Compatibility Analysis of Local Process Views in Interorganizational Workflow

Donghui Lin
Department of Social Informatics, Kyoto University
Yoshida-Honmachi, 606-8501, Kyoto, Japan
lindh@ai.soc.i.kyoto-u.ac.jp

Abstract

In an interorganizational collaboration environment, due to different cultural backgrounds, participants from different organizations always have different views of modeling workflow process. Moreover, organizations require preserving privacy and autonomy of its own workflow. Therefore, workflow inter-visibility is expected to be as little as collaborations need. This paper proposes an approach of modeling loosely coupled interorganizational workflow considering different views of organizations. In this approach, organizations have their own local process views of modeling workflow process instead of sharing a predefined global workflow. Furthermore, we design compatibility analysis mechanism for different local process views to detect incompatibilities among organizations. Finally, we provide a demo tool for the proposed mechanism.

1. Introduction

In recent years, computer-mediated collaboration has been rapidly increasing within individual organizations or between different organizations across nations. The process of collaboration can be modeled as interorganizational workflow, i.e., workflows crossing organizational boundaries inside a company or across companies. Previous researches on interorganizational workflow benefit people and enterprises. However, there are several typical problems in modern interorganizational workflows. One problem is that participants from different organizations have different cultural and knowledge backgrounds, which might have a great impact on the way that the cooperative work is conducted. 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 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 [5]. Because of such conflicts, many problems occur due to disagreements on views regarding workflow-process modeling. Another problem lies in the fact that individual organizations frequently want to keep their own local workflow secret. Though cooperation of organizations needs some visible interactions and data sharing, it is also important for the organizations to preserve their internal experiences and knowledge as much as possible. It is unreasonable for an organization to share its entire local workflow with other organizations. Therefore, when modeling the interorganizational workflow, how to preserve privacy and autonomy of local workflows of organizations should be considered.

Previously, van der Aalst proposed workflow interoperability in building interorganizational workflow model. He categorized several types of workflow interoperability in the interorganizational workflow, such as capacity sharing, chained execution, subcontracting, case transfer, and loosely coupled workflows [8]. The interorganizational workflow model in this paper 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 [9].

In order to support different cultural backgrounds of organizations and preserve privacy for the organizations at the best, we propose a new method of modeling interorganizational workflow taking different process modeling views of different organizations into consideration. Therefore, organizations do not share a global interorganizational workflow process with this method, but have their own local views of how to model the whole workflow process with their unique backgrounds and knowledge, which we call local process views. The local process views should be compatible with each other for execution. By sharing parts of the data among organizations and conducting 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

The 9th IEEE International Conference on E-Commerce
Technology and The 4th IEEE International Conference
on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007)
0-7695-2913-5/07 $25.00

can be established.

This paper is organized as follows: Section 2 introduces related works. Section 3 provides interorganizational workflow model based on local process views. In Section 4, a detailed compatibility analysis mechanism for local process views is presented. A demo tool is shown in Section 5 to support organizations to detect incompatibilities, followed by the conclusion in Section 6.

2. Related Work

Previous approaches of interorganizational workflow include the perspective of view [2], workflow interoperability [9], organization coordination [1] [3], agreement and contracts between organizations [4], workflow specification languages and so on. In [10], the author 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, the 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. The approach is technically sound, however, it supposes all the organizations with different cultural backgrounds could accept a common global workflow and cooperate with each other in the framework well. In [2], the authors present 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, however they concentrated on the interaction and cooperation but did not explain how the local workflows could adapt to the global workflow without potential conflicts. In this paper, we not only present the important issue of interaction protocols between organizations, but also consider the different process views of organizations.

