

Master Thesis

**Compatibility Analysis of
Interorganizational Workflow**

Supervisor Professor Toru Ishida

Department of Social Informatics
Graduate School of Informatics
Kyoto University

Yasumasa MITA

February 9, 2006

Compatibility Analysis of Interorganizational Workflow

Yasumasa MITA

Abstract

In collaborative tasks among different organizations, each organization achieves assigned tasks by following a fixed set of processes. During the execution of processes, some interaction among organizations may arise to achieve common goals. In such cases, a pre-defined global interorganizational workflow, which can be shared by all organizations, is needed. However, each organization may conceal sensitive information about their local workflow, limiting the organizations to agree only on interaction protocol or only on role of each organization. As a result, details of workflow are not shared among organizations when carrying out collaborative tasks, leading to disagreements and friction in participating organizations. This paper seeks to detect incompatibilities of processes by comparing each organization's interorganizational workflow prior to the execution of collaborative tasks. The research issues are following .

(1) Modeling of interorganizational workflow

In previous research, various models of interorganizational workflow such as those using Petri Nets and XML have been discussed. However, these models deal with analysis or expression of only one interorganizational workflow, and cannot be applied to the comparison of multiple interorganizational workflows. A new method of modeling interorganizational workflow, which compares numerous organizations' workflows to analyze compatibility, must be considered.

(2) Compatibility analysis considering autonomy of the organizations

To check incompatibilities among different modeling views, following requirements must be considered. (a)Autonomy of each organization: The organizations can conceal some part of their local workflow to protect their privacy. (b)Cooperation among different organizations: Cooperation is needed to achieve agreements on the interorganizational workflow modeling. With these requirements in mind, a method of checking incompatibilities must be considered.

(3) Comparison between concrete workflow and abstract workflow

Interorganizational workflow is composed of multiple local workflows of participating organizations. Therefore, in order to compare multiple local modeling

views of interorganizational workflow, each local workflow is compared. While each organization can describe complete details of its own local workflow, it can only describe abstract workflow of other organizations due to limited knowledge. Since it is difficult to compare incomplete abstract workflows, there needs to be a way to compare multiple workflows of differing levels of detail.

To solve the above issues, this paper proposes the following method of analyzing workflow compatibility.

(1)Local modeling view of interorganizational workflow

The local modeling view for compatibility analysis consists of following elements. (a)Concrete Workflow : this represents the detailed workflow within the local organization and consists of some tasks and ordering constraints. (b)Abstract Workflow : this represents the workflow of interacting organizations, and consists of abstract tasks and ordering constraints. (c)Interaction Protocol : this consists of messages which are exchanged between organizations and their orderings.

(2)Sharing of the abstract workflow and interaction protocol

In compatibility analysis, the abstract workflow and interaction protocol are shared among all the organizations. In this way, compatibility analysis can be performed without exposing the local workflow to other organizations.

(3)Block-structure transformation of concrete workflow

We devised a mechanism to directly compare the abstract workflow and concrete workflow. In this mechanism, the concrete workflow is transformed into block-structured workflow, which has abstract tasks and simple structure. By creating hierarchies of workflows to compare abstract workflow and block-structured workflow, the problem of abstract-concrete workflow comparison is solved.

Using the above method, we proposed an algorithm that can check the incompatibility of following elements.

- interaction protocol among individual organizations
- tasks defined in local workflow executed locally by individual organization
- ordering constraints of tasks in local workflows executed locally by individual organization

組織間ワークフローの整合性分析

三田 泰正

内容梗概

一般的に組織間の協調作業では、一定のプロセスにしたがって各組織が割り当てられたタスクを実行し、組織間でインタラクションを行うことで目標を達成する。したがって作業手順を規定する組織間ワークフローについて、組織間で合意が形成されている必要がある。しかし一方で、各組織は内部で実行するワークフローの詳細を開示することができないという制約を持つために、実際は組織間のインタラクションや各組織の役割に関する合意に止まっている。したがって、ワークフローの詳細に関する合意の形成は行われぬまま協調作業が実行され、このことはさまざまな摩擦が発生する要因となっていた。そこで本研究は、協調作業を実行する前に各組織が考える組織間ワークフローを比較することで、プロセス定義における不整合を検出することを目的とする。

そのために、以下の問題について考える。

(1) 組織間ワークフローのモデルの提供

既存研究では、単一の組織間ワークフローの解析や表現を目的としたモデル化が行われてきた。例として、ペトリネットやXMLによるモデル化が挙げられる。しかしながら、これらのモデルは単一の組織間ワークフローを対象としているために、複数のワークフローの比較には適していない。ワークフローの比較に基づく整合性分析を目的とした、組織間ワークフローのモデルを考える。

(2) 組織の独立性を考慮した不整合の検出

不整合を検出する際には、次のような相反する2つの要求を考えなければならない。(a) 各組織の独立性：各組織は自らが実行するワークフローの一部を、必要に応じて他の組織に対して秘匿する必要がある (b) 組織間の協調：組織間ワークフローについて合意の形成を行うためには組織間で協調する必要がある これらの要求を考慮した不整合の検出方法について、考える必要がある。

(3) 詳細レベルの異なるワークフローの比較

組織間ワークフローは、各組織がローカルで実行するワークフローの相互接続によって構成される。したがって、組織間ワークフローの比較を行う際には、各組織が実行するワークフローに分割した上で比較を行う。しかしながら、各組織が設計するワークフローは詳細レベルが異なるために、直接比較することは

難しい。各組織は、自ら実行するワークフローに関しては詳細な設計が可能だが、他の組織が実行するワークフローに関しては断片的な知識しか持たないため、概略を表す抽象的な表現しかできない。詳細レベルの異なるワークフローの比較方法について考える必要がある。

上記の問題に対して、以下のような整合性分析の方法を考えた。

(1) 抽象的な表現を含む組織間ワークフロー

タスクとその依存関係に着目し、次のような構成要素を含む組織間ワークフローのモデルを考えた。(a) ローカルワークフロー：ローカルで実行する詳細なワークフローであり、タスクとその順序関係によって規定される。タスクにはパラメータとして、前提条件と効果が設定される。(b) 概略ワークフロー：他の組織によって実行される抽象的なワークフローであり、抽象的なタスクとその順序関係によって規定される。(c) 組織間のインタラクション：組織間で交換するメッセージとその順序関係によって規定される。各組織がこのようなモデルによる組織間ワークフローを設計することで、整合性分析を行う。

(2) 概略ワークフローおよびインタラクションの交換による整合性分析

設計した組織間ワークフローのビューに含まれる構成要素のうち、各組織が想定している他の組織が内部で実行するワークフローと、組織間のインタラクションを組織間で交換する。整合性の分析は、他の組織によって設計された概略ワークフローを自身が設計したローカルワークフローと比較することで行う。これにより各組織がローカルで実行するワークフローを外部に公開することなく、整合性分析を行うことが可能になった。

(3) ワークフローの階層化

詳細レベルが異なるワークフローの直接比較を可能にするために、ワークフローを段階的に抽象化するアルゴリズムを考えた。階層化によって、抽象的なタスクが生成されるとともに、複雑な構造を持つワークフローが木構造に単純化される。ワークフローの階層化を行うことで、詳細レベルが異なるワークフローを直接比較することが可能になった。

以上の方法で、組織間ワークフローの次のような不整合の検出が可能になった。

- 組織間のインタラクションのプロトコルの不一致
- 各組織がローカルで実行するワークフローに含まれるタスクの不一致
- 各組織がローカルで実行するワークフローに含まれるタスク間の順序制約の不一致

Compatibility Analysis of Interorganizational Workflow

Contents

| | | |
|------------------|--|-----------|
| Chapter 1 | Introduction | 1 |
| Chapter 2 | Interorganizational Workflow | 4 |
| 2.1 | Types of interorganizational workflow | 4 |
| 2.2 | Loosely Coupled Interorganizational Workflow | 6 |
| 2.3 | Design of Interorganizational Workflow Process | 7 |
| 2.4 | Related Works | 10 |
| Chapter 3 | Modeling | 12 |
| 3.1 | The Local Modeling View | 12 |
| 3.2 | Concrete Workflow, Abstract Workflow | 13 |
| 3.3 | Interaction Protocol | 17 |
| Chapter 4 | Compatibility Analysis | 20 |
| 4.1 | Incompatibility of Multiple Local Modeling Views | 20 |
| 4.2 | Algorithms of Compatibility Analysis | 22 |
| 4.2.1 | Event Comparison | 24 |
| 4.2.2 | Block-structure Transformation of Workflow | 24 |
| 4.2.3 | Algorithm of Block-structure Transforming of Workflow | 29 |
| 4.2.4 | Comparison of Abstract Workflow and Block-structured Workflow | 32 |
| Chapter 5 | Implementation | 38 |
| 5.1 | Implementation System | 38 |
| 5.1.1 | System Module | 38 |
| 5.1.2 | System Interface | 39 |
| 5.2 | Case Study | 42 |
| 5.2.1 | Local Modeling Views of two organizations | 42 |
| 5.2.2 | Compatibility Analysis | 43 |
| Chapter 6 | Discussion | 48 |
| Chapter 7 | Summary | 50 |

| | |
|-----------------|----|
| Acknowledgments | 52 |
| References | 53 |

Chapter 1 Introduction

With the global expansion of Internet and distributed computing environments, computer mediated collaborative tasks have been rapidly increasing among different organizations i.e. offshore development, e-commerce, outsourcing and so on. In these works, each organization executes assigned tasks and cooperates with other organizations with resources within them to achieve common goals. The definite process to execute collaborative tasks is called interorganizational workflow.

To execute these works smoothly and efficiently, it is important to share the global view of interorganizational workflow among all the organizations. In other words, all the organizations have to achieve the agreements about the interorganizational workflow. However, it is difficult to construct the detail interorganizational workflow that is shared by all the organizational because of following reasons.

Protection of privacy of the local workflow

The organizations cannot disclose sensitive information about their local workflow to other organizations. A part of the local workflow need to be concealed.

Difference of workflow modeling views

Participants from different organizations have different cultural backgrounds and knowledge, which might have great impacts on the behaviors of conducting cooperative works. Therefore, different organizations may consider different interorganizational workflow. For example, Indian software companies have found they need to approach communication with U.S. and Japanese clients in very different ways. U.S. client companies normally work with extensive written agreements and explicit documentation, reinforced with 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[4].

Because of such reasons, it is difficult to achieve complete agreements about

the interorganizational workflow among all the organizations, and the organizations execute their tasks with their own interorganizational workflow. However, many problems occur due to the disagreements on the task orderings or process modeling views. Such problems are serious in such collaborative tasks that participants have very different culture background from each other i.e. intercultural collaboration. To solve these problems, the process modeling views of each organization must be compared, and checked the incompatibilities of them.

Previous research efforts mainly focus on the process execution rather than process modeling in the interorganizational workflow. In those researches, there is normally a pre-defined global view that can be shared by all the organizations and the problem that how all the organizations could reach a common global view is always ignored. The research issue of supporting multiple modeling views of different organizations under different knowledge backgrounds has seldom been covered.

In this work, we choose the loosely coupled interorganizational workflow process as the interoperability type. In the loosely coupled interorganizational workflow, each organization takes care of a special part of the process. Based on the loosely coupled interoperability, to model interorganizational workflow considering the different modeling views of different organizations, we propose a new method of analyze compatibility of interorganizational workflow. An approach for compatibility analysis is following.

- Modeling of the process modeling view of interorganizational workflow

Because of such constraints that the organizations cannot disclose the part of their own local workflow to other organizations, they do not have complete knowledge about the workflow which is executed by other organizations. However, the organizations have partly knowledge about the workflow which is executed by other organizations, and they describe their own process using these knowledge. In other words, each organization has own view of the workflow within other organizations. To express these partial knowledge, the process view of each organization can be expressed, which we call local modeling view. The local modeling view consists of the concrete workflow within each organization, the abstract workflow which is

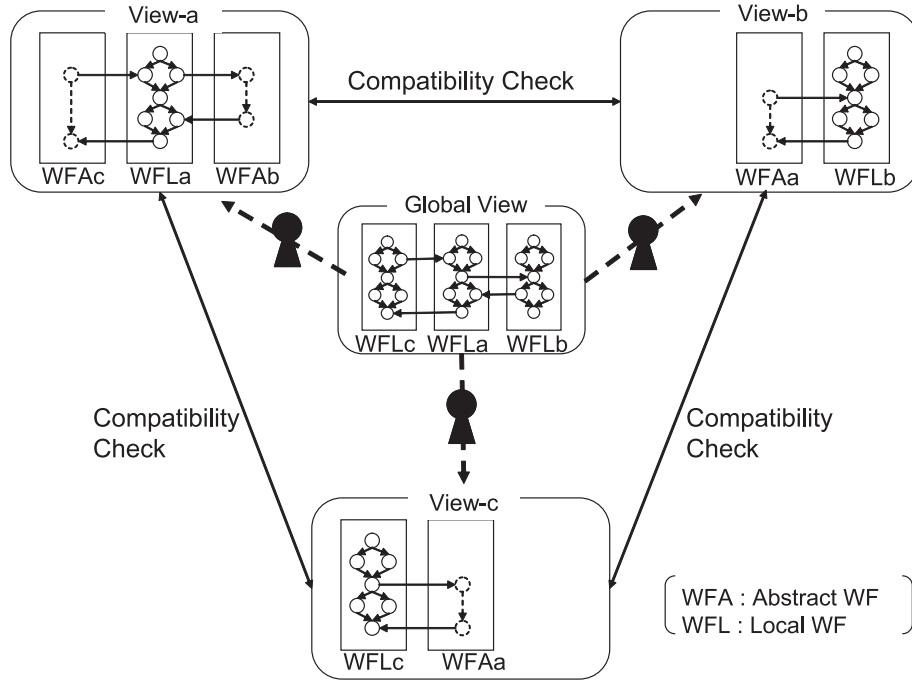


Figure 1.1: Concept of this research

executed by other organizations and the interaction protocol.

