PAPER Special Section on Knowledge-Based Software Engineering

Coordination of Local Process Views in Interorganizational Business Process

Donghui LIN†a), *Member and* **Toru ISHIDA**†, *Fellow*

SUMMARY Collaborative business has been increasingly developing with the environment of globalization and advanced information technologies. In a collaboration environment with multiple organizations, participants from different organizations always have different views about modeling the overall business process due to different knowledge and cultural backgrounds. Moreover, flexible support, privacy preservation and process reuse are important issues that should be considered in business process management across organizational boundaries. This paper presents a novel approach of modeling interorganizational business process for collaboration. Our approach allows for modeling loosely coupled interorganizational business process considering different views of organizations. In the proposed model, organizations have their own local process views of modeling business process instead of sharing pre-defined global processes. During process cooperation, local process of an organization can be invisible to other organizations. Further, we propose the coordination mechanisms for different local process views to detect incompatibilities among organizations. We illustrate our proposed approach by a case study of interorganizational software development collaboration.

key words: *interorganizational business process, workflow, process view, coordination, compatibility analysis*

1. Introduction

With the global expansion of the Internet and distributed computing environments, computer-mediated collaborative work has been rapidly increasing within individual organizations or between different organizations even across nations. Cooperative work can be modeled as an interorganizational business process [1]–[3]. Most of the existing specifications and approaches in interorganizational business process concentrate on process representation, verification and interaction protocols, with the assumption that organizations accept a common pre-defined model. However, there are some typical challenges with interorganizational business processes for real-life collaboration.

One challenge is that participants from different organizations have different cultural and knowledge backgrounds, which may produce a great impact on the way that the cooperative work is conducted [4]. In global software development, barriers of culture distance, process and management issues, organization, and other issues may result in conflicts for problem solving [5]. In previous investigation, the top three challenges of distributed software development are reported as effective communication, cultural differences, and coordination [6]. Those kinds of conflicts always cause many problems due to disagreements on views regarding process modeling. Another important aspect is privacy preservation of individual organizations during collaboration [3], [7]. Though cooperation of organizations needs some visible interactions and data sharing, it is also important for organizations to preserve their internal experiences and knowledge as much as possible. Moreover, flexibility support and process reuse have been attracting attention [8], [9]. To address the above issues, we deal with the following issues in this paper.

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

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

To validate the proposed interorganizational business process model and the coordination approach of local process views, we use a example of collaborative software development process as a case study.

2. Modeling Interorganizational Business Process for Collaboration Work

An interorganizational business process is always composed of a set of workflow processes, where *n* business partners are involved in one *global* process and each partner has its own

Manuscript received June 11, 2013.
Manuscript revised October 29, 2013.

†The authors are with the Department of Social Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.
a) E-mail: lindh@i.kyoto-u.ac.jp

DOI: 10.1587/transinf.E97.D.1119

local process. Each local process is controlled by its corresponding business partner. In the traditional loosely coupled interorganizational business process [1], although each organization has its own local process that can be executed independently, there is a common global view of interorganizational business process. Without considering the potential conflicts caused by different cultural backgrounds and understandings about the collaboration of different organizations, there is an assumption that all organizations accept pre-defined global process. However, this assumption is not reasonable because it is always difficult for all organizations to have the same opinions on creating a global process view at the beginning stage.

To address this issue, we propose an interorganizational business process model, where a single global view is not pre-defined. Instead, each organization has its own local process view based on its own consideration of the whole collaboration work. The local process view of each organization represents its unique consideration of the whole business process, including the information of its local process, interaction part between organizations, and virtual processes of its partners. Figure 1 outlines the architecture for the interorganizational business process model which supports multiple local process views. In this architecture, each organization has a local process view. Since there is no global view, they coordinate with each other by compatibility analysis, which is the analysis and detection of incompatibilities among different local process views, to detect conflicts and try to solve them.

In some important previous researches, interorganizational business processes are modeled as Petri nets based workflow net (WF-net) [10]. In this research, our focus is how to build a collaboration model of interorganizational business process rather than how to formalize the business process. To show the whole model and the coordination process more easily, we will describe the business process based on workflow graphs, which can also be easily transformed into WF-net [11].