3. Interorganizational Workflow Based on Local Process Views

3.1. Architecture

In traditional loosely coupled interorganizational workflow, there is a common global view of interorganizational workflow. Without considering the potential conflicts caused by different cultural backgrounds, there is an assumption that all organizations accept a pre-defined global workflow. In this paper, we do not create a single global view of interorganizational workflow. Instead, each organization has its own local process view. The local process view includes concrete information about the local workflow, interaction protocols between organizations, and abstract-level workflow processes of other organizations that interact with the local organization through interaction protocols. Each organization can selectively share its own abstract-level workflow with other organizations for reference.

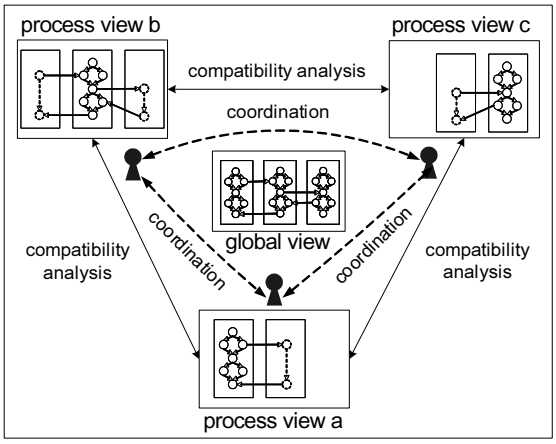

Figure 1. Interorganizational workflow architecture based on local process views.

Figure 1 outlines the architecture for the interorganizational workflow model supporting multiple local process views. In this architecture, each organization has a local process view. Since there is no global view, they conduct compatibility analysis with each other and coordinate with each other if there are conflicts. 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. Definition of Local Process View

In previous research, interorganizational workflows are mainly modeled by Petri nets. There are several reasons for using 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 [10]. In this research, the local process views might frequently change for eliminating detected incompatibilities, therefore, flexibility is the most important factor when we choose the workflow model. Work-

The 9th IEEE International Conference on E-Commerce
Technology and The 4th IEEE International Conference
on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007)
0-7695-2913-5/07 $25.00 © 2007 IEEE

flow graphs were developed with flexibility in mind. For this purpose we use workflow graphs, rather than Petri nets, to model the local process views.

Definition 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 $LPV = (I, WF_0, WF_1, WF_2, \ldots, WF_n, ESC)$, 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 $k \in \{1, 2, \ldots, n\}$:$WF_k$ is the expected workflow process of its partner k, which is modeled with limit information or its own consideration (a workflow process is defined by Definition 2); and

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

In [7], the author distinguishes two representation paradigms for business process modeling (BPM) language, namely the Graph-oriented BPM and the block-oriented BPM. In this paper, the definition of Workflow Process is graph-oriented. Further, we define it as a structured process graph [7].

Definition 2 (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
- c is the intra-connection type of the task such that $c \in C \rightarrow \{null, AND{-}split, AND{-}join, OR{-}split, OR{-}join, XOR - split, XOR - join\}$
- r is the inter-connection type of the task and $r \in R \rightarrow \{null, in, out\}$;

(3) 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.

In previous research, [8] 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 [6]. The representation of Message Sequences Charts is intuitive and focuses on messages between communication entities. In this paper, we focus on the modeling phase of interorganizational workflows. Therefore, we define the interaction events among the organizations by using a similar approach to message sequence charts, i.e., an Event Sequence Chart.

Definition 3 (Event Sequence Chart) An event sequence chart is defined as a tuple $ESC = (I, E, from, to, \{\preceq_i\}_{i \in I})$ where

(1) I is a finite set of organizations;

(2) E is a finite set of events;

(3) $from$ and to are functions from E to I; and

(4) For each $i \in I$:$\preceq_i$ is a partial order on $\{?e \mid e \in E \; and \; to(e) = i\} \cup \{!e \mid e \in E \; and \; from(e) = i\}$ where $?e$ represents a sending event and $!e$ represents a receiving event.