- Detection of incompatibility by comparison of multiple local modeling view
To detect incompatibility, different local modeling views are compared. In detection of incompatibility, constraints that each organization cannot expose its own local workflow to other organizations must be considered. Therefore, the abstract workflow and interaction protocols are shared among all the organizations. To detect incompatibilities, the abstract workflow and concrete workflow are compared, and different interaction protocols are compared.

The above approach is shown in Figure1.1 . The research issue is to detect incompatibilities of multiple local modeling views of different organizations.

This paper is organized as follows: Chapter 2 describes about interorganizational workflow. In Chapter 3 , definition of the local modeling view is presented. In Chapter 4 , detailed compatibility analysis mechanism is presented. Implementation System and a case study of China-Japan offshore software development is given in Chapter 5. In Chapter 6 , discussion about compatibility analysis method is presented, followed by the summary in Chapter 7.

Chapter 2 Interorganizational Workflow

In these days, workflow management system is introduced for the purpose of automation and increase in efficiency of business process in many organizations. Workflow is the automation and management of the process, documents and information which are exchanged among participants in order to execute collaborative tasks efficiently. Recently, there are a lot of computer mediated collaborative tasks among distributed organizations, so introduction of workflow for the purpose of support interorganizational collaboration are increase. Such mutual cooperation of a workflow is called interorganizational workflow.

There are many types of interorganizational workflow models. In this chapter, the interorganizational workflow model which we consider is presented. Furthermore, problems about the method of modeling interorganizational workflow process and our approach are described.

2.1 Types of interorganizational workflow

As indicated in the introduction, this paper focuses on interorganizational workflows. From a technical point of view these issues are also of concern to the WfMC. The WfMC provides various types of workflow interoperability. These types address the technical issue. However, these interoperability standards do not address the content of the coordination structure.

W.M.P. van der Aalst categorizes several types of workflow interoperability in the interorganizational workflow process. Each type of interoperability is as following.

- Capacity sharing

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

- Chained execution

The second form of interoperability is chained execution. In this form of interoperability, the whole workflow process can be split into a number of disjunct subprocesses. Each subprocess is executed by different organizations in a sequential order. This form of interoperability requires that an organization transfers or initiates the flow for a case after completing all the tasks. In contrast to capacity sharing, the control of the workflow is distributed over the organizations.

- Subcontracting

The third form of interoperability is subcontracting. In this form of interoperability, there is one organization which subcontracts subprocesses to other organizations. In the workflow process which is executed by top-level organization, the subprocesses appear to be atomic. For the other hand, in the workflow process which is executed by other organizations, the subprocesses can be very complex. In this form of interoperability, the control of the workflow is hierarchical, i.e., although there is a top-level organization, the control is distributed in a tree-like fashion.

- Case transfer

The fourth form of interoperability is case transfer. In this form of interoperability, the global interorganizational workflow is shared by all the organization. However, at any time, each case resides at exactly one organization. Process instances can be transferred from one organization to another.

- Loosely Coupled

The last form of interoperability is loosely coupled. In this form of interoperability, the whole process is made up of several pieces which may be active in parallel over different organizations. The definition of each of the subprocesses is local, and the organizations do not know the local process. Therefore, the global interorganizational workflow cannot be shared by all the organizations. Only the protocol(interaction event) which is used to communicate is public for other organizations.

The research goal is to analyze different local modeling views of interor-

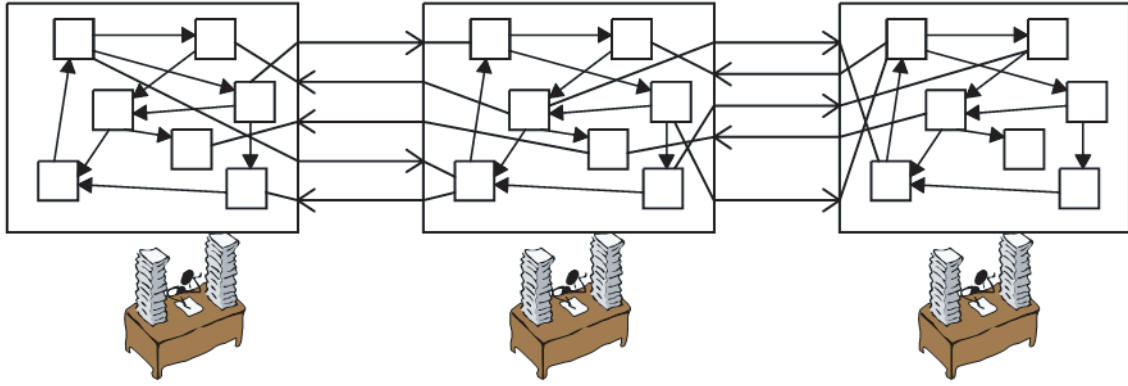


Figure 2.1: Loosely coupled[4]

organizational workflow. Which forms of interoperability are useful in this research? Capacity sharing is the only form of interoperability which assumes that there is one centralized workflow manager, so the control of the workflow is not distributed. Therefore, it is not suitable for this research because the workflow manager has the global interorganizational workflow and each organization does not need the knowledge about the workflow process. In the form of Case transfer, all the organization share the global interorganizational workflow. Therefore, there is no incompatibility among the organizations and this form of interoperability does not considered by this research. In the other three forms of interoperability, each organization can construct its own local subprocess. However, subcontracting can only be applied in an environment with clearly hierarchical structured organizations. In the form of chained execution, organization structure is straightforward and there is little interaction. Therefore, these types of interoperability cannot be considered by this research. This paper focus on the remaining architecture, loosely coupled, because there are more interesting from a research point of view.

The remainder of this paper focuses on loosely coupled workflow processes. Figure2.1 illustrates this form of interoperability.

2.2 Loosely Coupled Interorganizational Workflow

In the traditional loosely coupled interorganizational workflow process modeling, there are several pieces which may be active in parallel over different or-

ganizations, and exchange messages and information among organizations. In this form of interoperability, multiple organizations can exchange information closely and frequently, and altitude cooperation can be realized.

The characteristics of loosely coupled interorganizational workflow are as following.

- The whole workflow process is made up of several workflows which is executed by each organization.
- Each organization has no knowledge about the workflow process within other organizations.
- Multiple local workflow can be connected with interaction among organizations.

Loosely coupled interorganizational workflow consists of the local workflow which is executed by each organization and the interaction events among the organizations. Each organization has knowledge only about the own local workflow and the interaction events and has no knowledge about the workflow of other organizations. In the execution of collaborative tasks, each organization executes its own local workflow and exchange interaction events between other organizations.

2.3 Design of Interorganizational Workflow Process

To design of interorganizational workflow, two requirements which are conflict each other must be considered: Autonomy and cooperation of the organizations. These requirements are as following.

- **Autonomy of each organization**

Although each organization cooperates with each other, each organization does not be interfered by other organizations about its own local workflow. In other words, each organization has the authority to design the local workflow independently. This character of organizations is called “autonomy”. Because of this character, the local workflow of each organization is controlled and managed by each organization independently and locally.

- **Cooperation among different organizations**

Although each organization has autonomy, the interaction protocols must

be defined properly in designing of the interorganizational workflow process. From this point of view, different organizations should cooperate with each other rather than design independently. Because the local workflow which each organization executes locally is mutually independent, each organization can design its local workflow independently. However, the knowledge about the interaction protocols should be shared among all the organizations. Therefore, the cooperation among different organizations is indispensable.

Because of two above opposite requirements, the organizations have designed gradually the interaction protocols and the local workflows. The gradual design of the interorganizational workflow is performed by the following procedures. First, detailed local workflow which each organization performs is not clarified, but designs only the interaction protocols. Second, each organization designs the local workflow based on the interaction protocols which all the organizations share. This means the hierarchization of design the interorganizational workflow.

In this procedure, when each organization designs the local workflow, the knowledge shared among the organizations is only the interaction protocols, and the knowledge about the local workflow which other organizations execute is not shared. In other words, in design of the interorganizational workflow, the organizations do not disclose their local workflow to other organizations. This is because the contents which conflict with the privacy of each organization are included in the local workflow, and each organization has to conceal such contents. Therefore, the organizations execute the collaborative tasks without the complete knowledge about the workflow which other organizations execute, and the interorganizational workflow is not shared among all the organizations. Because different organizations have different modeling views of the interorganizational workflow, each organization executes the collaborative tasks according to its modeling view.

However, when a collaborative task is executed according to different interorganizational workflows, many problems occur due to the disagreements on the task orderings or process modeling views. Here, we explain this problem

using two following examples.

Case1: offshore software development

In offshore software development between Japanese company and Chinese company, there are some conflicts caused by the different customs and understandings over software development, which leads to the difference of local modeling views of the whole workflow process. For example, following conflicts always happen in the offshore software development between the Japanese company and the Chinese company.

- In the Japanese company, software specification is modified in parallel with development. However, in the Chinese company, specification is fully designed before development. Therefore, Chinese company often gets flustered by the Japanese way of development.
- In the Japanese company, software quality management is more severe than that in the Chinese company. Therefore, in the Chinese company, software test is conducted less frequently than in the Japanese company. Japanese company is often dissatisfied with the result of software test from Chinese company.

Generally, each organization does not consider such different views in advance before starting the software development. As a result, conflicts always occur during the whole cooperative work of software development between them.

Case2: electronic commerce

In electronic commerce, the workflow process includes the ordering, billing and shipping of products, involving the customer, the supplier, and the shipper and so on. There are some conflicts caused by the different understandings over purchase order, which leads to the difference of local modeling views of the whole workflow process. For example, following conflicts often happen in the purchase order between the consumer and supplier or also the consumer and consumer.

- Suppose a following case, a consumer considers that the payment should be executed after he receives the products and a supplier considers that the products should be sent after the payment is completed. In such case, deadlock arise in this collaborative task caused by the difference of ordering constraints between “payment” and “delivery”.

- Suppose another case, a consumer considers that the payment should be executed after delivery of products and invoice, but a supplier considers that only delivery of the products should be executed before payment. In such case, “invoicing” is included in one workflow process and is not included in the other workflow process.

2.4 Related Works

There are a lot of previous research about interorganizational workflows. W.M.P. van der Aalst[19][18][14] provides a method of analysis of interorganizational workflows. These researches formalized interorganizational workflows using Petri-Nets[13], and proposed the model of interorganizational workflow which consists of multiple workflow nets connected by interaction between the organizations. Aalst proposed the character of interorganizational workflows which must be required: soundness. On the other hand, this research does not aim at the analysis of the single interorganizational workflow, but the comparison and analysis of multiple different interorganizational workflows.

The analysis of the loosely coupled interorganizational workflow has been suggested by W.M.P. van der Aalst[15][16]. In those researches, there is normally a pre-defined global view that can be shared by all the organizations. In contrast, our research does not consider a pre-defined global view and tries to analyze compatibility among different process views for the purpose of achieving the agreements about the global interorganizational workflow.

Several formal interorganizational workflow process modeling techniques are suggested by [12][5]. In these researches, the interorganizational workflow is defined by XML. XML can design the interorganizational workflow in detail rather than Petri-Nets. These researches aim only at modeling of interorganizational workflows, and the analysis of interorganizational workflows is ignored.

The efficient process execution and management of interorganizational workflows was proposed by several researches such as [3][8][9]. However, in those researches, there is a pre-defined global interorganizational workflow shared by all the organizations, and the analysis of multiple process modeling views was ignored.

Several researches suggested the cooperation among the organizations and the method of sharing interorganizational workflows. In those researches such as [7][10], the local workflow which each organization executes is transformed into the abstract workflow, and these abstract workflows are shared by the organizations. Such process abstraction is performed by replacing multiple tasks by the single virtual task. Therefore, the organizations can share the knowledge about abstract tasks but concrete tasks. This means that this process abstraction approach cannot reveal incompatibilities of concrete tasks and their orderings. In contrast, our research provides the approach to detect incompatibilities of tasks and their orderings in the interorganizational workflow.

Effective design of collaborative processes in virtual enterprises is important[11]. In this research[11], the interaction protocols and public tasks are shared by all the organizations. The organizations design the own local workflows based on shared knowledge, and integrate those local workflows into the interorganizational workflow. However, the compatibility analysis of the interorganizational workflow is ignored.

In the area of planning in multiagent system, how to make the whole multiagent plan is important. In partial global planning[2], Durfee proposed the approach that the agents exchange the abstract plan with each other. Each agent has own view of the local plan and the abstract plan which is the abstraction of the local plan, and sends the abstract plan to other agents. Krogt[21] proposed the planning method used Graphplan[1], which the agents cooperate to construct the whole plan. In these multi agent planning research, agents share the knowledge about actions and operator within plans. However, in the design of interorganizational workflows, organizations do not share the complete knowledge about tasks and their orderings. Therefore, the problem of design the whole workflow is more difficult than in the area of agent planning.

Chapter 3 Modeling

In this paper, we do not create single global process view of the interorganizational workflow. Instead, each organization has its own local modeling view. To analysis compatibility of interorganizational workflow, these local modeling views are compared. In this chapter, the definition of the local modeling view is presented.

3.1 The Local Modeling View

This paper suggests the method of compatibility analysis of the loosely coupled interorganizational workflow. In this form of interoperability, each organization mainly has the following three parts of knowledge as following.

1. **Local concrete workflow**

This part contains the full information of the local workflow.