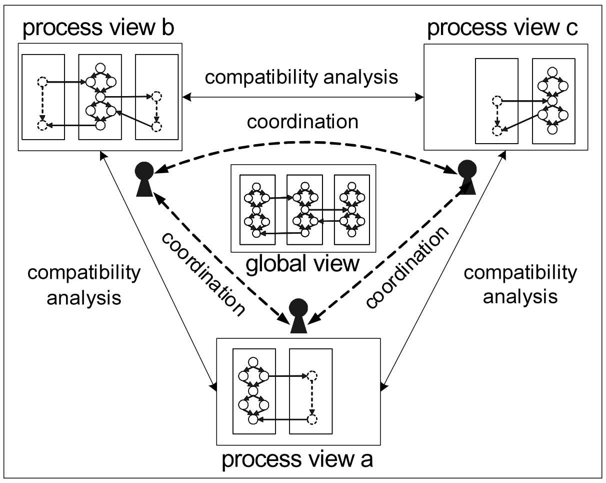

Fig. 1 Interorganizational business process with local process views.

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 $LPV = (I, WF_0, VWF_1, VWF_2, \ldots, VWF_n, ESC)$, where

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

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

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

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 (input and output if the task is a service) respectively
- c is the intra-connection type of the task such that $c \in C \rightarrow \{null, AND-split, AND-join, OR-split, OR-join, XOR-split, XOR-join\}$, where split and join should appear in pairs of $(AND-split, AND-join)$, $(OR-split, OR-join)$ and $(XOR-split, XOR-join)$ to ensure that the workflow process is sound [12]
- r is the inter-connection type of the task and $r \in R \rightarrow \{null, in, out\}$, where in and out represent the task is an input place interacted from other organizations and an output place interacting to other organizations respectively;

(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.

A virtual workflow $VWF = (i, VT, VF)$ has the similar form of workflow process in **Definition 2**, where VT is the finite set of all the task nodes, and VF is the finite set of flow relations in the virtual workflow process. Tasks in a virtual workflow always have the abstraction form of tasks, and the flow relations describe the flow of virtual tasks.

In this paper, we define the interaction protocols among the organizations by using Message Sequence Charts (MSC), which is a widespread graphical language to visualize communications between processes [13] and has been used in previous research [14].

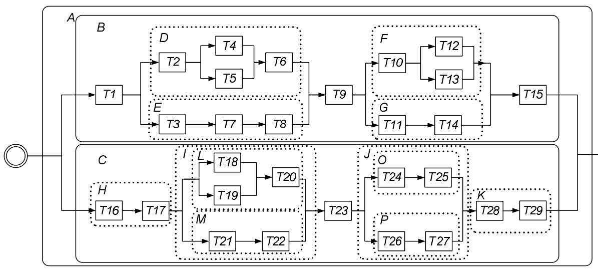

Fig. 2 Example of local process abstraction using subprocesses.

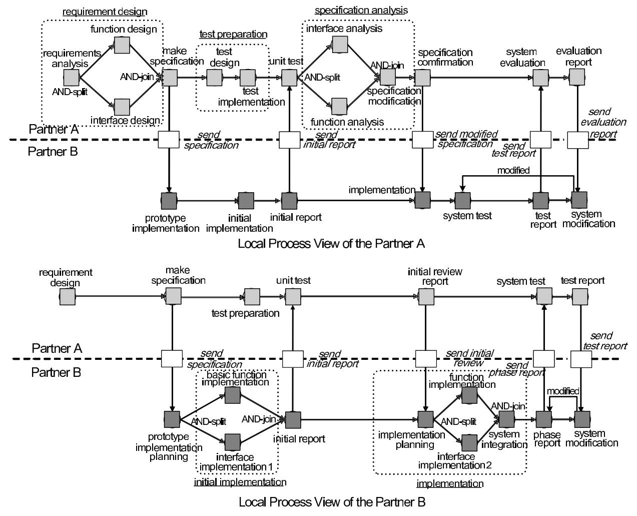

Fig. 3 Local process views of an interorganizational business process for software development collaboration. The light grey rectangles represent the tasks in the workflow process (top) and virtual workflow process (bottom) of Partner A. The deep grey rectangles represent the tasks in the workflow process (bottom) and virtual workflow process (top) of Partner B. The blank rectangles represent the messages (interactions events) in the message sequence charts.