3.3. An Intercultural Collaboration Case

To explain the proposed workflow model based on local process views, we demonstrate a case of intercultural software development collaboration between groups from two nations: a Japanese group and a Chinese group. In developing software, they have some rough agreements on the development assignment, e.g. the Japanese group designs the software specifications and the Chinese group develops the software based on the designed specifications. Considering the two groups come from different nations, it is quite possible that there are some conflicts caused by different customs and misunderstandings over software development, which leads to differences in local process views of the whole workflow process. When creating the whole workflow process for the collaboration development, each group wants to keep the detailed local workflow process secret. Therefore, they model the workflow process by their own customs. Figure 2 shows an example of their respective local process views.

In software development collaboration, the following conflicts always occur between the two countries: (1) In the Japanese groups, software specifications are always modified in parallel with development. However, in Chinese groups, specifications are almost totally designed before development. The Chinese group might not be able to understand why *specification check* is required. Therefore, when they model their local process view, *specification check* is not considered; (2) In the Japanese groups, the software quality management is more strict. Therefore, in the Chi-

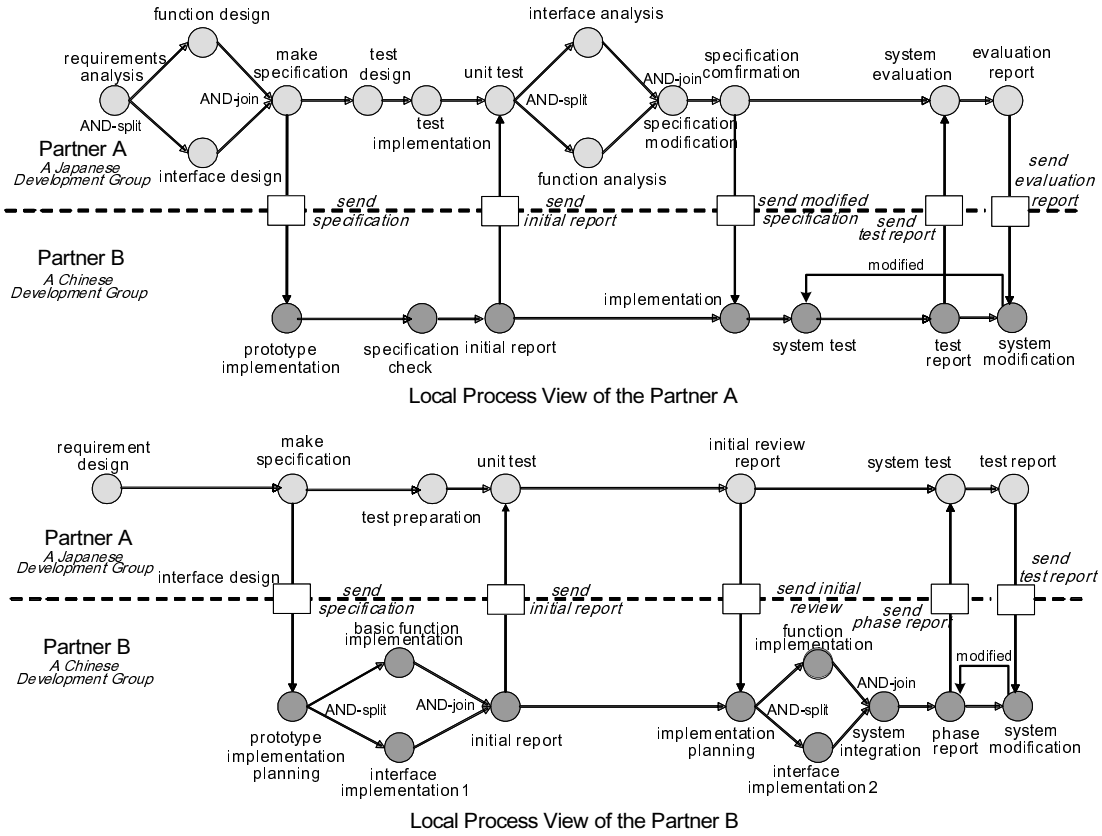

Figure 2. Example of local process views of an interorganizational workflow process.

nese groups, software tests are conducted less frequently. In figure 2, the Chinese group develops the software, but they think that *system test* should be done by the Japanese group, which is totally different with the consideration of the Japanese group. Because of above mutual misunderstandings, local process views of the two groups differs much with each other's 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.