2. **Sharable knowledge base**

Each organization has a local sharable abstract knowledge base which contains belief information of the concrete workflow. In the local knowledge base, some private parts of workflow are concealed. The local knowledge base of an organization is open to other organizations in order to make them have some knowledge about the shared organization.

3. **Abstract workflow of other organizations**

The local organization has its own view of the workflow of other organizations due to its special cultural background. Therefore, the abstract workflow of other organizations is built based on the shared abstract knowledge bases form other organizations and the modeling view of the local organization.

Following are a series of formalism definitions of the interorganizational workflow in support of multiple local modeling views.

Definition1(Local Modeling View)

A local modeling view of loosely coupled interorganizational workflow is as following.

$$LC - WF = (I, WFL, WFA_1, WFA_2, \dots, WFA_n, EVENT)$$

1. I is a set of the organizations;
2. WFL is the local concrete workflow (in Definition2), which we call "concrete workflow";
3. For each $k \in 1, 2, \dots, n$, WFA_k is an abstract workflow of the organization that has interaction with the local organization (in Definition3). Because of the lack of knowledge about the workflow which other organizations execute, each organization cannot describe this workflow in detail. Therefore, we call these workflow "Abstract workflow". These workflows contain some abstract tasks.
4. $EVENT$ is the Event Sequence Chart between the organizations which express the interaction protocols (in Definition4).

Figure3.1 shows the local modeling view of organization B of an interorganizational workflow example. Unlike traditional loosely coupled interorganizational workflow process definition, the local organization B only has abstract information (as abstract workflow) of organization A and C with B 's own view rather than concrete information provided by A and C .

In the local modeling view of the interorganizational workflow, the concrete workflow represents the workflow within the local organization and the abstract workflow is built with view of the local organization, which represents the workflows in organizations that have interaction with it.

3.2 Concrete Workflow, Abstract Workflow

Activity-based workflow models generally use tasks and dependencies to describe a process. Dependencies prescribe the ordering relationships between tasks within a process. According to WfMC, the following six ordering structures may appear in business processes.

- Sequence : A task has a single subsequent task.
- AND-split : A task splits into multiple parallel tasks that are all executed.

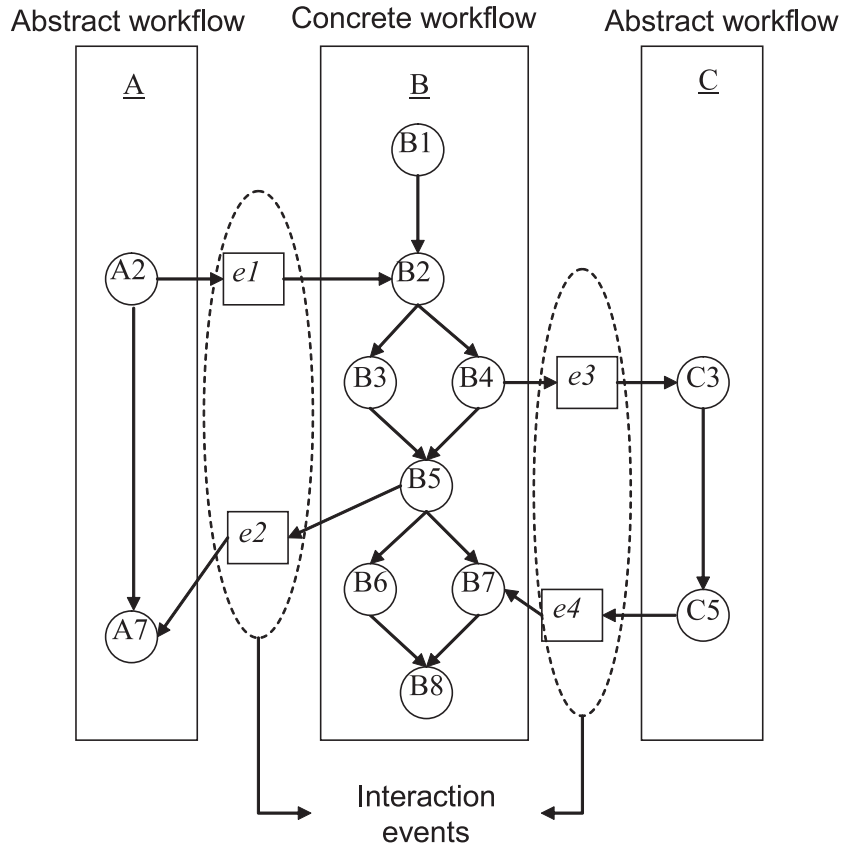


Figure 3.1: An example of the local modeling view

- OR-split : A task splits into multiple mutually exclusive alternative tasks, only one of which is followed.
- AND-join : Multiple parallel executing tasks join into a single activity.
- OR-join : Multiple mutually exclusive alternative tasks join into a single task.
- Loop : One or more tasks are repeatedly executed until the exit condition is satisfied.

To simplify the problem, the loop routing of tasks are not considered.

The definition of a concrete workflow is as following.

Definition2(Concrete Workflow)

A concrete workflow is a following tuple.

$$WFL = (i, TS, Intra\ flow, Inter\ flow, Connection)$$

1. i is the information of the organization that the workflow belongs to;
2. TS is the set of tasks in the workflow. Each task has the following elements: description, precondition, effect and subprocessID. A precondition is the condition which must be satisfied to execute the task. An effect is the effect which may be made by execution the task. A subprocessID represents the subprocess which the task belongs to. A task can be expressed as following.

$$T = (description, precondition, effect, SID)$$

3. *Intraflow* is the set of dependencies within the same organization. Each intra dependency includes a pair of tasks and the dependency relationship between two tasks, which has the form of following.

$$IntraDependency = (T_{from}, T_{to}, IntraRelationship) \quad T_{from}, T_{to} \in TS$$

$$IntraRelationship \in \{AND-split, AND-join, OR-split, OR-join, Sequential\}$$

4. *Interflow* is the set of dependencies between the tasks and interaction events across organizations. Each inter dependency includes a task, an event and the dependency relationship between the task and the event, which has the form as following.

$$InterDependency = (T, e, InterRelationship)$$

$$T \in TS \quad InterRelationship \in \{from_T, to_T\}$$

5. *Connection* is the set of causal connections in the workflow. Each causal connection has the form of $T_i \rightarrow T_j$ which means that T_i gets the condition P for T_j where $T_i, T_j \in TS$.

In the design of the concrete workflow, each organization can set subprocesses which consists of some tasks. Each subprocess has the same parameter of a task: description, precondition and effect. Each organization sets these parameters to a subprocess. The definition of the subprocess is as following.

Definition3(Subprocess)

A subprocess SP within the concrete workflow WFL is a following tuple .

$$SP = (id, PTS, p, e)$$

1. id is a unique identifier of subprocess.
2. PTS is the set of tasks which is contained in a subprocess. Each task must satisfy following condition: $PTS = \{t_1, t_2, \dots, t_n\} \subset TS$. And if there is a task $t \in PTS$ and intra relationship $(t, t_j, IntraRelationship) \in Intraflow$ ($t_j \in \{TS - PTS\}$), there must not be the other task t_i in the subprocess SP such as $(t_j, t_i, IntraRelationship)$.
3. p is the precondition of the subprocess.
4. e is the effect of the subprocess.

In contrast to traditional interorganizational workflow, the workflow which other organizations execute is abstract. The organizations have full information about the workflow which they execute, therefore they can design the concrete workflow completely. In other words, the tasks in the concrete workflow are primitive and concrete tasks. However, the workflow which other organizations execute cannot be expressed completely, because each organization has only partial knowledge about it.

The definition of the abstract workflow is as following.

Definition4(Abstract Workflow)

An abstract workflow is a following tuple.

$$WFA = (i, VTS, VIntraflow, VInterflow, VConnection)$$

1. i is the information of the organization that the workflow belongs to.
2. VTS is the set of abstract tasks. An abstract task has three parameters: *description, precondition, effect*.
3. $VIntraflow$ is the set of dependencies within the same workflow. Each dependency has a pair of abstract tasks, and the dependency relationship between the two abstract tasks. Each dependency has the same parameters within those of the concrete workflow.
4. $VInterflow$ is the set of dependencies between the abstract tasks and interaction events.
5. $VConnection$ is the set of causal connections in the workflow. The connection occurs between the abstract tasks.

In contrast to the concrete workflow, the tasks in the abstract workflow are the abstract tasks. Each abstract task can be regarded as the abstraction of a set of concrete tasks in the concrete workflow. The example of primitive tasks and abstract tasks is shown as Figure 3.2. Each task set expresses the development process. In case this process is executed in the local organization, the process can be expressed in detail by the concrete tasks such as “requirement analysis”, “specification design”, “coding” and “testing”. On the other hand, in case this process is executed by other organizations, this process may be expressed by the abstract tasks such as “design” and “implementation”. Obviously, these processes are the same process, but consist of different kind of tasks.

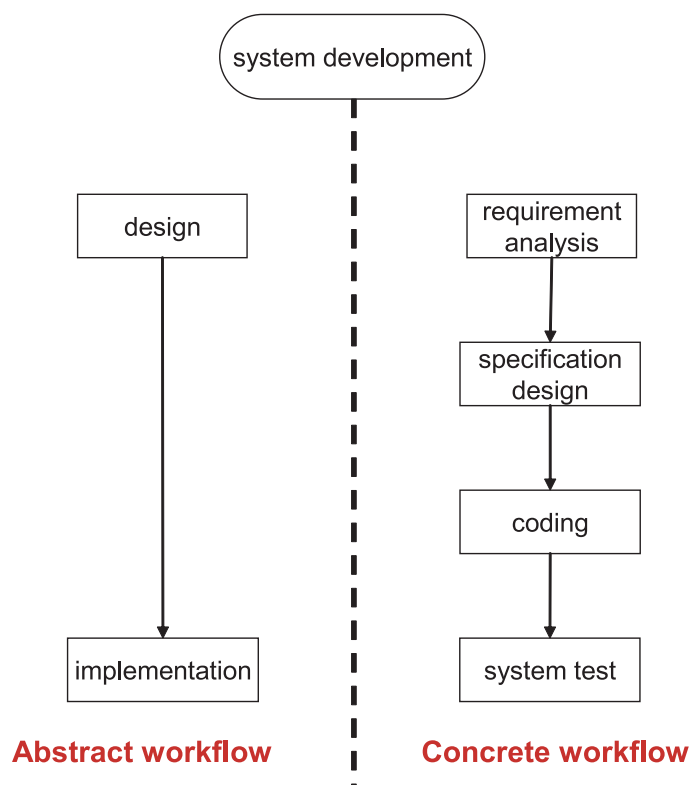


Figure 3.2: the abstract tasks and the concrete tasks

3.3 Interaction Protocol

In previous research[20][17], the interaction events in the loosely coupled inter-organizational workflow are specified by using *Message Sequence Charts (MSC)*.

Message Sequence Charts are a widespread graphical language for the visualization of communications between systems/processes[22]. The representation of message sequence charts is intuitive and focuses on the messages between communication entities.

In this work, we define the interaction protocols among the organizations by using the similar method as message sequence charts. The definition of the interaction protocols is as following.

Definition5(Event Sequence Chart)

An event sequence chart is defined as a following tuple.

$$EVENT = (I, E, from, to, \{\leq_i\})$$

1. I is a finite set of organizations;
2. E is a finite set of events;
3. to and $from$ are functions from E to I . For example , $to(e) = i (i \in I)$ means that the organization I receives the event E . Similarly, $from(e) = i (i \in I)$ means that the organization I sends the event E
4. For each $i \in I : \leq_i$ is a partial order on

$$\{?e \mid e \in E \text{ and } to(e) = i\} \cup \{!e \mid e \in E \text{ and } from(e) = i\}$$

where $?e$ represents sending event and $!e$ represents receiving event.

The example of event sequence charts is shown as Figure3.3. This event sequence chart has following events.

- send 1st specification document
- send report
- send review
- send 2nd specification document
- send final report

Each event has the pair of organizations: sender and receiver. For example, “send 1st specification” event is received by the Chinese company, and sent by the Japanese company. The orderings of events can be partial. In this event

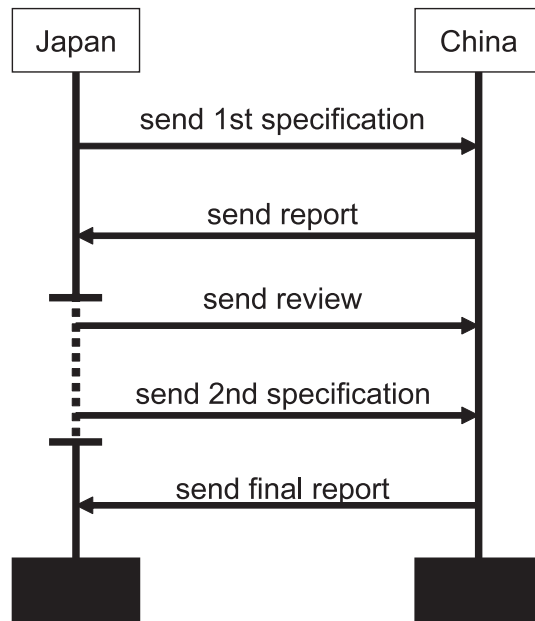


Figure 3.3: Event sequence charts

sequence chart, “send review” event and “send 2nd specification document” can be executed in any order. In traditional loosely coupled interorganizational workflow, there is one message sequence chart for the global workflow process. In our research, each local modeling view has an event sequence chart.

Chapter 4 Compatibility Analysis

In this chapter, the method of compatibility analysis with comparison of multiple local modeling views is presented. In our research, multiple local modeling views are compared for the purpose of detecting incompatibilities. First, what kind of the incompatibility we try to detect is shown. Next, the method of compatibility analysis is presented.