Definition 3 (Message Sequence Charts) Message sequence charts for representing interactions among organizations is defined as a tuple $ESC = (I, E, from, to, \{\leq_i\}_{i\in I})$ where

(1) I is a finite set of organizations;

(2) E is a finite set of message (interaction events);

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

(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 a receiving message and $!e$ represents a sending message.

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 abstract task;

(2) TS denotes a finite set of concrete tasks that composes the subprocess;

(3) $prec$ and eff denote the precondition and the effect of the subprocess as a whole.

A subprocess has the structure of a workflow process with strongly-connected elements. A workflow process is still a complete workflow process after replacing a set of tasks with a subprocess. Figure 2 is an example of a workflow process with many subprocesses. In the example, $T_1 - T_{29}$ are atomic tasks, and $A - K$ are subprocesses that consist of several atomic tasks.

To explain the proposed workflow model based on local process views, we demonstrate a case of intercultural software development collaboration between groups from different organizations: *partner A* (outsourcer) and *partner B* (supplier). Similar examples are also used and discussed in our previous work [15]. In developing software, they have some rough agreement on the development assignment, e.g. *partner A* designs the software specifications and *partner B* develops the software based on the designed specifications. Considering the fact that two groups come from different organizations, it is quite possible that there are conflicts caused by different customs and some misunderstandings over software development, which leads to differences in local process views of the whole business process. Such problems have been reported to be major causes for failures in offshore software development across nations [16]. When creating the whole business process for the collaboration development, each partner wants to keep the detailed local process secret. Therefore, they model the business process by their own customs (Fig. 3).

In software development collaboration, the following conflicts might occur between the two organizations: (1) In *partner A*, software specifications are always modified in parallel with development. However, in *partner B*, specifications are almost totally designed before development. *Partner B* might not be able to understand why the subprocess *specification analysis* is required. Therefore, when they model their local process view, *specification analysis* is not considered; (2) In *partner A*, the software quality management is more strict. However, in *partner B*, software tests are conducted less frequently. In Fig. 3, *partner B* develops the software, but they think that *system test* should be done by *partner A*, which is totally different from the consideration of *partner A*.

3. Coordination of Local Process Views by Compatibility Analysis

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

(1) *Design of local process views*: each organization designs its local workflow process, interaction protocols and

virtual workflow processes of its partners based on their own requirement and consideration. Organizations make interaction protocols and virtual processes of its partners public, while keeping the local workflow processes as privacy;

(2) *Coordination of interaction protocols*: organizations compare interaction protocols (message sequence charts) designed by partners with their own protocols. If there are conflicts, they negotiate to reach an agreement. Then, they modify interaction protocols and the related parts of their local workflow processes in the local process views;

(3) *Coordination of workflow processes*: organizations compare the workflow with partners by (a) exchanging public virtual processes and compare local workflow process and its related virtual workflow process designed by its partners; (b) exchanging comparison results of workflow processes and negotiate to reach an agreement on the processes;

(4) *Modification of local process views*: organizations revise their local process views according to the negotiation results and save as new business process services for reuse in collaborative business.

We describe the coordination by focusing on compatibility analysis of local process views, which is divided into two phases, namely the interaction protocol coordination and workflow process coordination. Since this paper focuses on describing incompatibility types and compatibility analysis processes, it is sufficient to use two organizations for this purpose. However, since the definitions of local process view, workflow process, message sequence charts deal with multiple organizations, it is possible to extend the definition of incompatibility and compatibility analysis algorithms into the case of more than two organizations. Figure 4 shows the compatibility mechanisms for different local process views.