4. Compatibility Analysis

4.1. Cases of Incompatibility

In the interorganizational workflow model based on local process views, each organization has a different view. Therefore there might be many conflicts between local process views of organizations. Before designing compatibility analysis mechanism, we first list cases of incompatibility between local process views.

Type 1: incompatibility of interaction protocols

In the interorganizational collaboration environment, the interaction protocol is the most important issue for all partners. In this research, we use Event Sequence Charts to define the interaction protocol between partners. However, since different partners have different considerations for the interaction, following cases of incompatibility might occur in the interaction protocols (we consider the situation of two partners).

(1) *Unilateral Event*: there exists an interaction event in one local process view but does not exist in the other;

(2) *Inconsistent Direction*: the sender of an interaction event in one local process view is the receiver of the other;

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

(4) *Disordered Sequence*: there exists two events that have opposite partial orders in two local process views.

The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007)
0-7695-2913-5/07 $25.00 © 2007 IEEE

Type 2: incompatibility of workflow processes

In a local process view, the local workflow process is represented at a concrete level. Moreover, the workflow processes of partners are also considered with the modeling view of the local partner. By this means, the potential conflicts and different views of the same interorganizational workflow are well embedded. The incompatibility of workflow processes include following cases.

(1) *Unilateral 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. This type of compatibility is judged by the comparison of preconditions and effects of related tasks;

(2) *Disordered Tasks*: there exist two tasks that have opposite sequential orders in two local process views.

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

4.2. Compatibility Analysis Mechanism

Since we mainly categorize incompatibilities in two types, we also divide our compatibility analysis mechanism into two phases. In this paper, we consider the situation of compatibility analysis between two organizations.

Phase 1: compatibility analysis of interaction protocols

Obviously, the incompatibility of interaction protocols might easily cause incompatibility of the workflow processes. Therefore, when conducting compatibility analysis, interaction protocols should be first analyzed, which includes two parts, i.e., the content and order of interaction events between two organizations.

The algorithm of compatibility analysis of interaction protocols is shown in **Algorithm 1**, which can detect all incompatibility cases of Type 1 we discussed in Section 4.1.

Phase 2: compatibility analysis of workflow processes

After Phase 1 of compatibility analysis, the incompatibilities of the interaction protocols are expected to be eliminated by coordination between organizations. If comparing interaction protocols results in compatibility, the workflow processes of two local process views should be further analyzed.

However, for each workflow process, the local organization models it as a relatively concrete one but other organizations model it at an abstract level based on their limited knowledge about the local organization. Therefore, there are following difficulties when analyzing the abstract-level and concrete-level workflow process of a same organization: (1) the structure of workflow process modeled by the local organization is always far more complicated and (2) some parts in the abstract-level workflow processes cannot be directly mapped into the concrete-level workflow processes.

