well. Using ECA rules, workflow can be easily represented into executable forms. However, they mainly focus on internal process control and cannot provide flexibility in controlling task execution. Moreover, their approaches cannot deal with the issue of interorganizational workflow execution which this research addresses.

Multiagent technology has been used in different ways in workflow management [Singh 99], e.g., to fulfill particular roles that are required by different tasks, to serve as part of the infrastructure associated with WfMS. There is also some work that concentrates on agent negotiation between organizations [Andonoff 05a], however, how negotiation would affect the local workflows is not discussed. Different from existing work, our agent-based approach is combined with rule-based approach to deal with both control and monitor of tasks within organizations, and interaction among
5.3 Overview of the Framework

In previous research, van der Aalst proposes several types of workflow interoperability [Aalst 99b], among which the loosely coupled interorganizational workflow is common in the real world. The model that we discuss is based on loosely coupled interoperability, where the whole interorganizational workflow is made up of local workflows, which may be active in parallel over different organizations. We represent the interorganizational workflow model by using workflow graphs which provide a visual means for users to understand the semantics of the workflow process easily. A graph-based interorganizational workflow model is shown in Fig. 4.5 by the example of collaborative software development. There are two organizations, the outsourcer (partner A) and the supplier (partner B). There is a local workflow for each organization, which represents the process by tasks and the relations among the tasks, such as sequential relation, parallel relations (represented by AND-split and AND-join), iterative relation and so on. A complete workflow includes the detailed description of tasks, such as task executors (human, application or services), related data, scheduling and so on.

Figure 5.1 shows the overall conceptual framework of the execution mechanism of interorganizational workflow by our approach. The execution mechanism is divided into two parts: the intra-execution, which means the workflow execution is performed within the same organization, and the inter-execution, which represents the workflow execution (interaction) between organizations. The following three modules are included in intra-execution, among which the workflow process execution control module is the core issue of this research.

Workflow process definition and instance module. Process definition creates workflow schema. A process instance is the execution representation of a process and is created based on the process definition. A graph-based
workflow editor can be used for workflow definition.

Figure 5.1: Framework of interorganizational workflow execution

*Workflow process execution control module.* The execution of the workflow process instances is controlled by the process agent and ECA rules. We use ECA rules to describe all the state transitions of the tasks (e.g., ready, running, suspended, committed and so on) in the workflow process instance. Therefore, the workflow process can be automatically executed according to
the ECA rules. The execution of a rule in the ECA rule engine means a step of state change of a certain task and the whole process instance. An ECA rule is executed while the engine is triggered by an event under a certain condition, e.g., during the execution of the workflow process, the start of a task is always triggered by the execution completion of its predecessor task. Events in ECA rules include internal events that can directly trigger rules in the engine, and external events such as the start of a process instance, the success of executing a task and so on. We control the external events by the process agent. The process agent keeps the information of the execution status of the process instance by tracing the execution of ECA rules and monitoring the execution of tasks.

Task execution and monitor module. Tasks in workflow process instances are executed by some roles. The process agent dispatches a task when it gets the information that the task is ready for execution, and monitors the whole execution process of the task by interacting with human, applications or services (or their agents). By monitoring task execution, the process agent can get external events and then send such events to the ECA rule engine to trigger some new events.

As for inter-execution, interaction between process agents of the organizations is used. When process instance execution of an organization comes to an interaction point with another organization, the process agents of the two organizations interacts with each other to control the execution.

5.4 ECA Rules for Workflow Execution

Workflow execution can be described by task state transitions. Basic task states include Disabled, Ready, Running, Committed, Aborted and Suspended. The initial state of a task is Disabled, which is changed into Ready when its predecessor finishes execution, and then is further changed into Running if all conditions for execution are satisfied. After the successful execution of the task, its state turns into Committed. The state of a task can also be Aborted if it is aborted due to some failures, or Suspended if it is
suspended. The transitions of the tasks can be totally represented and executed automatically by ECA rules. ECA rules, with the form "on event if condition do action" specify to execute the action automatically when the event happens, provided the condition holds. As is shown in Fig. 5.2, we have an ECA rule engine which contains a set of rules based on the state transition of tasks for each organization.

In the approach we present in this research, we first create graph-based interorganizational workflow, through which each organization can understand the workflow process. Further, a set of ECA rules is established based on the graph-based workflow as a rule engine which describes and conducts the whole execution of the workflows. ECA rule engines are distributed among the organizations because each organization needs to have its own engine to execute the local workflow. The execution status of the workflow can be displayed to each organization through the user interface. In this sec-

Figure 5.2: ECA rules and workflow execution
tion, we mainly present how to create ECA rules based on the graph-based interorganizational workflow which is described in Sect.5.3.

5.4.1 Blocks in the Workflow Process Model

Workflow specification can be understood from a number of different perspectives. In this research, we focus on the control-flow (process) perspective because this is the essential perspective of the workflow specification [Aalst 03a]. It describes information about tasks and the execution orders (dependencies between tasks). This research mainly deals with basic constructs of sequence, iteration, splits (AND, OR and XOR) and joins (AND, OR and XOR). Based on general constructs, Gokkoca et al. [Gokkoca 97] defines seven types of block, namely, serial, and parallel, or parallel, xor parallel, contingency, conditional and iterative blocks. By this means, we can transform a graph-based workflow into the block-based workflow. The transformation from graph-based workflow to block-based workflow is similar to approaches in previous researches [Casati 00, Mendling 05]. Take the local workflow of the supplier in Figure 1 for example, the graph-based workflow process can be transformed into following blocks: (1) and parallel block: \( P1 = (\text{basic function implementation} \& \text{interface implementation 1}) \); \( P2 = (\text{function implementation} \& \text{interface implementation 1}) \); (2) iteration block: \( I1 = (\text{Condition(modified)}; \text{phase report}; \text{system modification}) \); (3) serial block: \( S1 = (\text{prototype implementation planning}; P1; \text{initial report}; \text{implementation planning}; P2; \text{system integration}; I1) \). From above transforming example, we can observe that the block detection always begins from the most inner part of the workflow process. Moreover, blocks might be mutually embedded. Therefore, the whole workflow process can always be transformed to a single block with other blocks embedded, e.g., the above local workflow is finally turned into a serial block \( S1 \). We transform the graph-based workflow process into block-based workflow process to use the semantics of blocks to derive general ECA rules and automate the workflow execution.
5.4.2 Deriving ECA Rules

Since our proposed workflow execution mechanism is mainly based on state transitions of tasks, it is necessary to define the transactions for the state transition diagram in Fig.5.2. Therefore we use following transaction primitives: Enable, Begin, Commit, Suspend, Resume, Abort. Further, we define the history base $H$ in the ECA rule engine, which is a finite set of transactions that have occurred during workflow execution. Events in ECA rules include internal events and external events. Internal events can be described by transaction primitives. External events involve the operations during task execution, including START and END of the process instance, READY, DISPATCH, STOP, SUCCESS, FAIL to execute tasks. Therefore, ECA rules can be divided into internal ECA rules and external ECA rules. By using transaction primitives and history base, we describe internal ECA rules according to the semantics of blocks in workflow process. Similarly, we use semantics of external events to derive external ECA rules. Internal rules can be executed inside the ECA rule engine without interaction with task execution and monitor module, however, external rules need to interact with the process agent to get the external events.