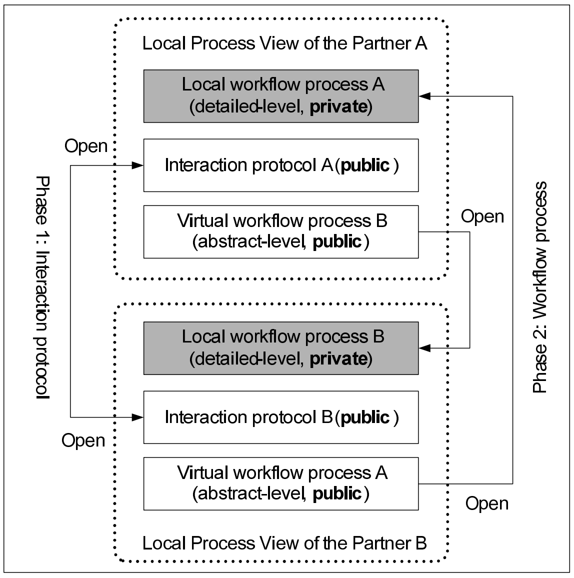
Local Process View of the Partner A
Local workflow process A (detailed-level, private)
Interaction protocol A (public)
Virtual workflow process B (abstract-level, public)
Local workflow process B (detailed-level, private)
Interaction protocol B (public)
Virtual workflow process A (abstract-level, public)
Local Process View of the Partner B
Open
Open
Open
Open
Phase 1: Interaction protocol
Phase 2: Workflow process

Fig. 4 Phases of compatibility analysis of local process views.

3.1 Phase 1: Interaction Protocol Coordination

When conducting compatibility analysis, interaction protocols should be first analyzed, which include two parts, i.e., the content and order of interaction events between organizations.

Definition 5 (Incompatibility of Interaction Protocols) If we define message (interaction events) sequence charts of two organizations A and B as

$$ESC_A = (I, E_A, from_A, to_A, \{\leq_i\}_{i\in I}),$$
$$ESC_B = (I, E_B, from_B, to_B, \{\leq_i\}_{i\in I})$$

respectively, $I = \{Org_A, Org_B\}$, then the following cases of incompatibility might occur in the interaction protocols.

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

$$(\exists e)((e \in E_A) \wedge (e \notin E_B)) \vee ((e \in E_B) \wedge (e \notin E_A));$$

(2) *Inconsistent Direction*: the sender of an interaction event in one local process view is the receiver of the other, notationally[†],

$$(\exists e)((from_A(e) = Org_A) \wedge (to_B(e) = Org_A))$$
$$\vee ((from_A(e) = Org_B) \wedge (to_B(e) = Org_B));$$

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

$$(\exists e, t_A, t_B)((des(t_A) = des(t_B)) \wedge (((t_A, e) \in F_A^I) \wedge ((t_B, e) \in F_B^I) \wedge (eff(t_A) \neq eff(t_B))) \vee (((e, t_A) \in F_A^I) \wedge ((e, t_B) \in F_B^I) \wedge (prec(t_A) \neq prec(t_B))));$$

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

$$(\exists e_i, e_j)((< e_i, e_j >\in \{\leq_i\}_A) \wedge (< e_j, e_i >\in \{\leq_i\}_B)).$$

The algorithm of compatibility analysis of interaction protocols is shown in **Algorithm 1**.

3.2 Phase 2: Workflow Process Coordination

In the first phase of compatibility analysis, organizations detect some conflicts about the interaction protocols among organizations. Such conflicts are expected to be eliminated through coordination and negotiation between involved organizations. After that, the compatibility analysis comes to the second phase, coordinating workflow processes designed by different organizations. For each organization i, it is necessary to compare its local workflow process WF_{0i} with the virtual workflow processes of i that are designed in

[†]We use the symbol = to denote equality (e.g., $from_A(e) = Org_A$) in Definition 5 and Definition 6.