Algorithm 1 Compatibility analysis phase 1

procedure Compatibility-Analysis-1(LPV, LPV')
1: $IncompatibleFlag \leftarrow false$
2: **for all** $e \in E$ **do**
3: **if** $(from(e) = Org_A) \wedge (to(e) = Org_B)$ **then**
4: **if** $(e \notin E')$ **then**
5: $IncompatibleFlag \leftarrow true$
6: display $Unilateral\ Event\ e$
7: **else if** $to'(e) = Org_A$ **then**
8: $IncompatibleFlag \leftarrow true$
9: display $Inconsistent\ Direction\ e$
10: **else if** $((t, e) \in F_0^I) \wedge ((t', e) \in F_A^{I'}) \wedge (eff(t) \neq eff(t'))$ **then**
11: $IncompatibleFlag \leftarrow true$
12: display $Inconsistent\ Condition\ t$
13: **end if**
14: **end if**
15: **if** $(to(e) = Org_A) \wedge (from(e) = Org_B)$ **then**
16: **if** $(e \notin E')$ **then**
17: $IncompatibleFlag \leftarrow true$
18: display $Unilateral\ Event\ e$
19: **else if** $from'(e) = Org_A$ **then**
20: $IncompatibleFlag \leftarrow true$
21: display $Inconsistent\ Direction\ e$
22: **else if** $((e, t) \in F_0^I) \wedge ((e, t') \in F_A^{I'}) \wedge (prec(t) \neq prec(t'))$ **then**
23: $IncompatibleFlag \leftarrow true$
24: display $Inconsistent\ Condition\ t$
25: **end if**
26: **end if**
27: **for all** $\{ e_i, e_j \} \subseteq E \cup E'$ **do**
28: **if** $((?e_i \vee !e_i) \preceq_A (?e_j \vee !e_j)) \wedge ((?e_j \vee !e_j) \preceq_B (?e_i \vee !e_i))$ **then**
29: $IncompatibleFlag \leftarrow true$
30: display $Disordered\ Sequence\ e_i, e_j$
31: **end if**
32: **end for**
33: **end for**
34: **if** $IncompatibleFlag = true$ **then**
35: **return** Incompatible
36: **else**
37: **return** Compatible
38: **end if**

For example, a task in the abstract-level workflow process might be equivalent to a group of tasks in the related concrete-level workflow process. To compare an abstract workflow with a concrete workflow, we need some additional semantic meanings for the abstractions of tasks. We

therefore induce subprocesses into the concrete-level workflow process for the local organizations.

Definition 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 subprocess;

(2) TS denotes a finite set of 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-connected elements. 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 additional abstract-level information of a concrete-level workflow process. Figure 3 shows an example to use subprocess to abstract current workflow process.

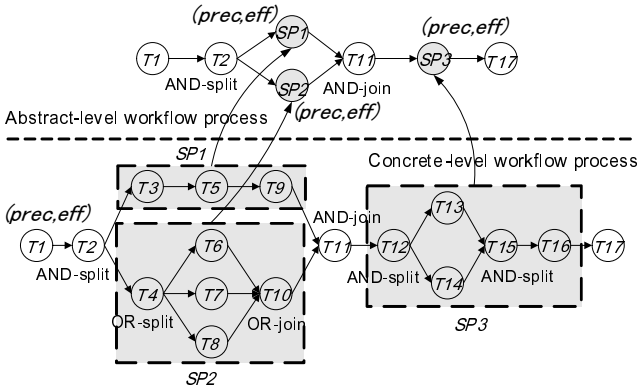

Figure 3. Subprocess in workflow process.

Since workflow process is defined as a structured process graph, it can be reduced to a single node [7]. Here we can reduce the graph by blocks. A block is a unit of representation that minimally specifies the behavioral pattern of a process flow. We used the following types of blocks that are discussed in this paper, iterative block, sequential block, branch block (including parallel block, conditional selective block and non-conditional selective block). Therefore, we can transform a concrete-level workflow process with its abstract-level subprocesses into a block-structured hierarchical workflow tree so that the tree contains not only concrete-level tasks but also abstract-level subprocesses. Then we can compare the abstract-level workflow process and the related concrete-level workflow process.