4.1 Incompatibility of Multiple Local Modeling Views

In Chapter 3, we define local modeling view and a series of related elements. In the interorganizational workflow, each organization has its own local modeling view, including elements of local concrete workflow, interaction events and abstract workflows of other organizations that interact with it. Information of the concrete workflow is completely included in the local modeling view. However, the part of the abstract workflow is defined in an incomplete way based on limited sharable knowledge by other organizations. Therefore, there might be conflicts between the abstract workflow and the related concrete workflow. Further more, when defining interaction events, there might also be conflicts between different local modeling views. In this research, we focus on the static workflow modeling, and detect following incompatibility on interaction events.

Incompatibility1 : Disagreements on the interaction events

Event sequence charts which is included in different local modeling views are consistent with each other. In other words, each event sequence chart has the same set of events and event orderings. When one organization tries to send an event to the other organization, the same event must be received the other organization in the local modeling view of the other organization has. Otherwise, the interaction among organizations cannot be executed properly. In such a case, multiple local modeling views are regarded as incompatible.

The abstract workflow and the concrete workflow are included in the local modeling view. Therefore, there are the abstract workflow and the concrete workflow which one organization executes after all the organizations design the

local modeling views. Then we try to detect incompatibilities between the abstract workflow and the related concrete workflow.

As Chapter 3 described, the abstract workflow is abstract rather than the concrete workflow, and the abstract tasks can be regarded as the set of the concrete tasks. Therefore, the abstract workflow can be regarded as the abstraction of the concrete workflow.

Liu[7] proposed the method of generating abstraction of a workflow process called process-view. Process-view is an abstraction of the concrete workflow, which consists of virtual tasks. In the abstraction procedure, some tasks are replaced by single virtual task. Liu suggested following conditions which process-view must satisfy.

- A virtual task's member may be a concrete task or a previously defined virtual task.
- The implied ordering relations between two virtual tasks' respective members must conform to the ordering relation in the concrete workflow process

Similarly, in our research, these conditions can be applied to compatibility analysis of the local modeling views. In this paper, we try to detect following incompatibilities by comparing of the abstract workflow and the related concrete workflow.

Incompatibility2 : Membership incompatibility

The abstract tasks must be an abstraction of the set of the concrete tasks in the concrete workflow. Therefore, each abstract task has the set of the concrete tasks which has the same precondition and effect in the concrete workflow. This incompatibility means that the abstract task cannot be found as any set of the concrete tasks.

Incompatibility3 : Ordering preservation incompatibility

The task order of the abstract tasks must be preserved in the task order of the related set of the concrete tasks. If the abstract task can be derived from the set of concrete tasks and the other abstract task can be derived from the set of the other concrete tasks, the task order of the abstract tasks preserved in the sets of the concrete tasks.

If the above incompatibilities cannot be detected in compatibility analysis, then multiple local modeling views are considered compatible. The definition of compatibility is as following. To simplify a problem, we consider the collaborative tasks by two organizations.

Definition6(Compatibility)

Let $LC-WF_A$ and $LC-WF_B$

$$LC - WF_A = (I_A, WFL_A, WFA_B, \dots, WFA_M, EVENT_A)$$

$$LC - WF_B = (I_B, WFL_B, WFA_A, \dots, WFA_N, EVENT_B)$$

be the local modeling views of two organizations A and B , they are called **compatible**, iff:

1. Let $EVENT_A$ and $EVENT_B$ be the event sequence charts.

$$EVENT_A = (I_A, E_A, from_A, to_A, \{\leq A\}_B)$$

$$EVENT_B = (I_B, E_B, from_B, to_B, \{\leq B\}_A)$$

$$E_A = E_B \text{ and } \{\leq B\}_A == \{\leq A\}_B$$

2. for each $vt_i \in VTS$, there exists $t_{i1}, t_{i2}, \dots, t_{iN} \in TS$ in the related concrete workflow, such that vt_i can be derived from $\{t_{i1}, t_{i2}, \dots, t_{iN}\}$,

$$PRECOND(vt_i) = PRECOND(\{t_{i1}, t_{i2}, \dots, t_{iN}\})$$

$$EFFECT(vt_i) = EFFECT(\{t_{i1}, t_{i2}, \dots, t_{iN}\})$$

3. for each $vt_i, vt_j \in VTS$ with the dependency $(vt_i, vt_j, IntraRelationship)$, there does not exist an order $(t_j, t_i, IntraRelationship)$ in the related concrete workflow where $t_i \in \{t_{i1}, t_{i2}, \dots, t_{iN}\}$ that vt_i can be derived and $t_j \in \{t_{j1}, t_{j2}, \dots, t_{jM}\}$ that vt_j can be derived.

4.2 Algorithms of Compatibility Analysis

As shown in Chapter 2, the constraint that each organization need to conceal the part of its local workflow must be considered. Each organization cannot

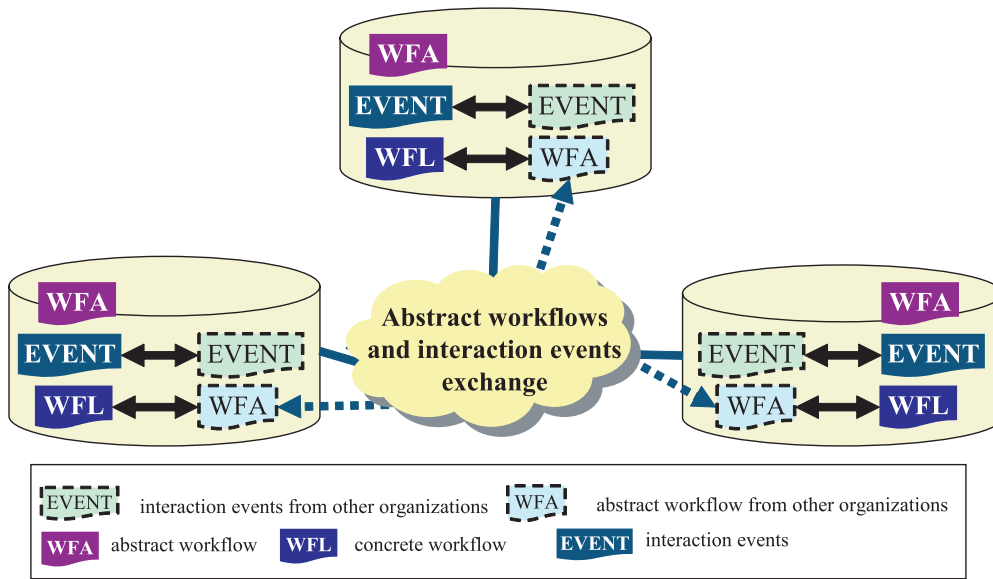


Figure 4.1: An approach of compatibility analysis of the local modeling views

disclose its concrete workflow to other organizations. In order to detect incompatibilities, we propose the approach that the organizations exchange their abstract workflow and event sequence charts in the local modeling view among the organizations. In this approach, each organization receives the abstract workflows and event sequence charts from other organizations. Then, each organization compare these elements with its local modeling view in order to detect incompatibilities.

Our approach is shown as Figure4.1 . The compatibility analysis process is as following.

1. Each organization designs the local modeling view which includes the concrete workflow, the abstract workflow and event sequence chart.
2. The organizations exchange the abstract workflow and the event sequence chart among them.
3. Event comparison is performed.
4. Abstract-concrete workflow comparison is performed.

The compatibility analysis process based on the local modeling view can mainly be divided into two comparison procedures: event comparison and abstract-concrete workflow comparison. The comparison of the local modeling view is shown as Figure4.2.

4.2.1 Event Comparison

The interaction protocols(EVENT) should first be analyzed and compared, which includes two parts, the contents of the events and the order of the events between two organizations. In Chapter 3, we have defined the event sequence chart of the local modeling view as

$$EVENT = (I, E, from, to, \{\leq i\})$$

Suppose there are two organizations, namely A and B . Each local modeling view has an event sequence chart. The algorithm of event comparison is shown in Algorithm1.

4.2.2 Block-structure Transformation of Workflow

If the event comparison results in compatible, the abstract workflow and its related concrete workflow should be further compared. However, it is difficult to compare the incomplete abstract workflow and the complicatedly structured

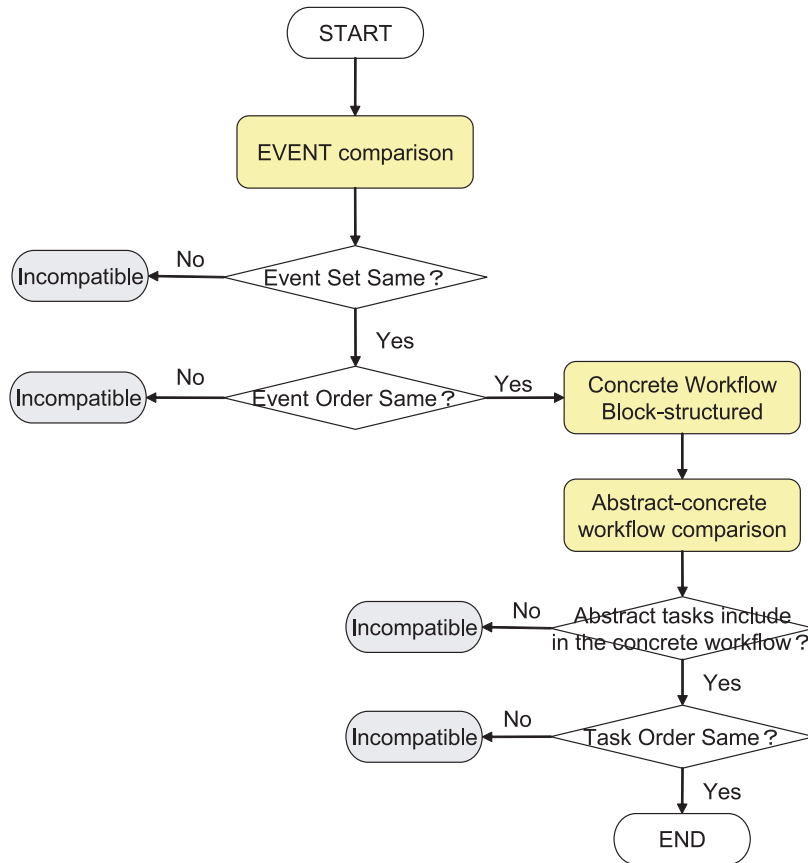


Figure 4.2: Compatibility analysis flowchart

concrete workflow, because:

- The structure of the local concrete workflow is always very complicated;
- Some abstract parts in the abstract workflow cannot be directly mapped into the concrete workflow. For example, an abstract task might be derived from a set of concrete tasks in the concrete workflow.

Therefore, we need establish a mechanism to compare the abstract workflow and concrete workflow. Lin[6] proposed the method of transformation workflow process into block-structured graph. Similarly to Lin's method, we transform the concrete workflow process into the block-structured workflow process and then make the comparison of abstract workflow and the related concrete workflow. In this part, we concentrate on the block-structure transformation algorithms.

In this block-structure transformation algorithm, multiple tasks in the concrete workflow are replaced by single virtual task called *block*. A block is a unit

Algorithm 1 EVENT comparison

```

 $E_{withA}, E_{withB} \Leftarrow \phi$ 
for all  $e_i \in E_A$  do
  if  $to(e_i) == B$  or  $from(e_i) == B$  then
     $E_{withB} \Leftarrow E_{withB} \cup \{e_i\}$ 
  end if
  for all  $e_j \in E_B$  do
    if  $to(e_j) == A$  or  $from(e_j) == A$  then
       $E_{withA} \Leftarrow E_{withA} \cup \{e_j\}$ 
    end if
  end for
  if  $E_{withB} == E_{withA}$  and  $\{\leq_B\}_A == \{\leq_A\}_B$  then
    return Compatible
  else
    return Incompatible
  end if
end for

```

of representation that can minimally specify the behavioral pattern of process flow. The behavior patterns in process models are classified into *sequential*, *parallel*, *non-conditional selective* and *conditional selective*.

The definition of a block is as following.

Definition7(Block) A block is a tuple

$$Block = (TS, D, p, e, O, Type)$$

1. TS is the set of tasks in the block;
2. D is the description of the block;
3. p is the precondition of the block as a whole;
4. e is the effect of the block as a whole;
5. O is the ordering constraints of tasks in the block;
6. $Type$ is the type of the block and $Type \in \{Sequential, Parallel, Non - conditional selective, Conditional selective\}$.

Figure4.3 shows the four block types discussed in this paper. The previous research work of Wil van der Aalst et al. has resulted in the identification of 20 workflow patterns that describe the behavior of business processes[17][16].

In this research, we focus on the static workflow modeling; therefore we do not concentrate on the complicated workflow patterns involved in the dynamic workflow execution period. Four types of routing constructs are following.

- **Sequential**

Tasks are executed sequentially if the execution of one task is followed by the next task. In Figure4.3(a), task B is executed after task A has been completed and before task C is started.

- **Parallel**

In Figure4.3(b) , task B and task C are executed in parallel. This means that B and C are executed at the same time or in any order. To model parallel routing, two building blocks are identified: (1)the *AND-split* and (2)the *AND-join*. The AND-split in Figure4.3(b) enables B and C to be executed after A has been completed. The AND-join synchronizes the two parallel flows, i.e., task D may start after B and C have been completed.