External ECA rules are listed as follows. We list external ECA rules for each external event in Table 5.1.

$$ER_1: \text{on } START(PI) \text{ if null do } Enable_B$$

$$ER_2: \text{on } Enable_{xi} \text{ if } (READY(x_i)) \land (Enable_{xi} \in H) \text{ do Begin}_{xi}$$

$$ER_3: \text{on } Suspend_{xi} \text{ if } (READY(x_i)) \land (Suspend_{xi} \in H) \text{ do Resume}_{xi}$$

$$ER_4: \text{on } STOP(x_i) \text{ if } (Begin_{xi} \in H) \text{ do } Suspend_{xi}$$

$$ER_5: \text{on } SUCCESS(x_i) \text{ if } (Begin_{xi} \in H) \text{ do Commit}_{xi}$$

$$ER_6: \text{on } FAIL(x_i) \text{ if } (Begin_{xi} \in H) \land \neg (Commit_{xi} \in H) \text{ do } Abort_{xi}$$
\( ER_7 : \) on \( \text{Begin}_{x_i} \) if \( (\text{Begin}_{x_i} \in H) \) do \( \text{DISPATCH}(x_i) \)

\( ER_8 : \) on \( \text{Resume}_{x_i} \) if \( (\text{Resume}_{x_i} \in H) \) do \( \text{DISPATCH}(x_i) \)

\( ER_9 : \) on \( \text{Commit}_{B} \) if \( (\text{Commit}_{B} \in H) \) do \( \text{END}(PI) \)

### Table 5.1: External events and ECA rules

<table>
<thead>
<tr>
<th>Event</th>
<th>Semantic Description</th>
<th>External ECA Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{START}(PI) )</td>
<td>Start of process instance enables the block that represents whole workflow process.</td>
<td>( ER_1 )</td>
</tr>
<tr>
<td>( \text{READY}(x_i) )</td>
<td>Ready of an enabled task causes the start of task execution; ready of a suspended task causes the resume of that task.</td>
<td>( ER_2 \ ER_3 )</td>
</tr>
<tr>
<td>( \text{STOP}(x_i) )</td>
<td>Stop of a running task makes it suspend.</td>
<td>( ER_4 )</td>
</tr>
<tr>
<td>( \text{SUCCESS}(x_i) )</td>
<td>Success of executing a task makes it commit.</td>
<td>( ER_5 )</td>
</tr>
<tr>
<td>( \text{FAIL}(x_i) )</td>
<td>Failure of executing a task makes it abort.</td>
<td>( ER_6 )</td>
</tr>
<tr>
<td>( \text{DISPATCH}(x_i) )</td>
<td>The start or resume of a task makes it dispatched for execution.</td>
<td>( ER_7 \ ER_8 )</td>
</tr>
<tr>
<td>( \text{END}(PI) )</td>
<td>Commit of the block that represents the whole workflow process makes the end of the process instance.</td>
<td>( ER_9 )</td>
</tr>
</tbody>
</table>
Internal ECA rules are listed as follows. We list internal ECA rules for each block in Table 5.2 and 5.3.

\[ IR_1 : \text{on Enable}_B \text{ if } (\text{Enable}_B \in H) \text{ do } \text{Enable}_x \]

\[ IR_2 : \text{on Commit}_{x_i} \text{ if } (\text{Commit}_{x_i} \in H) \text{ do } \text{Enable}_{x_{i+1}} \]

\[ IR_3 : \text{on Commit}_{x_n} \text{ if } (\text{Commit}_{x_n} \in H) \text{ do } \text{Commit}_B \]

\[ IR_4 : \text{on Abort}_{x_i} \text{ if } (\text{Abort}_{x_i} \in H) \text{ do } \text{Abort}_B \]

\[ IR_5 : \text{on Enable}_B \text{ if } (\text{Enable}_B \in H) \text{ do } \text{Enable}_{x_i} \]

\[ IR_6 : \text{on Commit}_{x_i} \text{ if } \forall i (i \in n) (\text{Commit}_{x_i} \in H) \text{ do } \text{Commit}_B \]

\[ IR_7 : \text{on Commit}_{x_i} \text{ if } (\text{Commit}_{x_i} \in H) \text{ do } \text{Commit}_B \]

\[ IR_8 : \text{on Abort}_{x_i} \text{ if } \forall i (i \in n) (\text{Abort}_{x_i} \in H) \text{ do } \text{Abort}_B \]

\[ IR_9 : \text{on Commit}_{x_i} \text{ if } (\text{Commit}_{x_i} \in H) \text{ do } (\text{Commit}_B) \land (\forall j (j \neq i) \text{Abort}_{x_j}) \]

\[ IR_{10} : \text{on Abort}_{x_i} \text{ if } (\text{Abort}_{x_i} \in H) \text{ do } \text{Enable}_{x_{i+1}} \]

\[ IR_{11} : \text{on Abort}_{x_n} \text{ if } (\text{Abort}_{x_n} \in H) \text{ do } \text{Abort}_B \]

\[ IR_{12} : \text{on Enable}_B \text{ if } (\text{Enable}_B \in H) \land C_i \text{ do } \text{Enable}_{x_i} \]

\[ IR_{13} : \text{on Abort}_{x_i} \text{ if } (\text{Abort}_{x_i} \in H) \land C_i \text{ do } \text{Abort}_B \]

\[ IR_{14} : \text{on Commit}_{x_i} \text{ if } (\text{Commit}_{x_i} \in H) \land C_i \text{ do } \text{Commit}_B \]

\[ IR_{15} : \text{on Commit}_{x_n} \text{ if } (\text{Commit}_{x_n} \in H) \land C \text{ do } \text{Enable}_{x_1} \]
$IR_{16} : \text{on } Commit_{x_n} \text{ if } (Commit_{x_n} \in H) \land \neg C \text{ do } Commit_B$