Algorithm 1 Compatibility analysis of interaction protocols

```
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:              /* Incompatibility type: $Inexistent\ Event\ e$ */
7:          else if $to'(e) = Org_A$ then
8:              $IncompatibleFlag \leftarrow true$
9:              /* Incompatibility type: $Inconsistent\ Direction\ e$ */
10:         else if $((des(t) = des(t')) \wedge ((t, e) \in F_0^I) \wedge ((t', e) \in F_A^{I'}) \wedge$
            $(eff(t) \neq eff(t'))$ then
11:             $IncompatibleFlag \leftarrow true$
12:             /* Incompatibility type: $Inconsistent\ Condition\ t$ */
13:         end if
14:     end if
15:     Repeat line 3-14 by reversing the direction of $e$
16:     for all $\{e_i, e_j\} \subseteq E \cup E'$ do
17:         if $(< e_i, e_j > \in \{\leq_i\}_A) \wedge (< e_j, e_i > \in \{\leq_i\}_B)$ then
18:             $IncompatibleFlag \leftarrow true$
19:             /* Incompatibility type: $Disordered\ Sequence\ e_i, e_j$ */
20:         end if
21:     end for
22: end for
23: if $IncompatibleFlag = true$ then
24:     return Incompatible
25: else
26:     return Compatible
27: end if
```

the local process views of its partners.

Definition 6 (Incompatibility of Workflow Processes) If we define the local process views of two organizations A and B as

$$LPV_A = (I, WF_{0A}, VWF_B, ESC_A),$$
$$LPV_B = (I, WF_{0B}, VWF_A, ESC_B)$$

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

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

$$(\exists t)((t \in VT_A) \wedge (t \notin T_{0A})) \vee ((t \in VT_B) \wedge (t \notin T_{0B}));$$

(2) *Inconsistent Condition*: the preconditions or effects of the same task (e.g., the situation where a task exists in both WF_{0A} of LPV_A and VWF_A of LPV_B) are different in two local process views, i.e.,

$$(\exists t, tt)((((t \in VT_A) \wedge (tt \in T_{0A})) \vee ((t \in VT_B) \wedge$$
$$(tt \in T_{0B}))) \wedge (des(t) = des(tt)) \wedge$$
$$((prec(t) \neq prec(tt)) \vee (eff(t) \neq eff(tt))));$$

(3) *Disordered Tasks*: there exist two tasks that have opposite sequential orders in two local process views, notationally,

$$(\exists t_i, t_j)((t_i, t_j \in VT_A) \wedge (t_i, t_j \in T_{0A}) \wedge ((t_i, t_j) \in VF_A^N)$$
$$\wedge((t_j, t_i) \in F_{0A}^N)) \vee ((t_i, t_j \in VT_B) \wedge (t_i, t_j \in T_{0B}) \wedge$$
$$((t_i, t_j) \in VF_B^N) \wedge ((t_j, t_i) \in F_{0B}^N)).$$

If two local process views could avoid all the above incompatibilities, we call they are *compatible*. In the compatibility analysis process, we do not focus on the difficulties of defining terminology for names and elements of tasks and messages with different languages, cultural backgrounds and semantics from different organization. We can assume that business partners use some pre-defined ontology for terminology that would be used in designing the business process, or conduct coordination to solve the problem of conflicted semantics in the terminology.

There are following difficulties when analyzing the abstract-level virtual process and the detailed-level process of a same organization: (1) the structure of local workflow process modeled by the local organization is always far more complicated and (2) some parts in the abstract-level virtual processes cannot be directly mapped into the detailed-level local processes. For example, a task in the abstract-level virtual process might be equivalent to a group of tasks in the related detailed-level local process. To compare a virtual process and a detailed local process, the abstractions of tasks in the detailed local process are necessary. Therefore virtual tasks and subprocesses are added into the detailed-level local process for each organization when designing the local process view.

Take the process in Fig. 2 for instance, a local workflow process can be abstracted into different levels of abstract processes by using subprocess abstraction. Since workflow process is defined as a structured process graph, it can be reduced to a single node [17]. Here we can simplify the graph by blocks and subprocesses. A block is a unit of representation that minimally specifies the behavioral pattern of a process flow [18]. 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 complicated process with its abstract-level subprocesses into a block-structured hierarchical process tree so that the tree contains not only the atomic-level tasks but also abstract-level subprocesses. Process transformation has also been used in previous research for different purposes [18], [19]. Hierarchical tree-based workflow process makes it easy to identify the task nodes that need to be coordinated after compatibility analysis. If a block or a subprocess is detected to be incompatible with other organizations, only its child nodes require coordination.