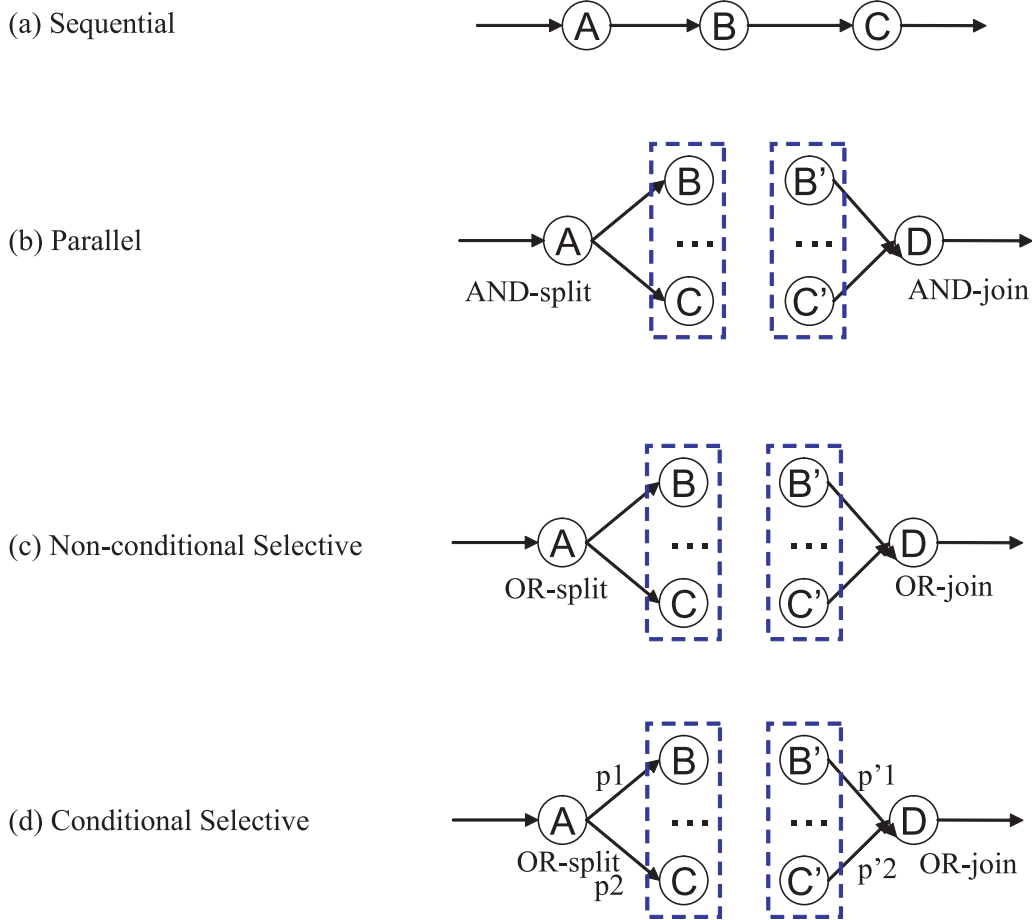


Figure 4.3: Task routing constructs

- **Non-conditional selective**

In Figure 4.3(c), either task B or task C is executed. To model a choice between two or more alternatives we use two building blocks: (1) the *OR-split* and (2) the *OR-join*. If task A is executed, a choice is made between B and C . Task D may start after B or C is completed.

- **Conditional selective**

In Figure 4.3(d), a condition p is set to the flow between two tasks. If task A is completed, then the condition is evaluated. And the task connected to the flow with the conditions judged to be true is executed. Similarly to Non-conditional selective, we use two building blocks: *OR-split* and *OR-join*. In Figure 4.3(d), condition p_1 and p_2 are evaluated after task A has been completed. In a similar way, condition p'_1 and p'_2 are evaluated before

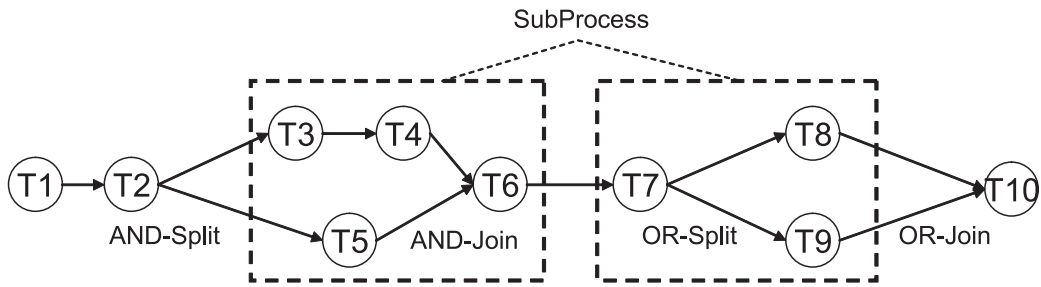


Figure 4.4: Concrete Workflow with Semantic Subprocess

task D starts.

A block is generated by replacing the task set belonging to a common task routing construct. Therefore, blocks can be generated by analyzing the process structure. A block has little semantic meaning, however, a subprocess has. A subprocess is a collection of some task nodes defined by the organization and can represent an abstract task. A precondition and effect of subprocess are set by the organizations similarly to the tasks.

Figure 4.4 is an example of concrete workflow process definition network with semantic subprocesses. Our goal in this section is to find out all the blocks and subprocesses in this network and turn the network into the block-structured workflow process definition. In Figure 4.4, there are two subprocesses in the concrete workflow.

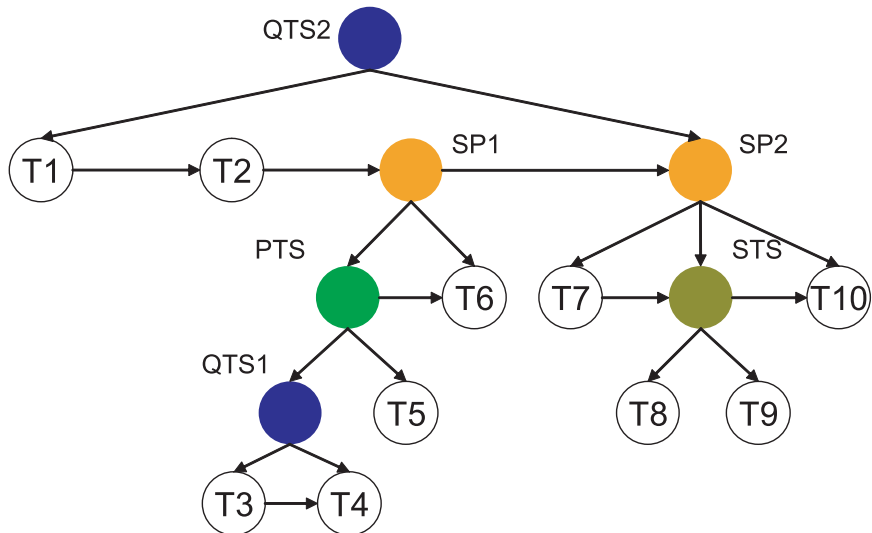


Figure 4.5: Block-structured Workflow Process

We reach such a goal by the following steps.

1. We find out a sequential block $QB_1\{T_3, T_5, T_9\}$, a parallel block $PB_1\{T_{13}, T_{14}\}$ and a non-conditional selective block $SB_1\{T_6, T_7, T_8\}$. Then, we replace $\{T_3, T_5, T_9\}$ with QB_1 , $\{T_{13}, T_{14}\}$ with PB_1 and $\{T_6, T_7, T_8\}$ with SB_1 . After the replacements, the graph network becomes smaller.
2. We find out a parallel block $PB_2\{QB_1, QB_2\}$ and then replace $\{QB_1, QB_2\}$ with PB_2 .
3. We find out a subprocess $SP_1\{T_2, PB_2, T_{11}\}$ and a subprocess $SP_2\{T_{12}, PB_1, T_{15}, T_{16}\}$. Then, we replace $\{T_2, PB_2, T_{11}\}$ with SP_1 and $\{T_{12}, PB_1, T_{15}, T_{16}\}$ with SP_2 .
4. We find out a sequential block $QB_3\{T_1, SP_1, SP_2, T_{17}\}$ and replace $\{T_1, SP_1, SP_2, T_{17}\}$ with QB_3 .
5. We find out that the whole concrete workflow graph network becomes into a single sequential block node QB_3 .

The process of above replacements can be expressed by the block-structured workflow process tree as shown in Figure4.5. The process tree has two merits: simple structure and semantic meaning.

4.2.3 Algorithm of Block-structure Transforming of Workflow

Figure4.6 shows the whole flowchart of the algorithm of transforming the process definition network to the block-structured workflow process tree. There are mainly four algorithms:

- task node value computation
- sequential block detection
- branch block detection
- the subprocess detection

The algorithm of computing the value of task nodes is shown in Algorithm2. Each task node has a value to show its position in the whole definition network. The start task node has the value 1, which means it is the most outer node in the network. If there is a split (AND-SPLIT or OR-SPLIT) node, then the value of the successor node would be larger than the split node. If there is a join (AND-JOIN or OR-JOIN), then the value of the successor node would be less than the join node. As a result, the most inner node has the largest value.

All the task nodes that belong to a same block have the same values. Therefore we use this point to detect the blocks in the definition network.

After computing values of task nodes, sequential block detection would be executed, which is shown in Algorithm3. Since the nodes that have the largest value are the most inner nodes, the detection starts from such nodes. The detection process is executed by checking the in-degree and out-degree of the nodes. Once a sequential block is detected, the related nodes would be replaced by the sequential block and the whole definition network would become smaller.

The whole detection process starts from the most inner part of the process definition network, if there is no sequential block, there must be some parallel blocks or selective blocks in the most inner part. The task nodes that belong to a same parallel block or selective block have the same predecessor split node

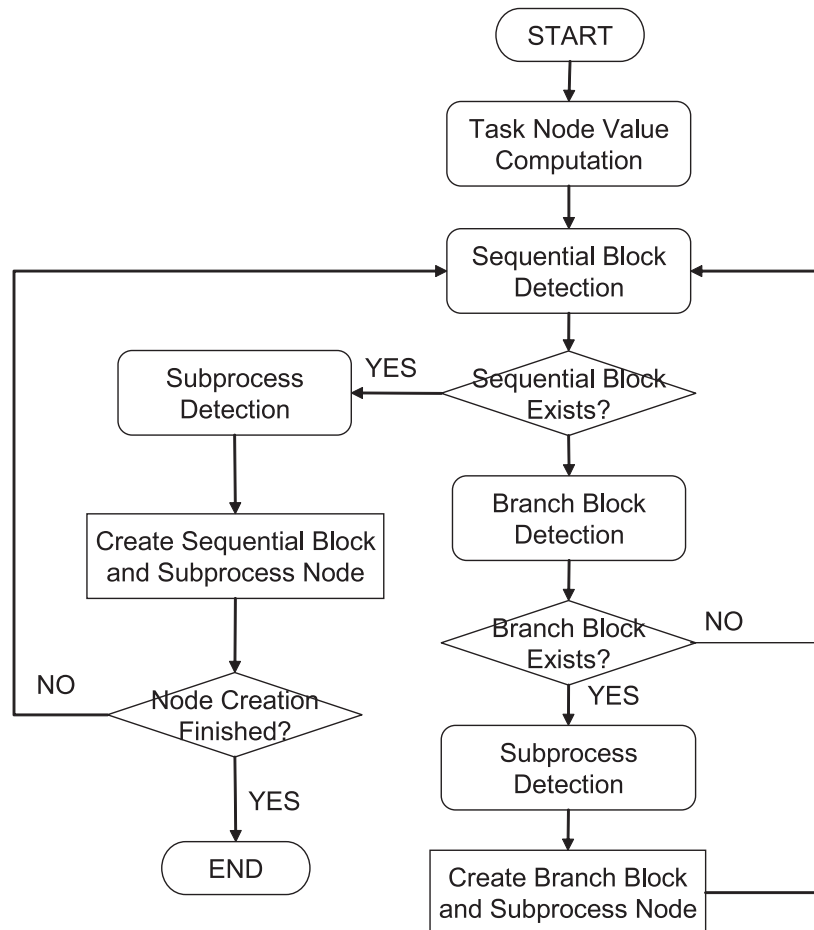


Figure 4.6: Block structure transformation flowchart

Algorithm 2 Compute the value of task nodes

INPUT: task node definition network $G = (V, E)$
OUTPUT: the max value of all the task nodes v_{max}
 $color[s] \Leftarrow$ GREY; $value[s] \Leftarrow 1$;
 $v_{max} \Leftarrow 1$; $Q \Leftarrow \phi$;
for all $u \in V - \{s\}$ **do**
 $value[u] \Leftarrow 0$; $color[u] \Leftarrow$ WHITE;
end for
ENQUEUE(Q, s);
while $Q \neq \phi$ **do**
 $u \Leftarrow$ DEQUEUE(Q);
 for all $v \in succ[u]$ **do**
 if $color[v] =$ WHITE **then**
 switch(*TypeOfEdge*)
 case **SEQ**: $value[v] \Leftarrow value[u]$
 case **SPLIT**: $value[v] \Leftarrow value[u] + 1$
 case **JOIN**: $value[v] \Leftarrow value[u] - 1$
 end switch
 if $value[v] > v_{max}$ **then**
 $v_{max} \Leftarrow value[v]$
 end if
 ENQUEUE(Q, v); $color[v] \Leftarrow$ GREY
 end if
 end for
end while

and successor join node. Once a parallel block or selective block is detected, the related nodes would be replaced by the block and the whole definition network would become smaller. Further, some new sequential blocks might appear. Therefore, after the creation of the parallel block or selective block node, the whole process would come back to the sequential block detection. The parallel and selective block detection is shown in Algorithm 4.