The transformation from a workflow process graph into a hierarchical workflow process tree begins from the innermost sequential block. The whole process for the example in Figure 3 is as follows. (1) Replace $\{T_3, T_5, T_9\}$ with sequential block QB_1; (2) Replace QB_1 with SP_1; (3) Replace $\{T_{13}, T_{14}\}$ with parallel block PB_1 and $\{t_6, t_7, t_8\}$ with non-conditional selective block SB_1; (4) Replace $\{t_{12}, PB_1, t_{15}, t_{16}\}$ with subprocesses SP_3 and $\{T_4, SB_1, T_{10}\}$ with sequential block QB_2; (5)Replace QB_2 with subprocess SP_2; (6) Replace $\{SP_1, SP_2\}$ with parallel block PB_2; (7) Replace $\{T_1, T_2, PB_2, T_{11}, SP_3, T_{17}\}$ with sequential block QB_3; and (8) The entire concrete workflow becomes a single sequential block node, QB_3. During the above replacement, a workflow tree is created as shown in Figure 4.

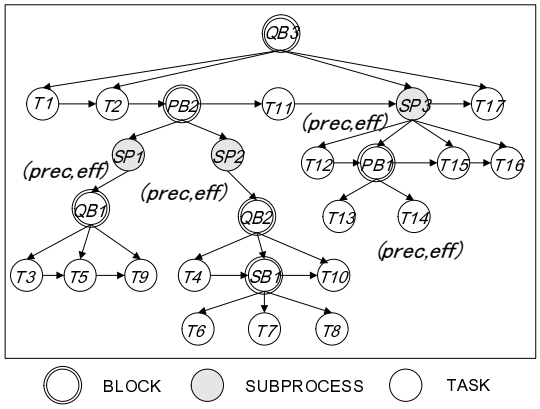

Figure 4. Graph-based workflow transformed into hierarchical workflow.

Algorithm 2 Task weight computation

procedure Task-Weight-Computation(WF)

1: $split \leftarrow \{AND-split, OR-split, XOR-split\}$
2: $join \leftarrow \{AND-join, OR-join, XOR-join\}$
3: **for all** $t \in T$ **do**
4: $\quad weight(t) \leftarrow 0$
5: **end for**
6: $weight[t_s] \leftarrow 1\ u \leftarrow t_s\ max_w \leftarrow 0$
7: **for all** $v \in succ[u]$ **do**
8: $\quad$ **if** $weight[v] = 0$ **then**
9: $\quad\quad$ **if** $c[u] = null$ **then**
10: $\quad\quad\quad weight(v) \leftarrow weight(u)$
11: $\quad\quad$ **else if** $c[u] \in split$ **then**
12: $\quad\quad\quad weight(v) \leftarrow weight(u) + 1$
13: $\quad\quad$ **else if** $c[u] \in join$ **then**
14: $\quad\quad\quad weight(v) \leftarrow weight(u) - 1$
15: $\quad\quad$ **end if**
16: $\quad\quad$ **if** $weight[v] > max_w$ **then**
17: $\quad\quad\quad max_w \leftarrow weight[v]$
18: $\quad\quad$ **end if**
19: $\quad$ **end if**
20: **end for**
21: **return** max_w

The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007)
0-7695-2913-5/07 $25.00 © 2007 IEEE

The algorithms of the transformation process is shown in **Algorithm 2** and **3**. **Algorithm 2** is used to compute weights of all the tasks. The start task node has weight 1, which means it is the most outer node in the network. If there is a split task node, then the weight of its successor node would be larger than it. If there is a join task node, then the weight of the successor node would be less than it. As a result, the most inner node has the largest weight. **Algorithm 3** uses the task weight to decide what kind of blocks should be replaced in every step. Besides blocks, the subprocesses are also replaced in certain time. The algorithm stops when the workflow process tree is totally created.