Table 5.2: Internal ECA rules for blocks (1)

<table>
<thead>
<tr>
<th>Block Type</th>
<th>Semantic Description</th>
<th>Internal ECA Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>serial</td>
<td>$B = (x_1; x_2; \ldots; x_n)$</td>
<td>$IR_1\ IR_2\ IR_3\ IR_4$</td>
</tr>
<tr>
<td></td>
<td>Tasks are executed consecutively. A</td>
<td></td>
</tr>
<tr>
<td></td>
<td>task is enabled when its prior task</td>
<td></td>
</tr>
<tr>
<td></td>
<td>succeeds. Block aborts if any task</td>
<td></td>
</tr>
<tr>
<td></td>
<td>aborts.</td>
<td></td>
</tr>
<tr>
<td>and parallel</td>
<td>$B = (x_1 &amp; x_2 &amp; \ldots &amp; x_n)$</td>
<td>$IR_5\ IR_6\ IR_4$</td>
</tr>
<tr>
<td></td>
<td>Tasks are executed concurrently.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Block is completed provided completion</td>
<td></td>
</tr>
<tr>
<td></td>
<td>of all tasks. Block fails if any task</td>
<td></td>
</tr>
<tr>
<td></td>
<td>fails.</td>
<td></td>
</tr>
<tr>
<td>or parallel</td>
<td>$B = (x_1</td>
<td>x_2</td>
</tr>
<tr>
<td></td>
<td>Block succeeds if there exists one</td>
<td></td>
</tr>
<tr>
<td></td>
<td>task that succeeds in executing.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Block fails if all tasks abort.</td>
<td></td>
</tr>
<tr>
<td>xor parallel</td>
<td>$B = (x_1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If there is one task that completes,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>block succeeds and all the other</td>
<td></td>
</tr>
<tr>
<td></td>
<td>tasks abort.</td>
<td></td>
</tr>
</tbody>
</table>

5.5 Process Agent in the Interorganizational Workflow Model

In this research, ECA rules are designed based on general constructs in graph-based workflow model, which can be applied in different workflow processes and adaptive to be specialized according to different process def-
<table>
<thead>
<tr>
<th>Block Type</th>
<th>Semantic Description</th>
<th>Internal ECA Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>contingency</td>
<td>( B = (x_1, x_2, \ldots, x_n) ) Each task has a priority. Task with the highest priority executes first. If it fails, then task with the next highest priority will execute. Block succeeds if any task completes.</td>
<td>( IR_1 ), ( IR_{10} ), ( IR_7 ), ( IR_{11} )</td>
</tr>
<tr>
<td>conditional</td>
<td>( B = ((C_1, x_1) \mid (C_2, x_2) \mid \ldots \mid (C_n, x_n)) ) All tasks have conditions compared with or parallel block. Only the task that satisfies the condition executes.</td>
<td>( IR_{12} ), ( IR_{13} ), ( IR_{14} )</td>
</tr>
<tr>
<td>iterative</td>
<td>( B = (Condition(C); x_1; x_2; \ldots; x_n) ) The iterative condition gives a while loop between the start task and the end task on serial block. The loop continues until the iterative condition becomes false or any task aborts.</td>
<td>( IR_1 ), ( IR_2 ), ( IR_{15} ), ( IR_{16} ), ( IR_4 )</td>
</tr>
</tbody>
</table>

Definitions. Therefore, in an interorganizational workflow, rules can be used in local workflow for all organizations. Moreover, general ECA rules do not necessarily need modification even when process definition is dynamically changed during execution. It is also very convenient to modify rules in the engine. However, to design general ECA rules for workflow process, workflow data and role model are not defined in rules. Therefore, we use process agent to handle the external events control of each workflow process instance for intra-execution. Process agent is also required to interact with process agents of other organizations to provide inter-execution. For a
loosely coupled interorganizational workflow, process agents are suitable to provide a distributed platform because agents are loosely coupled components forming an open system.

The usage of process agents has advantages for interorganizational workflow execution. First, within each organization, process agent controls and monitors the execution of tasks, which provides flexibility to the local workflow execution. Process agent can dynamically control the external execution status of tasks according to the change of environment. Process agent can also dispatch tasks to suitable execution roles according to the strategies. Second, process agents facilitate cooperation among organizations, which preserves the autonomy of organizations. In our approach, we use standard interaction protocols for interoperability among organizations, which is essential for interorganizational workflow. Therefore, organizations cooperate with each other even if they have different process definition tools. By this means, local workflows can be executed distributedly and autonomously.

5.5.1 Process Agent in Intra-Execution

Within an organization, process agent is in charge of controlling workflow process instance. To achieve this goal, process agent is required to interact with ECA rule engine by external events to continue process state transition. Moreover, process agent needs to dispatch tasks that are ready for execution and monitor tasks that are in execution. Process agent for intra-execution of an organization includes following actions.

1. Start the workflow process instance when the organization is ready, create an external event $START(PI)$ and send it to the ECA rule engine.

2. Check the conditions of tasks that are enabled for execution, create an external event $READY(x_i)$ if execution condition of an enabled task is satisfied. $READY(x_i)$ is required for an enabled task to start.

3. Dispatch tasks that are ready for execution (when receiving the external event $DISPATCH(x_i)$). Executors of tasks might be pre-defined, negotiated or dynamically adjusted and discovered.
(4) Monitor task execution and receive execution event. Since tasks are executed by executors (human, applications or services), process agent needs to interact with executors or their execution agents to acquire detailed information of task execution. Therefore, process agent can create external events of execution results such as $SUCCESS(x_i)$, $FAIL(x_i)$ and $STOP(x_i)$. Then, it sends the events to ECA rule engine to trigger new events. Different execution results need different actions. $SUCCESS(x_i)$ might trigger execution of succeeding task, $FAIL(x_i)$ might need some exception handling process, and if $STOP(x_i)$ occurs, the executing task needs to wait to be resumed.

(5) Finish the workflow process instance if all the tasks are successfully executed (when receiving the external event $END(PI)$).

5.5.2 Agent Interaction Protocols for Inter-Execution

Interorganizational workflow environments can be modeled as multiagent systems. Within each organization, agent remains autonomy and heterogeneity. Across organizations there are interactions, which can be regarded as agent interaction problem. We use protocol to handle interactions that take place during workflow execution. The protocols involve all the organizations that have interaction with each other to address specific purposes, e.g., sending and receiving software specification between organizations in an offshore software development environment.