Algorithm 3 Detect the sequential blocks

INPUT: $G = (V, E)$ OUTPUT: v_{max} and sequential block seq_block $flag \leftarrow 1$; $seq_block \leftarrow \phi$; $Q \leftarrow \phi$; ENQUEUE(Q, s)**while** $flag = 1$ **do** **if** $Q = \phi$ **then** **return** *null* **end if** $u \leftarrow$ DEQUEUE(Q); **if** ($value[u] = v_{max}$) \wedge ($out[u] = 1$) \wedge ($in[succ[u]] = 1$) **then** $flag = 0$; $seq_block \leftarrow seq_block \cup \{u\}$; **while** ($out[u] = 1$) \wedge ($in[succ[u]] = 1$) **do** $seq_block \leftarrow seq_block \cup \{succ[u]\}$; $u \leftarrow succ[u]$; **end while** update *relatednodes* with *sequentialblock*; $value[seq_block] \leftarrow value[u]$; **else** ENQUEUE($Q, succ[u]$) **end if****end while**

The algorithm of subprocess detection is shown in Algorithm5, which is executed after each block is detected. Within a block, if all the task nodes have a same subprocess ID, then the subprocess ID is assigned to the whole block. If a part of task nodes are detected to own a same subprocess ID, then the part of nodes are detected as a subprocess.

4.2.4 Comparison of Abstract Workflow and Block-structured Workflow

In the abstract-concrete workflow comparison procedure, the concrete workflow has been turned into the block structure and has the form of block tree. Then the problem of abstract-concrete workflow comparison is simplified into how to map abstract tasks in the abstract workflow into the block-structured concrete

Algorithm 4 Detect parallel and selective blocks

INPUT: $G = (V, E)$ OUTPUT: v_{max} and parallel or selective block bra_block $flag \Leftarrow 1$; $bra_block \Leftarrow \phi$; $Q \Leftarrow \phi$;ENQUEUE(Q, s)**while** $flag = 1$ **do** $u \Leftarrow$ DEQUEUE(Q); **if** ($value[u] = v_{max}$) **then** $flag = 0$; **for all** $v \in succ[pred[u]]$ **do** $bra_block \Leftarrow bra_block \cup \{v\}$; **end for** update *relatednodes* with *parallelorselectiveblock*; $value[bra_block] \Leftarrow$ $value[u] - 1$; update v_{max} ; **else** ENQUEUE($Q, succ[u]$) **end if****end while**

workflow. The comparison includes the following two aspects.

1. Search and map the abstract tasks of abstract workflow process in the block-structured concrete workflow process tree;
2. Compare the task order of abstract workflow and the related task order of the concrete workflow process tree.

In the comparison of the abstract workflow and the related block-structured concrete workflow process tree, the preconditions and effects of tasks are compared. All the abstract tasks should be found in the block-structured concrete workflow process tree. The pair of tasks which have the same preconditions and effects are regarded as the same tasks. If there is the abstract task which cannot be mapped in the block-structured workflow process tree, this results in the incompatibility in the two workflows.

Algorithm 5 Detect subprocesses

INPUT: Detected block in Algorithm3 or 4

OUTPUT: Subprocess *subprocess* and subprocess ID of the whole block*block_subid**block_subid* $\leftarrow \phi$; *detected_subid* $\leftarrow \phi$;**for all** *u* \in *block* **do** **if** *subid*[*u*] \notin *block_subid* **then** *block_subid* \leftarrow *block_subid* \cup *subid*[*u*] **end if****end for****for all** *subid* \in *block_subid* **do** *flag* \leftarrow 0; **for all** *u* \in *block* **do** **if** *subid* \notin *subid*[*u*] **then** *flag* \leftarrow 1; *subprocess* \leftarrow *block* - {*u*}; **end if** **end for** **if** *flag* = 1 **then** *detected_subid* \leftarrow *detected_subid* \cup {*subid*}; **end if****end for***block_subid* \leftarrow *block_subid* - *detected_subid*

The algorithm of mapping tasks between the abstract workflow and block-structured workflow process tree is shown as Algorithm6. To search tasks, preconditions and effects of tasks are compared. When an abstract task *vt* can be mapped in a node *bt* in the block-structure workflow process, the result of mapping is kept as $map[vt] = bt$. This procedure is applied to all the abstract tasks.

After such task mapping procedure, the task orderings in the abstract workflow and block-structured workflow are compared. The algorithm of comparison of the task orderings between the abstract workflow and the block-structured

Algorithm 6 Search and map abstract tasks in block-structured workflow

INPUT: abstract workflow and block-structured workflow of the concrete workflow

OUTPUT: mapping result of all the abstract tasks

(* VTS is the set of the abstract task*)

(* BTS is the set of nodes in the block-structured workflow process such as tasks, blocks and subprocesses*)

for all $vt \in VTS$ **do**

$map[vt] \leftarrow \phi$

end for

for all $vt \in VTS$ **do**

for all $bt \in BTS$ **do**

if $precond(vt) == precond(bt) \wedge effect(vt) == effect(bt)$ **then**

$map[vt] \leftarrow bt$; break;

end if

end for

end for

workflow is shown as Algorithm7. If there is an order constraint between a pair of abstract tasks vt_i, vt_j , this order constraint must be satisfied between a pair of nodes bt_i, bt_j , which vt_i can be mapped in bt_i and vt_j can be mapped in bt_j . If there is no order constraint between abstract tasks, then there must be no order constraint between related nodes in block-structured workflow process.

The whole algorithm of compatibility analysis to support multiple local modeling views is shown as Algorithm8. First, the interaction events are compared. In comparison of interaction events, interaction events and event orderings are compared. If there is no incompatibility among interaction events, then the concrete workflow and abstract workflow are compared. In abstract-concrete workflow comparison, the concrete workflow is transformed into the block-structure workflow process tree. Next, the abstract tasks are searched and mapped in the block-structure workflow process. If all the abstract tasks can be mapped in the block-structure workflow, then the task orderings in the abstract workflow

Algorithm 7 Task order comparison of the abstract workflow and block-structured workflow

INPUT: abstract workflow and block-structured workflow of the concrete workflow

OUTPUT compatible or incompatible

```
for all  $(vt_i, vt_j) \in VTS$  do  
  if  $(vt_i \in succ[vt_j]) \wedge (map[vt_i] \in succ[map[vt_j]])$  then  
    continue;  
  else if  $(vt_i \in pred[vt_j]) \wedge (map[vt_i] \in pred[map[vt_j]])$  then  
    continue;  
  else if  $(vt_i \notin succ[vt_j] \cup pred[vt_j]) \wedge (map[vt_i] \notin succ[map[vt_j]] \cup pred[map[vt_j]])$  then  
    continue;  
  else  
    return Incompatible  
  end if  
end for
```

and block-structure workflow are compared.

After all the procedures are applied to multiple local modeling views, if no incompatibility cannot be detected, multiple local modeling views are considered compatible.

Algorithm 8 Compatibility analysis

INPUT: local modeling views, LC-IOWF_A and LC-IOWF_B

OUTPUT: compatible or incompatible

(* Event_Comparison is the procedure shown as Algorithm 1 *)

if Event_Comparison(EVENT_A, EVENT_B) == *incompatible* **then**

return Incompatible

end if

(* Block-Structure_Transformation is the procedure shown as Algorithm 2,3,4,5 *)

(* Task_Mapping is the procedure shown as Algorithm 6 *)

Task_Mapping(WFA_A, Block-structure_Transformation(WFL_B))

Task_Mapping(WFA_B, Block-structure_Transformation(WFL_A))

for all $vt_a \in VTS_A$, $vt_b \in VTS_B$ **do**

if $map[vt_a] == \phi \vee map[vt_b] == \phi$ **then**

return Incompatible

end if

end for

(* Task-Order-Comparison is the procedure shown as Algorithm 7 *)

if Task-Order-Comparison(WFA_A, Block-Structure_Transformation(WFL_B)) == *incompatible* **then**

return Incompatible

end if

if Task-Order-Comparison(WFA_B, BlockStructure_Transformation(WFL_A)) == *incompatible* **then**

return Incompatible

end if

return Compatible

Chapter 5 Implementation

To realize our method of compatibility analysis, we implemented the system. In this chapter, the implementation system is presented. Furthermore, to explain our method, an example of offshore software development between Japanese company and Chinese company is shown.

5.1 Implementation System

5.1.1 System Module

The system architecture is shown as Figure 5.1. In this system, user can design the abstract workflow and the concrete workflow and analyze compatibility between the own concrete workflow and the abstract workflow which other users design. The workflow process can be designed visually, and saved as XML document. The abstract workflow is shared among organizations, and the concrete workflow is transformed into block-structured workflow process tree which is compared with the abstract workflow in order to detect incompatibilities. Incompatibilities are visualized and informed of users.

Implemented system modules are following.

1. **Workflow process design module**

This is the function to design the workflow process visually. In the design of workflow process, users allocate some tasks expressed as rectangle and connect them with edges. Moreover, task parameters such as precondition and effect can be set to the tasks. Dependency parameters such as routing constructs and condition can be set to the control flow, too. And some subprocesses can be constructed by choosing the set of the tasks.

2. **Concrete workflow transformation module**

In this function, the concrete workflow which is designed using above function can be transformed into block-structured workflow process tree. The virtual tasks such as blocks and subprocesses have the precondition and effect, which is set by the user. The block-structured workflow process tree is visualized as process tree.

3. **Compatibility analysis module**

In this function, the abstract workflow is compared with the block-structured workflow process tree, and detected incompatibilities. The tasks of the abstract workflow are mapped in the block-structured workflow process, and the task order of the abstract workflow is compared with that of the block-structured workflow process. The result of these procedures is visualized, and users can get the information about incompatibilities.

5.1.2 System Interface

System interface consists of following three parts.

1. Workflow process design area

In this area, one can design the workflow process visually such as the concrete workflow and the abstract workflow.

2. Abstract workflow display area

The abstract workflow which other organizations design is loaded in this area.

3. Block-structured workflow display area

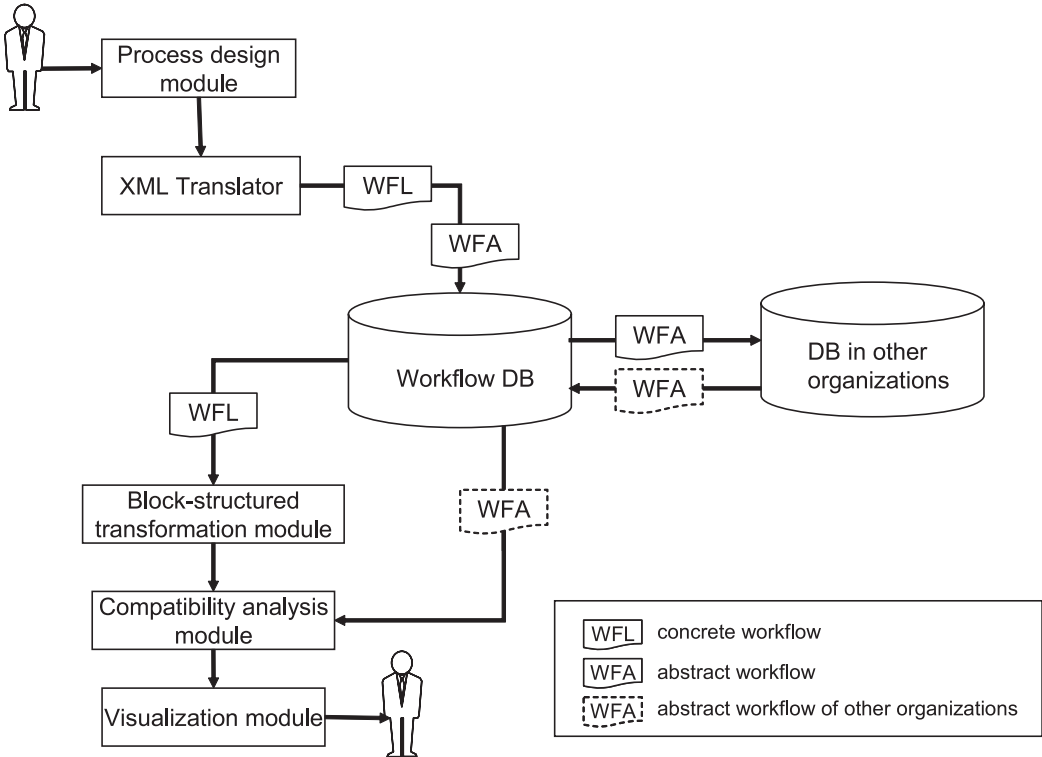


Figure 5.1: System Architecture

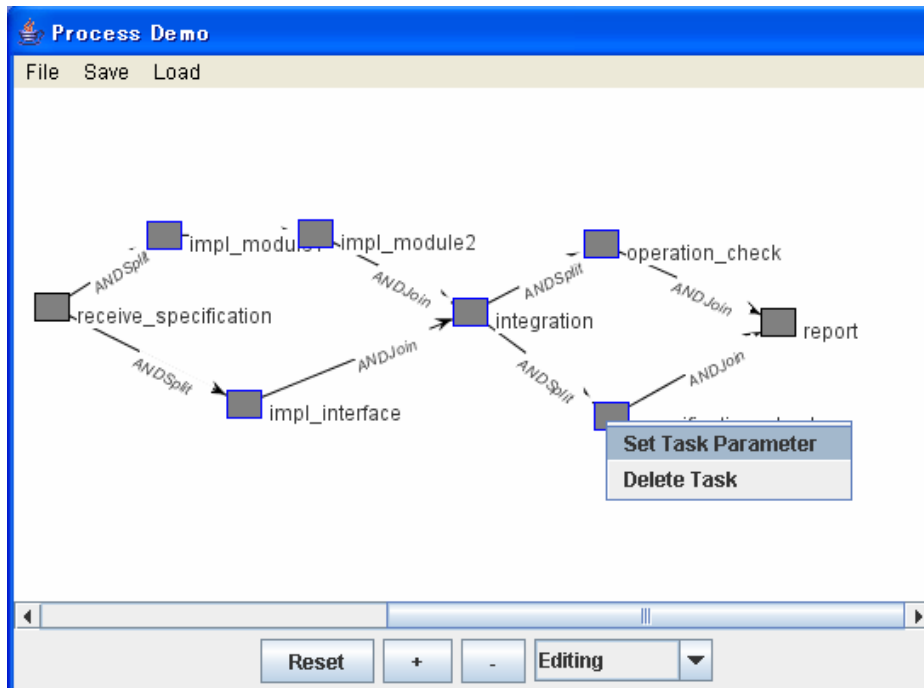


Figure 5.2: The design of workflow process

The block-structured workflow process of the concrete workflow is displayed in this area.