Algorithm 3 Workflow process transformation

procedure Process-Transform(WF, WF_{SP}, max_w)
1: detect iterative block
2: **if** $IterativeDetected = true$ **then**
3: delete iterative blocks and update WF
4: **end if**
5: **repeat**
6: detect sequential block with max_w
7: **if** $SequentialDetected = true$ **then**
8: replace related task nodes with sequential block
9: detect subprocess
10: **if** $SubprocessDetected = true$ **then**
11: replace related task nodes with subprocess
12: **end if**
13: update WF and max_w
14: **else**
15: detect branch block with max_w
16: **if** $BranchDetected = true$ **then**
17: replace related task nodes with branch block
18: detect subprocess
19: **if** $SubprocessDetected = true$ **then**
20: replace related task nodes with subprocess
21: **end if**
22: update WF and max_w
23: **end if**
24: **end if**
25: **until** only one node is remained
26: **return** workflow process tree

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 workflow process in the transformed workflow tree, i.e., for each task $t \in T$, check whether there exists an element in the workflow tree so that t is equivalent to tt in semantic meaning, $prec(t) = prec(tt)$ and $eff(t) = eff(tt)$; (2) Compare the order of tasks in the abstract-level workflow process and the related task order in the workflow tree. We can also detect cases of incompatibility described in Section 4.1. The detailed algorithm is shown in **Algorithm 4**.

Algorithm 4 Compatibility analysis phase 2

procedure Compatibility-Analysis-2(LPV, LPV')
1: $IncompatibleFlag \leftarrow false$
2: $TWF_0 \leftarrow$ Process-Transform(WF_0, WF_{SP}, max_w)
3: **for all** $t \in T_A$ **do**
4: $map[t] \leftarrow null$
5: **for all** $tt \in TT_0$ **do**
6: /*TT_0 is the task set of TWF_0.*/
7: **if** ($prec(t)$=$prec(tt)$)$\wedge$($eff(t)$=$eff(tt)$) **then**
8: $map[t] \leftarrow tt$
9: break
10: **end if**
11: **end for**
12: **if** $map[t] = null$ **then**
13: $IncompatibleFlag \leftarrow true$
14: display $Unilateral\ Task\ t$
15: **end if**
16: **end for**
17: **for all** $(t_i, t_j) \in F_A^N$ **do**
18: **if** $(map[t_j], map[t_i]) \in TF_0^N$ **then**
19: /*TF_0^N is the internal flow set of TWF_0.*/
20: $IncompatibleFlag \leftarrow true$
21: display $Disordered\ Tasks\ t_i, t_j$
22: **end if**
23: **end for**
24: **if** $IncompatibleFlag = true$ **then**
25: **return** Incompatible
26: **else**
27: **return** Compatible
28: **end if**

5. Implementation

We have implemented a demo tool for compatibility analysis between local process views based on the compatibility analysis mechanism we proposed in this paper. In this tool, user can design the local process view, including interaction protocol, local workflow process and 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 interaction protocols and abstract-level workflow process of the local organization that is designed by the modeling view of the other organization. The implemented demo tool includes following three modules: (1) workflow process design module; (2) workflow process transformation module; (3) compatibility analysis module.

Figure 5 shows the screenshot of second phase compatibility analysis by Chinese development group (Example in Figure 2). The first phase is to analyze different interaction

The 9th IEEE International Conference on E-Commerce Technology and The 4th IEEE International Conference on Enterprise Computing, E-Commerce and E-Services(CEC-EEE 2007)
0-7695-2913-5/07 $25.00 © 2007 IEEE

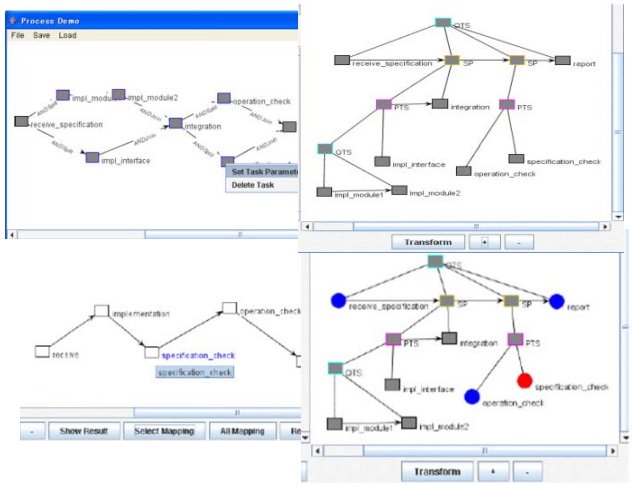

Figure 5. A tool for compatibility analysis.