To achieve flexibility among organizations, a process agent is required not only to deal with basic interaction such as sending or receiving results, but also to support coordination and negotiation between organizations if the results need to be modified. Autonomous agents can negotiate with each other to execute the allocated task flexibly and dynamically [Jiang 05]. In [Jennings 01], several negotiation protocols are proposed such as auction protocol, heuristic protocol, argumentation protocol, contract net protocol and so on. As for the intercultural software development issues, contract protocol is most suitable since it deals with cooperative negotiation [Andonoff 05a]. Figure 5.3 shows the examples.
The actions of process agent interaction among organizations would affect intra-execution. Figure 5.4 shows scenarios of how process agent interaction (as shown in Fig. 5.3) affects intra-execution. Simple agent interactions are execution conditions for certain tasks and the interaction actions trigger ECA rules in intra-execution. However, interaction with negotiation will cause internal state changes of certain tasks and therefore workflow adaptation should be conducted by process agent for further negotiation process. For example, in protocol (b) in Fig. 5.3, when the Outsourcer process agent sends `requestRevise()`, the Supplier process agent would change its local workflow to repeat executing current task with updated conditions. In some cases, a part of workflow process might be changed in the process of negotiation according to the strategies of process agents.
5.6 Case Study and Discussion

We study the case of offshore software development to explain the proposed approach. First, a graph-based interorganizational workflow model is created as shown in Fig.4.5. Next, the graph-base workflow of each organization is automatically transformed into the block-based workflow. Then, the block-based local workflows of all the organizations are executed distributedly. The local workflows interact with each other at certain points by process agent interaction protocols. Table 5.4 shows part of workflow execution steps of the Supplier. As is presented in Sect.5.4.1, the block-based workflow contains four blocks: $S_1$, $P_1$, $P_2$ and $I_1$, among which $S_1$ is the main stream. In the table, every step shows the current active block and tasks, current event, the ECA rule that the event would trigger, and the action of process agent. The event history stores all the executed events. Process

---

*The tasks in the table are as follows. $x_1$: prototype implementation planning; $x_2$: P1 (regarded as a "task" in S1); $x_{21}$: basic function implementation; $x_{22}$: interface implementation 1*
agent of the local workflow process instance interacts with process agent from other organizations when there is an interaction point. By the control of ECA rule engine and process agent, the whole interorganizational workflow can be distributed for execution among organizations.

Further, we explain the inter-execution by the process agent interaction of *send initial report* in Fig.4.5. The whole process is as follows. With coordination or negotiation, the whole interorganizational workflow execution becomes flexible.

(1) When process agent of the Supplier $PA_S$ knows commitment of the task *initial report* by the ECA rule engine, it sends initial report to process agent of the Outsourcer $PA_O$.

(2) $PA_O$ receives initial report from $PA_S$ and replies to $PA_S$ according to the received initial report. If the received report is OK, $PA_O$ will send a confirmation to $PA_S$. Otherwise, it will send a revise request or explanation request to $PA_S$ with requirement.

(3) If $PA_S$ receives a confirmation, the READY condition of its next task *implementation planning* will be satisfied and a new rule will be triggered. The interaction steps end.

(4) If $PA_S$ receives a revise or explanation request from $PA_O$, it will adjust the local workflow process and intra-execution state. The specification and state of task *initial report* will be changed for re-execution according to the received request.

(5) Repeat from step (1).

In our approach, we design ECA rules based on general definition of blocks to provide an adaptable and modular approach to achieve workflow execution. The general ECA rules can be adopted in different workflows because they are based on blocks. Further, in order to control and monitor the whole workflow execution process more flexibly, we introduce agent technology into our mechanism. We use process agent to control the execution of each task in a local workflow. Though the task dispatch mechanism is not the focus of this research, process agent can provide flexibility in dealing with dynamic changes and exceptions. We also use agent interaction
Table 5.4: Interorganizational workflow execution example

<table>
<thead>
<tr>
<th>Block</th>
<th>Task</th>
<th>Event</th>
<th>ECA</th>
<th>Action of Process Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Confirm receiving specific from process agent of Outsourcer.</td>
</tr>
<tr>
<td>1</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>Start process instance, send $START(PI)$ to ECA rule engine.</td>
</tr>
<tr>
<td>2</td>
<td>–</td>
<td>$START(PI)$</td>
<td>$ER_1$</td>
<td>–</td>
</tr>
<tr>
<td>3</td>
<td>$S_1$</td>
<td>–</td>
<td>$IR_1$</td>
<td>–</td>
</tr>
<tr>
<td>4</td>
<td>$S_1$</td>
<td>$x_1$</td>
<td>$Enable_{S_1}$</td>
<td>$ER_2$</td>
</tr>
<tr>
<td>5</td>
<td>$S_1$</td>
<td>$x_1$</td>
<td>$Begin_{S_1}$</td>
<td>$ER_7$</td>
</tr>
<tr>
<td>6</td>
<td>$S_1$</td>
<td>$x_1$</td>
<td>$DISPATCH(x_1)$</td>
<td>–</td>
</tr>
<tr>
<td>7</td>
<td>$S_1$</td>
<td>$x_1$</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>8</td>
<td>$S_1$</td>
<td>$x_1$</td>
<td>$SUCCESS(x_1)$</td>
<td>$ER_5$</td>
</tr>
<tr>
<td>9</td>
<td>$S_1$</td>
<td>$x_1$</td>
<td>Commit($x_1$)</td>
<td>$IR_2$</td>
</tr>
<tr>
<td>10</td>
<td>$S_1$</td>
<td>$x_2$</td>
<td>$Enable(x_2)$</td>
<td>$IR_5$</td>
</tr>
<tr>
<td>11</td>
<td>$P_1$</td>
<td>$x_{21}$</td>
<td>$x_{22}$</td>
<td>$Enable_{x_{21}}$</td>
</tr>
<tr>
<td>12</td>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

protocols to deal with the interactions between different organizations so that organizations can not only send basic message to each other, but also
manage to coordinate and negotiate with each other. As a result, the whole mechanism can provide automation, flexibility, and adaptation to the execution of interorganizational workflow. Moreover, our approach supports distributed execution of local workflows of different organizations, which provides autonomy of organizations.

Despite the effectiveness, there are some limitations in the proposed approach. Although ECA rules designed in this research can be generally used, the assumption is that workflow model is graph-based. When designing the internal ECA rules, we only deal with basic types of blocks. How to extend the proposed approach to other types of workflow models and workflow model with more complicated structure might be interesting issues in future research. Moreover, although the agent-based approach provides autonomy and flexibility, there are still open issues that we have to deal with in future work. For example, when process agents are conducting coordination, some criteria like workflow soundness and optimal modification must be considered for local workflow adaptation.