Each interface of the function indicated in 5.1.1 is as following.

- **Design of workflow process**

The concrete workflow and the abstract workflow can be designed shown as Figure5.2. The parameters of precondition and effect can be set for workflow process. Parameter setting window is shown as the right side of Figure5.2. In this window, one can set the description, precondition and effect for the tasks. Similarly, the edges between the tasks can be set the parameters such as task routing construct and condition.

- **Block-structure Transformation of concrete workflow**

After the design of the concrete workflow, the block-structure workflow process tree is generated shown as Figure5.3 . Block-structure workflow process is expressed as tree. Subprocesses and blocks in the block-structure workflow process are marked by the specific colors in order to distinguish between these elements and concrete tasks. The parameters of precondition and effect can be set for subprocesses and blocks.

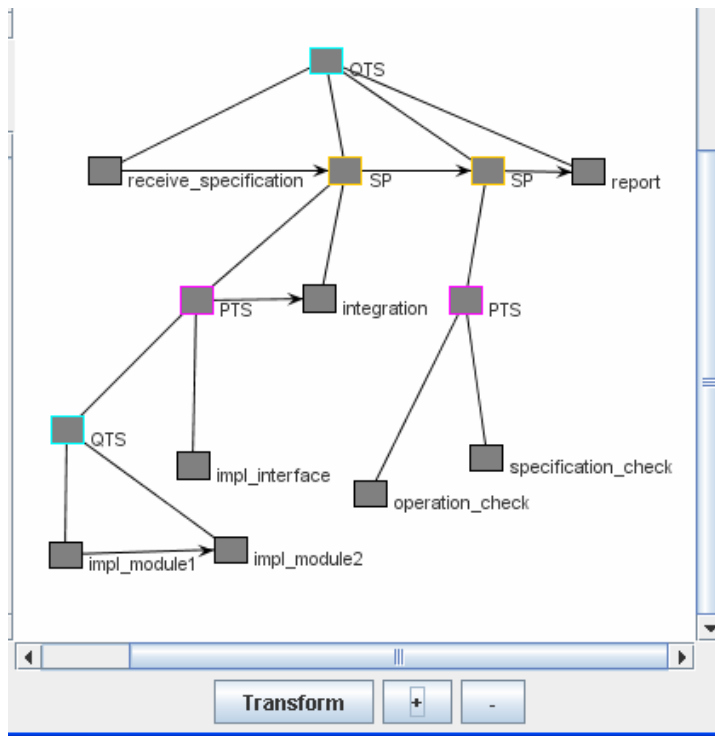


Figure 5.3: Block-structure Transformation of concrete workflow

- **Compatibility analysis**

The abstract workflow and the block-structure workflow process are compared shown as Figure5.4. The abstract workflow is shown in the left side, and the block-structure workflow process is shown in the right side. The abstract tasks are searched and mapped in the block-structure workflow process with “All Mapping” and “Select Mapping” buttons. The result of mapping is visualized by transforming related nodes into circle shape.

Moreover, all the results of compatibility analysis is informed by message window shown as Figure5.5. In this example, following two incompatibilities are informed: (1)The abstract task named “Implementation” cannot be found in the block-structure workflow process; (2)The task order between a pair of tasks named “specification_check” and “operation_check” is inconsistent with that of related tasks within block-structure workflow process.

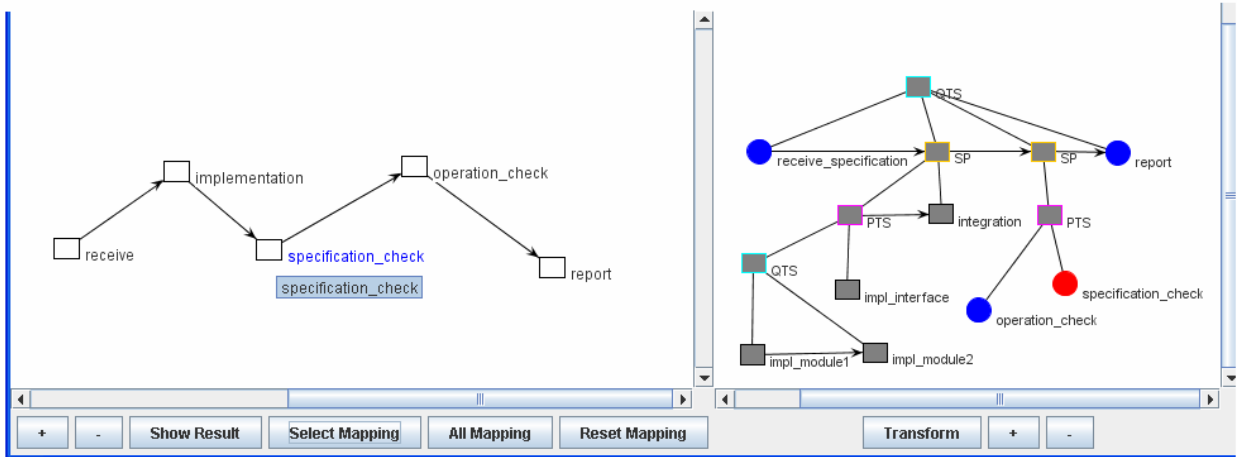


Figure 5.4: Compatibility analysis

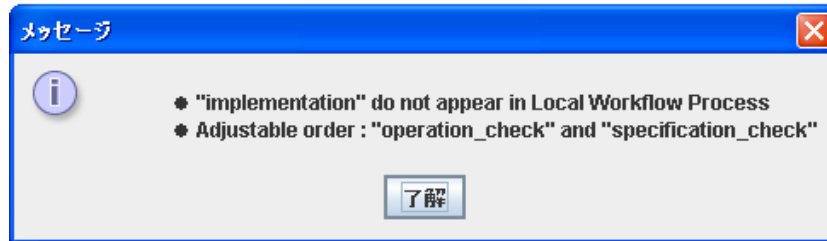


Figure 5.5: Incompatibilities informed by message window

5.2 Case Study

To explain our method, we use an example of offshore software development between Japanese company and Chinese company. In the offshore software development, Japanese company designs a specification of software and places an order with Chinese company. Chinese company develops software based on a specification which is designed by Japanese company .

In the offshore software development work, there are some conflicts caused by the different customs and understandings over software development, which leads to the difference of local modeling views of the whole workflow process.

5.2.1 Local Modeling Views of two organizations

Here we solve the problems using the compatibility analysis method proposed in this paper. First, both company build their own local modeling views of the whole interorganizational workflow based on their different knowledge backgrounds(task role, roadmap, interaction and so on). While building the local

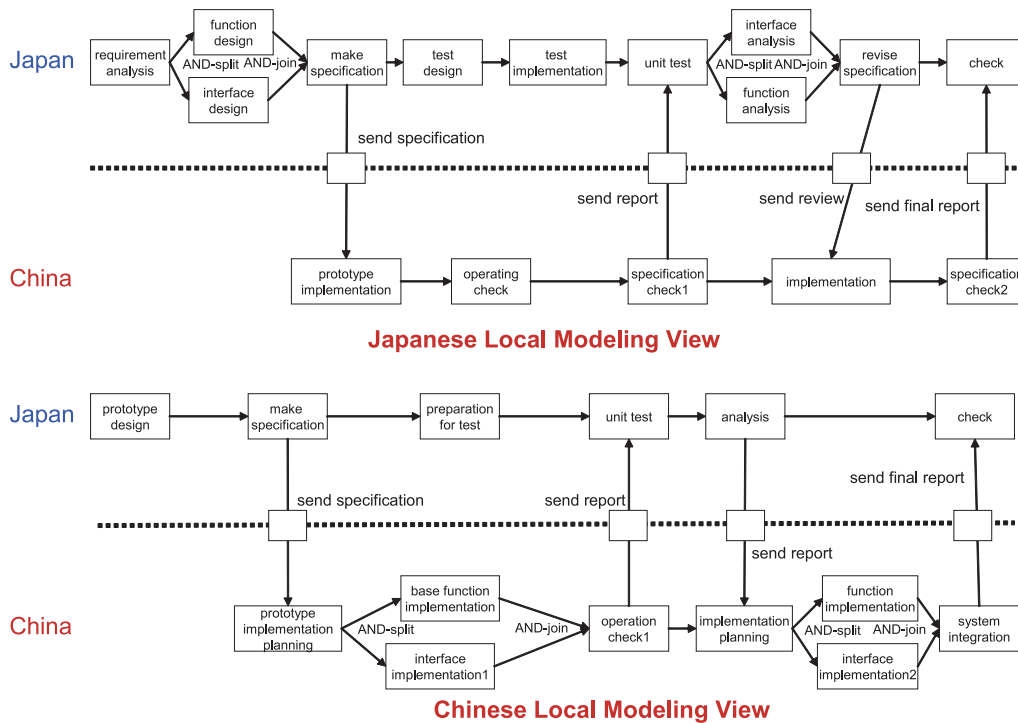


Figure 5.6: An example of China-Japan offshore software development workflow

modeling view, each company refers to the sharable knowledge about its own role.

For example, each company builds different local modeling view shown as Figure 5.6. The local modeling view of the Japanese company is shown in the upper part, and that of the Chinese company is shown in the lower part. Each local modeling view consists of local concrete workflow, abstract workflow of the other company and interaction events between them.

There are several incompatibilities between the two local modeling views. For example, in the Japanese local modeling view, abstract workflow includes tasks of “check specification”, but the Chinese local modeling view does not include such tasks.

5.2.2 Compatibility Analysis

Now we compare these local modeling views shown as Figure 5.6, and detect the incompatibilities between them. First, the interaction events between two companies are compared. The Japanese local modeling view includes following four interaction events: (1) send specification document; (2) send report; (3) send

review; (4)send result. If Chinese local modeling view also includes such interaction events and the order of events are same, there would be no incompatibility between the interaction parts of two organizations.

In this example, Chinese local modeling view includes above four interaction events and their order is also the same. Therefore, the interaction parts included in the two local modeling views are compatible.

Second, local concrete workflow and abstract workflow of the other company would be compared. To compare these workflows, we first transform the concrete workflow process into the block-structured workflow process. The block-structured workflow process which is transformed from the concrete workflow in

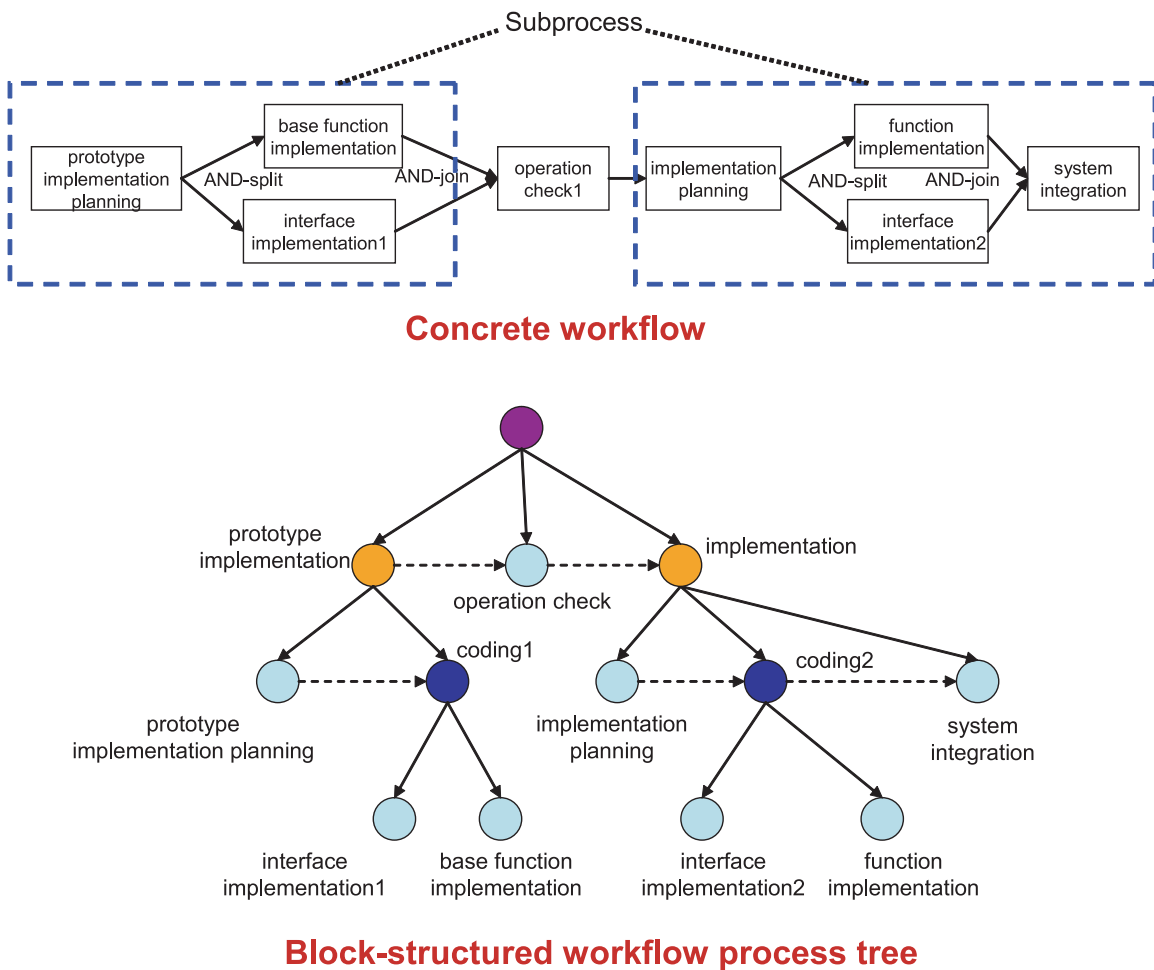


Figure 5.7: The concrete workflow and block-structured workflow process in the Chinese local modeling view

the Chinese local modeling view is shown in Figure 5.7. The concrete workflow in the Chinese local modeling view is shown in the upper part, which includes two subprocesses.