protocols designed by Chinese group and Japanese group. By using this tool, Chinese development group adds some subprocesses in its local workflow so that both abstract-level and concrete-level are involved in the second analysis phase. Further, in order to compare the abstract-level workflow with the concrete-level 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 groups, 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.

6. Conclusion and Future Work

This paper presents a modeling approach to interorganizational workflow considering different process views of organizations and privacy preservation. In this approach, organizations have their own local process views instead of sharing a pre-defined common global workflow. Furthermore, we design the compatibility analysis mechanism for different local process views to detect incompatibilities among organizations. The mechanism is divided into two phases for interaction protocols and workflow processes respectively. By conducting compatibility analysis between organizations, potential conflicts can be detected before start of workflow execution. To demonstrate the proposed interorganizational workflow model and analysis algorithms in this paper, we provide a tool of compatibility analysis for different local process views. Future work will mainly concentrate on the coordination mechanism for eliminating incompatibilities. The issue of semantics in local process views and the complexity analysis will also be considered in the future.

Acknowledgements

The author wishes to thank Prof. Toru Ishida at Department of Social Informatics, Kyoto University for his advice and guidance, and Mr. Yasumasa Mita at IBM Japan, Ltd. for his collaborative work. This research was partially supported by a Grant-in-Aid for Scientific Research (A) (18200009, 2006-2008) from Japan Society for the Promotion of Science (JSPS).

References

[1] E. Andonoff, L. Bouzguenda, C. Hanachi, and C. Sibertin-Blanc. Using agent technology for coordination in loose inter-organizational workflow. In *WETICE '05: Proceedings of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 33–38, Washington, DC, USA, 2005. IEEE Computer Society.

[2] I. Chebbi, S. Dustdar, and S. Tata. The view-based approach to dynamic inter-organizational workflow cooperation. *Data Knowl. Eng.*, 56(2):139–173, 2006.

[3] M. Divitini, C. Hanachi, and C. Sibertin-Blanc. Inter-organizational Workflows for Entreprise Coordination . In A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, editors, *Coordination of Internet Agents: Models, Technologies and Applications* , pages 369–398. Springer, 2001.

[4] P. Grefen, K. Aberer, H. Ludwig, and Y. Hoffner. Crossflow: Cross-organizational workflow management for service outsourcing in dynamic virtual enterprises. *IEEE Data Engineering Bulletin*, 24(1):52–57, 2001.

[5] S. Krishna, S. Sahay, and G. Walsham. Managing cross-cultural issues in global software outsourcing. *Commun. ACM*, 47(4):62–66, 2004.

[6] S. Mauw and M. A. Reniers. An algebraic semantics of basic message sequence charts. *The Computer Journal*, 37(4):269–277, 1994.

[7] J. Mendling, K. Lassen, and U. Zund. Transformation strategies between block-oriented and graph-oriented process modelling languages. Technical Report JM-200510 - 10, WU Vienna, October 2005.

[8] W. M. P. van der Aalst. Process-oriented architectures for electronic commerce and interorganizational workflow. *Inf. Syst.*, 24(9):639–671, 1999.

[9] W. M. P. van der Aalst. Loosely coupled interorganizational workflows: Modeling and analyzing workflows crossing organizational boundaries. *Information & Management*, 37(2):67–75, 2000.

[10] W. M. P. van der Aalst. Inheritance of interorganizational workflows to enable business-to-business e-commerce. *Electronic Commerce Research*, 2(3):195–231, 2002.