5.7 Summary

Interorganizational workflow has been becoming more and more important in the global economic environment. However, there are several major challenges in the interorganizational workflow execution problem, including flexibility, adaptation and distribution. These issues are rarely covered together in previous research. In this research, we propose a new framework for the interorganizational workflow execution based on process agents and ECA rules to address above issues.

To design the execution mechanism of interorganizational workflow, we divide the whole workflow execution into two parts: the *intra-execution* and the *inter-execution*. We design execution mechanisms for the two parts to achieve the goal. In this approach, the whole interorganizational workflow is modeled as a multiagent system with a process agent in each organization. Therefore, local workflows of organizations are distributed for execution.
and interaction with each other at certain points. For each local workflow, a method of transforming the graph-based workflow model into block-based workflow model is applied to derive general ECA rules from blocks. We further design ECA rules to control internal state transition and use process agent to control the external state transition of tasks in the local workflow process. Workflow execution across organizations is achieved by process agent interaction protocols.

The proposed approach can provide automatic execution of interorganizational workflow with flexibility and adaptation. It can also provide autonomy for local workflows. A case study of offshore software development is provided to demonstrate the effectiveness of our approach. Our future work will be the implementation and evaluation of the proposed approach based on process agents and ECA rules.
Chapter 6

Interorganizational Workflow Model Implementation

In this chapter, we will propose an interorganizational workflow collaboration methodology, considering coordination of different local process views and implementation of the whole interorganizational workflow model. The proposed approach is illustrated by a case study of collaborative software development which is realized in a business process management system.

6.1 Introduction

As has been discussed in Chapter 3, there have been various types of business process modeling methods in previous work, including the purely graphical models, mathematical models, object oriented models, independent process modeling models and so on [Specht 05, Vergidis 08]. In collaborative business environments, enterprises cooperate with each other with resources and business processes within them to achieve common goals. Interorganizational workflow is modeled for the collaboration and cooperation across organizational boundaries [Aalst 00, Divitini 01, Shen 01, Aalst 03b, Liu 03, Chiu 04, Andonoff 05b, Chebbi 06].

However, most of the previous models are of limited impact on practical application due to the lack of formal methodological backgrounds and igno-
rance of proven collaboration frameworks. Moreover, when it comes to the collaborative business processes among organizations, flexible management and privacy preservation might be main issues that should be considered beyond the existing concepts for business process management. Therefore, it is extremely important to design interorganizational workflow collaboration models that take into consideration the flexibility, privacy preservation, and the issue of how to implement such models by using or extending common business process management framework. To address above issues, we consider the following issues in this chapter.

Interorganizational workflow collaboration approach In the interorganizational workflow model, since each organization has its own local process view, it is necessary to develop the approach of interorganizational workflow collaboration from coordinating local process views of different organizations to implementing the whole model. To realize this process, we divide the collaboration process into two phases: bottom-up coordination and top-down implementation.

Validating the approach by case study To validate the proposed interorganizational workflow collaboration approach, it is necessary to apply in real-world case with implementation. In this paper, we take collaborative software development workflow for instance.

This chapter is organized as follows: Section 6.2 provides related work. In Sect.6.3, the interorganizational workflow collaboration approach is introduced. A case study and discussion of collaborative software development for the proposed approach is given in Sect.6.4 and 6.5. Section 6.6 is the summary of the chapter.

6.2 Related Work

As has been discussed in Chapter 2, previous approaches of interorganizational workflow mainly include organization contracts, workflow inheritance, workflow cooperation and composition, and so on. Those attempts
belong to either top-down (from global process to local parts) [Grefen 01, Aalst 02a] or bottom-up (from local parts to global process) [Chebbi 06] design approach. However, the top-down approach cannot support modern business environment well especially the service-oriented architecture, where each organization always has its own business process (services) and has its own consideration about a potential interorganizational collaboration. Moreover, current bottom-up approach does not address the important issue of how to implement the interorganizational workflow model. Therefore, to consider the support of service-oriented environment and the implementation issues, we propose an interorganizational workflow collaboration approach by combining the bottom-up and top-down methods.

In this research, we apply the ARIS business process management platform to illustrate interorganizational workflow collaboration. ARIS is the abbreviation of Architecture of Integrated Information Systems, which is a well-known approach to enterprise modeling of IDS Scheer AG [Scheer 92]. It started as the academic research, and now has an explicit industrial background, which is not only a standard, but a product widespread. In addition to the high level architectural framework, it is a business modeling method, which is supported by a series of tools, including the strategy platform, design platform, implementation platform and controlling platform. It is intended to serve various purposes: documentation of existing business process types, blueprint for analyzing and designing business processes and support for the design of information systems. The design platform is for distributed modeling, simulation, optimization, and publishing of business processes and managing IT architectures, which can provide answers to the following questions: Who does what, in which order, what is achieved, and what software systems are deployed? This makes it possible to identify organizational, structural, and technical problems in workflows and to unlock improvement potential. Process models can be published worldwide for role-based access. The design platform is intended for system designers. In designing business process, it has an elaborated decomposition of an enterprise in several views: data view, function view, organization view, product/service view and, to realize the connection between these views,
the process view (also known as control view) [Scheer 92, Scheer 00]. In this research, we mainly concentrate on the process view. To implement the proposed interorganizational workflow model based on local process views, we adopt the business process modeling system ARIS Business Architect in the ARIS design platform, which mainly focuses on the modeling design and administration.

In modern collaborative business environments, organizations do not create a single global view for interorganizational workflow collaboration. Instead, each organization has its own local process view as we propose in Chapter 3. Since we will implement the interorganizational workflow collaboration model using the ARIS platform of IDS Scheer AG [Scheer 00], in the definition of local process view (Definition 3.1), for any partner $i$, $WF_i$ can be modeled as standard process models in ARIS design platform, such as EPC (event-driven process chain) diagram [Aalst 99a], value-added chain diagram [Kirchmer 04] and so on. The Interaction Sequence Chart can be modeled as part of the e-business scenario diagram [Kirchmer 04], which is also a standard type of model in ARIS design platform.

6.3 Interorganizational Workflow Collaboration Steps

In the collaborative business environment, each organization has its own consideration, understanding and preference about the whole collaboration work, which can be represented using the model based on local process views we propose in Chapter 3. Therefore, there are two phases for interorganizational workflow collaboration. In the first phase, organizations coordinate with each other about their different local process views, which is a bottom-up process. In the second phase, organizations implement the interorganizational workflow model based on the agreement they reach in the first phase, which is a top-down process.
6.3.1 Phase 1: Bottom-up Coordination

The phase of coordination deals with how organizations reach agreement with each other from different local process views, which can be described as following steps.