These subprocesses are also included in the block-structured workflow pro-

Table 5.1: The parameter of the block-structured workflow process

| Task | Precondition | Effect |
|-----------------------------------|---|---|
| prototype implementation | specification document | prototype interface , moduleA , moduleB |
| prototype implementation planning | specification document | prototype implementation plan |
| coding1 | prototype implementation plan | prototype interface , moduleA , moduleB |
| interface implementation1 | prototype implementation plan | prototype interface |
| base function implementation | prototype implementation plan | moduleA , moduleB |
| operation check | prototype interface , moduleA , moduleB | operation check result |
| implementation | review | interface, moduleA , moduleB, moduleC |
| implementation planning | review | implementation plan |
| interface implementation2 | implementation plan | interface |
| function implementation | implementation plan | moduleA , moduleB, moduleC |
| coding2 | implementation plan | moduleA , moduleB, moduleC, interface |
| system integration | moduleA , moduleB, moduleC, interface | system |

cess tree. Moreover, there are three blocks which are generated by replacing the set of tasks. We set parameters of precondition and effect for these subprocesses and blocks.

Then, the abstract workflow in the Japanese local modeling view and the block-structured workflow process which is transformed from the concrete workflow in the Chinese local modeling view would be compared. To compare these two workflows, we search and map the abstract tasks of abstract workflow process in the block-structured workflow process. When searching the abstract tasks, the precondition and effect of tasks are compared.

An example of the precondition and effect of the block-structured workflow process is shown as Table5.1 and those of the abstract workflow is shown as Table5.2.

Table 5.2: The parameters of the abstract workflow

| Task | Precondition | Effect |
|--------------------------|---|---|
| prototype implementation | specification document | prototype interface , moduleA , moduleB |
| operation check | prototype interface , moduleA , moduleB | operation check result |
| specification check1 | prototype interface , moduleA , moduleB | specification check result1 |
| implementation | review | interface, moduleA , moduleB, moduleC |
| specification check2 | interface, moduleA , moduleB, moduleC | specification check result2 |

The result of searching and mapping the abstract tasks is shown as following. The abstract workflow includes five abstract tasks and two abstract tasks cannot be mapped in the block-structured workflow process: “specification check1” and “specification check2”. From this result, the Chinese company is informed by the system that there are incompatibilities between the Japanese local modeling

view and the Chinese local modeling view: in the Chinese company, the tasks of “check specification1” and “check specification2” are not considered, however in the view of Japanese company, these tasks are expected to be executed by the Chinese company.

After such compatibility analysis, incompatibilities are checked out before the execution of the incompatible workflow. Therefore, the organizations can execute the whole interorganizational workflow with less potential conflicts that might be caused by the incompatibilities of the local modeling views.

Chapter 6 Discussion

In this research, we provide a new method of modeling the local modeling view of loosely coupled interorganizational workflow and compatibility analysis of multiple local modeling views. In this chapter, these methods are discussed.

Modeling of the local modeling view

The definition of the local modeling view of loosely coupled interorganizational workflow is presented in order to support multiple local modeling views. The local modeling view consists of three elements: the local concrete workflow, the abstract workflow and interaction events. We focus on the incompatibility of tasks and dependencies between them, and define the workflow with tasks and order constraints. The parameters of precondition and effect can be set for tasks.

The comparison of tasks in multiple local modeling views can be realized by this modeling. In previous researches, the dynamic behavior of workflow process has been focused on. In contrast to this, we focus on the static structure of workflow process, and the incompatibilities of static structure of workflow process can be detected by our method. However, the workflow process which includes complicated structure such as loop task routing construct and compensation routing cannot be modeled by our modeling method.

Algorithm of compatibility analysis

Our method of compatibility analysis can be mainly divided into three procedures: event comparison, block-structure transformation and the abstract-concrete workflow comparison. In the event comparison procedure, interaction events are compared. Therefore, there must be some rules or ontology of describing interaction events shared by all the organizations. In the abstract-concrete workflow comparison procedure, precondition and effect of tasks are compared. Therefore, some rules or ontology of task description are also needed.

In previous researches, detection of the incompatibilities of interaction protocols has been realized, but those of tasks and task orderings in multiple workflow processes. Our method of compatibility analysis realizes the comparison of tasks and task orderings.

In this research, we focus on the loosely coupled interorganizational workflow. For example, electronic commerce and software development are executed based on this form of interoperability. Particularly, our method is useful for such collaborative tasks that the participants from the organizations have different culture and customs such as intercultural collaboration. Moreover, compatibility analysis of multiple local modeling views benefits ad-hoc collaborative tasks such as CtoC which consumers may not describe the workflow process.

In the future we hope to extend the compatibility analysis approach in several directions. When incompatibilities are checked out, the problem of how to establish the negotiation mechanism among organizations is also expected to be solved. In addition, it is important to evaluate the efficiency of the proposed modeling method. Furthermore, the incompatibilities in dynamic behavior of the interorganizational workflow are also expected to be checked out.

Chapter 7 Summary

In this paper, we propose a new method of modeling loosely coupled interorganizational workflow that supports the organizations to have their own local views of modeling the workflow process. First, a series of formalism definitions are given related to the new method of interorganizational workflow modeling. Each organization has three knowledge before collaborative tasks execution: complete knowledge about the local workflow, incomplete knowledge about the workflow of other organizations and sharable knowledge such as task role, roadmap and so on. The local modeling view of interorganizational workflow consists of the local concrete workflow, the abstract workflow of other organizations and interaction events. Each organization designs the local modeling view which consists of following elements.

- **The concrete workflow**

This is the detailed workflow within the local organization. All the tasks and control flows are described completely, and subprocesses which includes some tasks can be set by the organization.

- **The abstract workflow**

This is the workflow of interacting organizations. The abstract workflow consists of some abstract tasks which are regarded as an abstraction of set of the concrete tasks.

- **Event sequence chart**

This is the interaction protocols between two organizations. Event sequence chart consists of interaction events and their orderings.

Then, by comparing different local modeling views from the organizations, the incompatibilities are checked out. In this paper, the incompatibilities of following elements can be detected.

- interaction protocol among individual organizations
- tasks defined in local workflow executed locally by individual organization
- ordering constraints of tasks in local workflows executed locally by individual organization

In compatibility analysis, the constraint that each organization cannot disclose the local concrete workflow completely to other organizations must be considered. Then, the abstract workflow and the interaction events are shared by the organizations, and the concrete workflow is concealed. Sharable elements are compared with the related elements in the local modeling view.

To compare the abstract workflow and the concrete workflow, the block-structured workflow process is presented. Through transforming the concrete local workflow process into block-structured form, the comparison of concrete workflow and abstract workflow can be conducted. Block-structured transformation of the concrete workflow is realized by replacing the set of concrete tasks which belong to the same task routing construct or the same subprocess. In the result of block-structured transformation, virtual tasks such as block and subprocess are generated, and can be mapped to the abstract tasks in the abstract workflow.

The whole procedure of compatibility analysis among the local modeling views of the interorganizational workflow is as following.

1. Design the local modeling view
2. Exchange the abstract workflow and interaction events
3. Comparison of interaction events
4. Block-structured transformation of the concrete workflow
5. Comparison of the abstract workflow and the block-structured workflow process tree of the related concrete workflow

Then, the loosely coupled interorganizational workflow can be executed based on the compatible multiple local modeling views from the organization.

Acknowledgments

I would like to express my sincere gratitude to Professor Toru Ishida at Graduate School of Informatics, Kyoto University, for invaluable advice and discussion.

I am also grateful to my research adviser, Associate Professor Mizuho Iwaihara at Department of Social Informatics, Kyoto University, and Associate Professor Nobuaki Arai Department of Social Informatics, Kyoto University.

I would like express special thanks to Donghui Lin at Ishida Laboratory for his wisdom, kindness and help. I also thank all the members of Ishida Laboratory at Kyoto University for their support.

References

- [1] Blum, A. and Furst, M.: Fast Planning Through Planning Graph Analysis, *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, pp. 1636–1642 (1995).
- [2] Durfee, E. and Lesser, V.: Using partial global plans to coordinate distributed problem solvers, *the Tenth International Joint Conference on Artificial Intelligence*, pp. 875–883 (1987).
- [3] Grefen, P., Aberer, K., Hoffner, Y. and Ludwig, H.: CrossFlow: Cross-Organizational Workflow Management in Dynamic Virtual Enterprises, *International Journal of Computer Systems Science & Engineering*, Vol. 5, pp. 277–290 (2000).
- [4] Krishna, S., Sahay, S. and Walsham, G.: Managing Cross-Cultural Issues in Global Software Outsourcing, *Communications of the ACM*, Vol. 47, pp. 62–66 (2004).
- [5] Lenz, K. and Oberweis, A.: Modeling Interorganizational Workflows with XML Nets, *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)-Volume 7*, Washington, DC, USA, IEEE Computer Society, p. 7052 (2001).
- [6] Lin, D.: An Interorganizational Workflow Model Based on Agent and ECA Rules, Master's thesis, Shanghai Jiao Tong University (2005).
- [7] Liu, D.-R. and Shen, M.: Workflow Modeling for Virtual Processes: an Order-Preserving Process-View Approach, *Information Systems*, Vol. 28–6, pp. 505–532 (2003).
- [8] Meng, J., Su, S. Y. W., Lam, H. and Helal, A.: Achieving Dynamic Inter-organizational Workflow Management by Integrating Business Processes, Events, and Rules, In Annual Hawaii International Conference on System Sciences (HICSS'02), Big Island, Hawaii, USA (2002).
- [9] Merz, M., Liberman, B. and Lamersdorf, W.: Using mobile agent to support inter-organizational workflow management, *Applied Artificial Intelligence*, Vol. 11, No. 6, pp. 551–572 (1997).
- [10] Shen, M. and Liu, D.-R.: Coordinating Interorganizational Workflows

- based on Process-Views, *the 12th International Conference on Database and Expert Systems Applications (DEXA '01)*, Munich, Germany, pp. 274–283 (2001).
- [11] W.M.P. van der Aalst: Inheritance of Interorganizational Workflows: How to Agree to Disagree Without Loosing Control, BETA Working Paper Series, WP 46, Eindhoven University of Technology, Eindhoven, 2000.”, (2000).
 - [12] W.M.P. van der Aalst and Kumar, A.: XML Based Schema Definition for Support of Inter-organizational Workflow, University of Colorado and University of Eindhoven report (2000).
 - [13] W.M.P. van der Aalst: The Application of Petri Nets to Workflow Management, *The Journal of Circuits, Systems and Computers*, Vol. 8, pp. 21–66 (1998).
 - [14] W.M.P. van der Aalst: Modeling and Analyzing Interorganizational Workflows, *International Conference on Application of Concurrency to System Design (CSD'98)* (Lavagno, L. and Reisig, W.(eds.)), IEEE Computer Society Press, pp. 1–15 (1998).
 - [15] W.M.P. van der Aalst: Process-Oriented Architectures for Electronic Commerce and Interorganizational Workflow, *Information Systems*, Vol. 24, No. 8, pp. 639–671 (1999).
 - [16] W.M.P. van der Aalst: Loosely Coupled Interorganizational Workflows: Models and Analyzing Workflows Crossing Organizational Boundaries, *Information and Management*, Vol. 37, pp. 67–75 (2000).
 - [17] W.M.P. van der Aalst, A.H.M. ter Hofstede and Kiepuszewski, B.: Workflow Patterns, *Distributed and Parallel Databases*, Vol. 14–3, pp. 5–51 (2003).
 - [18] W.M.P. van der Aalst and van Hee, K.: *Workflow Management: Models, Methods, and Systems*, the MIT Press (2004).
 - [19] W.M.P. van der Aalst: Interorganizational Workflow: An Approach based on Message Sequence Charts and Petri Nets, *Systems Analysis - Modelling - Simulation*, Vol. 34–3, pp. 335–367 (1999).
 - [20] W.M.P. van der Aalst, ter Hofstede, A., Kiepuszewski, B. and Barros, A.:

Advanced Workflow Patterns, *7th International Conference on Cooperative Information Systems (CoopIS 2000)* (Etzion, O. and Scheuermann, P.(eds.)), Springer-Verlag, Berlin, pp. 18–29 (2000).

- [21] van der Krogt, R. and de Weerdt, M.: Self-interested Planning Agents Using Plan Repair, *Proceedings of the ICAPS 2005 Workshop on Multiagent Planning and Scheduling*, pp. 36–44 (2005).
- [22] WfMC: The Workflow Management Coalition Specification - Terminology & Glossary, WfMC-TC-1011 (1999).