(1) Design local process views: each organization designs its local workflow, interaction protocols and virtual workflows of its partners based on their own requirement and consideration. Organizations make the parts of interaction protocols and virtual workflows of its partners public, while keeping the local workflow processes as privacy;

(2) Coordinate interaction protocols: organizations exchange and compare the interaction protocols (Interaction Sequence Charts) designed by partners with their own interaction protocols. If there are conflicts, they negotiate and coordinate to reach an agreement on the interaction protocols between organizations. Then, they modify interaction protocols and the related parts of their local workflow processes in the local process views;

(3) Coordinate local workflow processes: organizations compare the workflow part with their partners by (a) exchanging public virtual workflow processes with each other and compare the local workflow and its related virtual workflow of the local organization designed by its partners; (b) exchanging comparison results of workflow processes and negotiate to reach an agreement on the workflow processes;

(4) Modify local process views: organizations revise their local process views according to the coordination results and save as new workflow for reuse in further collaboration.

6.3.2 Phase 2: Top-down Implementation

After coordination and negotiation of different local process views, it can be assumed that organizations have reached agreement of interaction protocols and mutual understandings of local workflows. Therefore, the interorganizational workflow model based on local process views should be further implemented. The implementation of the model includes the following three
steps.

(1) **Establish overall organizational structures**: in the collaborative business environment, there are always different roles from multiple organizations involved. Thus, it is necessary to model the organizations and the organizational units that are related to the interorganizational workflow. Roles for members and groups in each organization can also be defined in this step;

(2) **Form common interaction part among organizations**: in this step, the collaboration among organizations can be derived from the compatible interaction protocols (Interaction Sequence Charts) and virtual workflow processes after coordination and negotiation in phase 1. Organizations cooperate to design the interaction scenario of the whole collaborative business. To ensure the flexibility of workflow integration and dynamic changes, each organization only includes part of the necessary high level processes of its local workflow that have interaction with other organizations;

(3) **Build local process models of organizations**: after designing the organization structure and common parts among organizations, the local process models are defined in this step. Each organization designs its local process model distributively in details based on the coordination results in phase 1. Business processes are normally modeled using a multi-level procedure. Starting with the top process hierarchy, processes are detailed consecutively. This detailing is usually carried out over several hierarchy levels using subprocesses for different levels of members within organizations. The interaction parts with other organizations are also designed in details in the local model based on the high level parts designed in the former step.

### 6.4 Case Study

To describe the implementation of interorganizational workflow model based on local process views, we introduce an example of collaborative software development by different partners. In the example scenario in this research, there are three teams (organizations) that collaborate with each
other to conduct the software development, namely the Alpha Team, Beta Team and Gamma Team. Each team is responsible for different part of the whole collaboration: Alpha Team mainly conducts the design and evaluation, Beta Team does the coding job and implementation, and Gamma Team focuses on the testing process. Figure 6.1 shows a global function and interaction view of interorganizational workflow collaboration scenario for the case with three organizations.

In our interorganizational workflow approach based on local process
views, there is no pre-defined global view. Instead, each organization has a local process view at the beginning. The local process views designed by three organizations might be different. For example, in the design team, software specifications are modified in parallel with development. However, the understanding of implementation team is that specifications are fully designed before development. To detect those conflicts, compatibility analysis of different local process views is required as we describe in phase 1 of Sect.6.3. The detailed compatibility analysis mechanism has been discussed in Chapter 4. Three teams are expected to reach an agreement of interaction protocols and local workflows by compatibility analysis and negotiation. After coordination of local process views of different organizations in phase 1, we can apply the top-down implementation steps described in Sect.6.3 using the ARIS design platform in phase 2 as follows.

First, we design the organization structure of the collaborative software development. Since there are three organizations involved, a main business house and three sub business houses are designed for this purpose. The main house means the common business model (interaction parts) which involves all the three teams. In the main house, the five views (organization, data, process, function and product/service) include the business models that are related to the common part of all the organizations. For example, in the organization view, the members or groups that are related to the interaction parts between organizations in the three teams are involved. The process view links all the other four views together. The process view of the main business house will be defined in step 2 of phase 2. The three sub business houses represent the local business models of the three teams that will be designed in step 3 of phase 2.

Second, the common part of the three organizations as the process view in the main business house designed in step 1 of phase 2 is formed using the e-Business Scenario Diagram, which is derived from the coordinated Interaction Sequence Charts and public virtual workflow processes in phase 1. Figure 6.2 is the screenshot of designing the common interaction process of all the organizations in the business process modeling system. The interaction process starts from the Alpha Team. Alpha Team sends the spec-
Figure 6.2: Design of common interaction part among organizations

ification of the software to Beta Team after finishing Development Analysis subprocess. Beta Team will then conduct the initial Implementation and interacts with Alpha Team with initial report after finishing the subprocess. The three organizations interact with each other in this way until reaching the Software Release subprocess in the Beta Team. The diagram is designed by the involved organizations and expresses how the organizations interact with each other. All the elements here (the subprocesses, data, organizations) have links with the business model in the local organizations. For example, Development Analysis is an interaction subprocess here, but it is
also a subprocess in the local workflow process of Alpha Team. The de-
tailed information of the elements of each organization will be designed in
step 3. Comparing the interaction part in Fig.6.2 with the process of Design
Team in Fig.6.1, we can see that the Development Analysis of Alpha Team
in Fig.6.2 is a high level subprocess which includes four lower level subpro-
cesses in Fig.6.1, namely Requirement Analysis, Function Design, Interface
Design and Specification Creation. Also, not all the subprocesses of Design
Team in Fig.6.1 is open to the public in the interaction part in Fig.6.2. This
is simply because only interaction subprocesses of all the organizations are
designed in the interaction part as the common processes.

Finally, local process models of the organizations are designed distribu-
tively. The business models of Alpha Team, Beta Team and Gamma Team
will be designed in the three sub business houses in step 1 of phase 2. In
each sub house, the five views are designed according to the business case
of each organization. Here we give an example of designing the business
model in Alpha Team and we focus on the process view. In the business
process modeling system, processes are designed by multi-level structures.
The modeling system ARIS Business Architect provides various types of
models for designing process view. As shown in Fig.6.3, to design the top
level main process of Alpha Team, we use the value-added chain diagram.
The main process of Alpha Team includes four subprocesses, namely Devel-
opment Analysis, Prototype Analysis, Initial Implementation Analysis and
System Evaluation. After the main process is designed, we further design
each subprocess in more details. Figure 6.4 shows a second level process
which described the subprocess Development Analysis of Alpha team us-
ing event-driven process chain (EPC) diagram, which is compatible with
the function view of Design Team in Fig.6.1. In Fig.6.4, there are also sev-
eral subprocesses that need to be designed in more details in the third level
process of Alpha Team. Using the multi-level design approach, the process
view of Alpha Team can be designed until the atom tasks. By this means,
we can also design the process views of Beta Team and Gamma Team.

By adopting above steps, we can implement the interorganizational
workflow model in example of collaborative software development process
Figure 6.3: Top level main process of Alpha team

Figure 6.4: Parts of a second level process (Development Analysis)

completely for the global view in Fig.6.1.
6.5 Discussion

In this section, we discuss how the proposed approach is effective with the issues of flexibility and privacy preservation, and also how to extend current business process management framework to support collaborative business.

6.5.1 Workflow Flexibility

The proposed interorganizational workflow collaboration approach provides flexibility of management in the following aspects. First, since each organization has its own local process view in the coordination phase, it is flexible for organizations to design the whole collaboration based on its understanding. Second, when designing the common interaction parts in the implementation phase, very little information of the organizations is required. For example, in Alpha Team, only a part of subprocesses in the top level main process are modeled as interaction processes. Therefore, when there are dynamic changes in the collaboration process, the local workflow will be less affected than in the situation that all information of local workflows are shared with other organizations as in some traditional interorganizational workflow models. Moreover, the multi-level design method provides various granularities for business model; therefore it provides different members inside an organization with different business views according to their roles. Finally, organizations can configure and manage local workflows more flexible strategies due to the distribution and autonomy of the local workflows.

6.5.2 Privacy Preservation

In our approach, organizations do not open their local workflows to other organizations in the coordination phase, instead they design virtual workflows for other organizations based on their own understandings. In the implementation phase, access of process groups can be controlled by designing user access control mechanisms in the business process modeling system. For each process group, users in the whole system can be controlled by man-
aging the read, write and delete permissions of users. If there are several organizations in the business process modeling system, we can define the access permission for each process group. When organizations access the process server, they can only access the parts that they have the access right. Therefore, using this property, we can manage interorganizational workflow while preserving the privacies of organizations. Local process views of each organization are different because of the different access rights to the whole workflow process. Take the case of collaborative software development for example, we can control access rights for different users in different groups. Table 6.1 shows an example of the access rights of users and groups of the ARIS Network (r,w,d mean the right of read, write and delete respectively).

In Table 6.1, A.Alpha Team means the information group of the Alpha Team;

<table>
<thead>
<tr>
<th>User Group</th>
<th>User</th>
<th>A.Alpha Team</th>
<th>D. Interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alpha Team</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alpha_Level1</td>
<td>r-</td>
<td>r-</td>
</tr>
<tr>
<td></td>
<td>Alpha_Level2</td>
<td>rwd</td>
<td>rwd</td>
</tr>
<tr>
<td></td>
<td>Alpha_Level3</td>
<td>r-</td>
<td>r-</td>
</tr>
<tr>
<td>Beta Team</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Beta_Level1</td>
<td>—</td>
<td>r-</td>
</tr>
<tr>
<td></td>
<td>Beta_Level2</td>
<td>—</td>
<td>rw-</td>
</tr>
<tr>
<td>Gamma Team</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Gamma_Level1</td>
<td>—</td>
<td>r-</td>
</tr>
<tr>
<td>Level 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alpha_Level1</td>
<td>—</td>
<td>rwd</td>
</tr>
<tr>
<td></td>
<td>Beta_Level1</td>
<td>—</td>
<td>rwd</td>
</tr>
<tr>
<td></td>
<td>Gamma_Level1</td>
<td>—</td>
<td>rwd</td>
</tr>
<tr>
<td>Level 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alpha_Level2</td>
<td>—</td>
<td>rw-</td>
</tr>
<tr>
<td></td>
<td>Beta_Level2</td>
<td>—</td>
<td>rw-</td>
</tr>
<tr>
<td>Level 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alpha_Level3</td>
<td>—</td>
<td>r-</td>
</tr>
</tbody>
</table>
D.Interaction means the information group pf the common interaction part. The group of users in Alpha Team has the access right of the common interaction process and the local business model of Alpha Team, but they do not have the rights to access the business model of Beta Team and Gamma Team. Even within a same organization, there are different levels of users who have different rights to access the local model and common part. By this means, we can achieve the privacy preservation of organizations in a collaborative business environment.

6.5.3 Extensions

Besides workflow flexibility and privacy preservation, our approach support workflow reuse since each organization has a local process view including not only local workflows but also interaction information that can be preserved as a workflow service. However, there are still something that need to be extended to support the whole business process management lifecycle. For example, the coordination of organizations to reach an agreement on the common interaction scenario cannot be well supported in current concept in the strategy stage. Moreover, in a global business environment, the issue of how to support and analyze the different cultural influences in controlling stage is still to be worked out. Therefore, the current ARIS Business Process Management Lifecycle could be extended to a new lifecycle to support collaborative business in a more effective way. In this research, we mainly focus on the design stage, modeling and implementing the interorganizational workflow collaboration in the concept of global view and local views. Such concept can also be used in all the stages (strategy, design, implementation and controlling) in business process management lifecycle, which will be our focus in future work.

6.6 Summary

In this chapter, we propose an interorganizational workflow collaboration approach based on local process views for collaborative business consider-
ing real practice. We provide a methodology for interorganizational workflow collaboration by two phases: the bottom-up coordination phase and the top-down implementation phase. The coordination phase consists of the following steps: (1) design local process views, (2) coordinate interaction protocols, (3) coordinate local workflow processes, and (4) modify local process views. The implementation phase is composed of steps of (1) establish overall organizational structure, (2) form common interaction part among organizations, and (3) build local process models of organizations. We study a case of collaborative software development process to illustrate the proposed collaboration approach. The approach proposed in this research is effective in several aspects. First, the interorganizational workflow model based on local process views provides flexibility in managing the local workflows of organizations. Next, organizations can preserve privacy of their local workflows by designing access control mechanisms in the business process modeling system. Moreover, the proposed approach can be implemented by extending commercial business process management systems.
Chapter 7

Conclusion

7.1 Contributions

In this thesis, we have proposed a novel methodology for interorganizational workflow modeling and collaboration considering requirements of organization and workflow design. The contributions of this thesis are summarized as follows:

(1) Interorganizational Workflow Modeling and Coordination

In interorganizational environments, due to different cultural backgrounds, participants from different organizations always have different views of modeling workflow process. Moreover, organizations require to preserve privacy and autonomy of its own workflow from other organizations. Therefore, workflow inter-visibility is expected to be as little as collaborations need. Here, the contributions include the following two aspects.

i. Modeling interorganizational workflow based on local process views

An interorganizational workflow approach is provided for organizations to model collaborative business based on their own consideration of the whole process, and coordinate their unique views with partners to reach mutual understanding and compatibility. We proposed an interorganizational workflow model, taking different pro-
cess modeling views of different organizations into consideration. In this model based on local process views, organizations do not share a pre-defined global interorganizational workflow process, but have their own understandings of how to model the whole workflow process with their unique backgrounds and knowledge. The local workflows are not designed to be shared among organizations. Organizations can manage its own local workflow processes autonomously. The proposed interorganizational workflow model for collaboration has the following features: consideration of different workflow modeling views of different organizations, privacy preservation of organizations and flexibility of local workflows.

ii. Coordinating interorganizational workflow using compatibility analysis mechanisms

Since different local process views should be compatible with each other for execution, we proposed the coordination mechanism for different local process views to detect conflicts among organizations, which is achieved by conducting compatibility analysis of local process views of different organizations. The compatibility analysis is divided into two main phases: coordination of interaction protocols, and coordination of local workflows. During this process, incompatibilities are expected to be detected and eliminated before workflow execution, and therefore a compatible interorganizational workflow that supports multiple local process views can be established. The proposed interorganizational workflow model and coordination approach is demonstrated by a case study of collaborative software development collaboration. We also discuss the extension of this research to applications in the service-oriented environment by reusing workflows of partners.

(2) Designing Interorganizational Workflow Execution Mechanisms

A new framework for the interorganizational workflow execution based on process agents and ECA rules is proposed to address the issues of flex-

102
ibility, adaptation and distribution. To design the execution mechanism of interorganizational workflow, we divide the whole workflow execution into two parts: the *intra-execution* and the *inter-execution*. We design execution mechanisms for the two parts to achieve the goal. In this approach, the whole interorganizational workflow is modeled as a multiagent system with a process agent in each organization. Therefore, local workflows of organizations are distributed for execution and interaction with each other at certain points. For each local workflow, a method of transforming the graph-based workflow model into block-based workflow model is applied to derive general ECA rules from blocks. We further design ECA rules to control internal state transition and use process agent to control the external state transition of tasks in the local workflow process. Workflow execution across organizations is achieved by process agent interaction protocols. The proposed approach can provide automatic execution of interorganizational workflow with flexibility and adaptation. It can also provide autonomy for local workflows. A case study of collaborative software development is provided to demonstrate the effectiveness of our approach.

**(3) Implementation of Interorganizational Workflow Collaboration**

An interorganizational workflow collaboration approach based on local process views is proposed for collaborative business which can be realized in business process management systems. Since each organization has its own local process view, the approach of interorganizational workflow collaboration is developed from coordinating local process views of different organizations to implementing the whole model. The whole process is described by two phases: the bottom-up coordination phase and the top-down implementation phase. The coordination phase consists of four steps: (1) design local process views, (2) coordinate interaction protocols, (3) coordinate local workflow processes, and (4) modify local process views. The implementation phase is composed of three steps: (1) establish overall organizational structure, (2) form common interaction part among organizations, and (3) build local process models of organizations. Further, a case study of collaborative software development process is provided to illustrate the
effectiveness of the proposed collaboration approach. First, the interorganizational workflow model based on local process views provides flexibility in managing the local workflows of organizations. Next, organizations can preserve privacy of their local workflows by designing access control mechanisms in the business process modeling system. Moreover, the proposed approach can be implemented by extending commercial business process management systems.

7.2 Future Directions

Our future research directions of modeling and coordination in interorganizational workflow are listed below.

- **Negotiation in Interorganizational Workflow**
  In this thesis, we mainly concentrate on modeling and coordination in interorganizational workflow. We proposed the compatibility analysis of local process views for coordination. However, the compatibility analysis process can only detect the conflicts of different views of different organizations. The issue of how to eliminate such conflicts is not the focus in this thesis. Therefore, it is necessary to design negotiation mechanisms for organizations to reach agreement on conflicts of workflows. This research issue can be regarded as a multiagent negotiation problem [Kraus 95, Kraus 97, Jennings 00, Kraus 01].

- **Organizational Interaction with Business Protocols**
  In the interorganizational workflow modeling of this thesis, we use interaction sequence chart to define the interaction among organizations, which is based on the message sequence charts [Mauw 94]. Although we have extended the message sequence charts to support interaction points which include related data, it is still not enough to express the interaction protocols when coordinating and executing the interorganizational workflow. Therefore, it is important to
develop formal protocols for business interaction in the interorganizational workflow, considering the modeling, coordination and execution [Desai 04, Desai 05].

- **Interorganizational Business Process Management System Development**
  
The proposed interorganizational workflow model and implementation approach in this thesis provides flexibility and privacy preservation for organizations in the business design stage. However, design is only one step in the whole business process management (BPM) lifecycle. In the ARIS methodology, BPM lifecycle includes strategy, design, implementation and controlling [Scheer 00]. In the IBM Websphere BPM concept for service-oriented architecture, the lifecycle consists of modeling, assembly, deployment and management [Ferguson 05]. Therefore, it is necessary to extend our research to support the whole business process management lifecycle.

- **Applications in Service-oriented Environment**
  
Service-oriented computing [Papazoglou 03, Singh 05] is becoming the prominent paradigm for distributed computing and e-commerce, which makes developers create applications through using services as fundamental elements. The research in this thesis is expected to be extended in the service-oriented environment. In the service-oriented architecture, there are always many services available for a certain function part. Therefore, multiple organizations (service providers) might be involved. In such cases, organizations always need to find appropriate partners for collaboration. Coordination of organizations is necessary for such processes and also for eliminating incompatibilities in the service composition process. Therefore, it is important to extend the model and mechanisms in this thesis for the service-oriented environment to provide scalability of the whole research.
Bibliography


110


Publications

Major Publications

Journals


International Conference


Workshops

1. **Donghui Lin**, Yasumasa Mita, and Toru Ishida, “Compatibility Analysis of Local Modeling Views in Interorganizational Workflow,” in
AAMAS Workshop on Coordination of Inter-Organizational Workflow: Agent and Semantic Web based Models (CIOW-06), 2006.

Other Publications

Chapter in Book


International Conference