For a workflow process $WF = (i, T, F)$, we have the following rules for the transformation of a process graph structure into the hierarchical structure, including iterative block transformation rule, sequential block transformation rule, branch block transformation rule and subprocess transformation rule. *Rule*1 is for handling the iterative block. In the process of transforming the process into hierarchical structure, the iterative arcs are removed for separate analy-

[†]Split and join are used in parallel block ($AND - split, AND - join$), conditional selective block ($OR - split, OR - join$) and non-conditional selective block ($XOR - split, XOR - join$). They can be categorized as branch block.

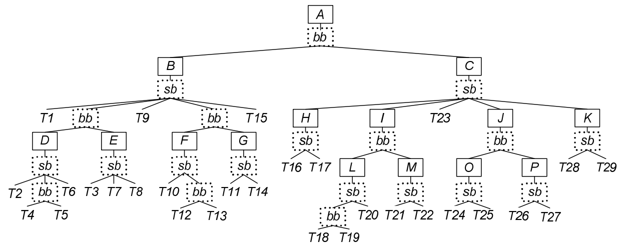

Fig. 5 Process structure after transformation (bb = branch block; sb = sequential block).

sis. *Rule*2 and *Rule*3 are for sequential block transformation and branch block transformation respectively. While these two types are detected in the workflow process, the involved tasks are replaced by the block, which is also regarded as a special task in the updated workflow process. In the workflow process tree, a new node for the detected block is created at the same time. All the tasks in the block then become the child node of the block in the tree. *Rule*4 is for transforming subprocesses. The operation is similar to *Rule*2 and *Rule*3.

(1) iterative block transformation

$\exists(t_1, t_2, \ldots, t_n) \in T$ where $ib = (t_1, t_2, \ldots, t_n)$

$Rule1 : ((i, T, F), ib) \rightarrow (i, T', F')$

$T' = T$

$F' = F - \{(t_n, t_1)\}$

(2) sequential block transformation

$\exists(t_1, t_2, \ldots, t_n) \in T$ where $sb = (t_1, t_2, \ldots, t_n)$

$Rule2 : ((i, T, F), sb) \rightarrow (i, T', F')$

$T' = T \cup \{sb\} - \{t_1, t_2, \ldots, t_n\}$

$F' = F \cup \{(t_{in}, sb)/(t_{in}, t_1) \in F\} \cup \{(sb, t_{out})/(t_n, t_{out}) \in F\} - (T \times \{t_1\}) \cup \{(t_1, t_2), \ldots, (t_{n-1}, t_n)\} \cup (\{t_n\} \times T)$

(3) branch block transformation

$\exists(t_1, t_2, \ldots, t_n) \in T$ where $bb = (t_1, t_2, \ldots, t_n)$

$Rule3 : ((i, T, F), bb) \rightarrow (i, T', F')$

$T' = T \cup \{bb\} - \{t_1, t_2, \ldots, t_n\}$

$F' = F \cup \{(t_{in}, bb)/(t_{in}, t_1) \in F\} \cup \{(bb, t_{out})/(t_1, t_{out}) \in F\} - (T \times \{t_1, t_2, \ldots, t_n\}) \cup (\{t_1, t_2, \ldots, t_n\} \times T)$

(4) subprocess transformation

$\exists(t_1, t_2, \ldots, t_n) \in T$ where $sp = (t_1, t_2, \ldots, t_n)$

$Rule4 : ((i, T, F), sp) \rightarrow (i, T', F')$

$T' = T \cup \{sp\} - \{t_1, t_2, \ldots, t_n\}$

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

The workflow process in Fig. 2 can be transformed into a process tree as shown in Fig. 5 based on the transformation rules. After the workflow process is transformed into a hierarchical process, we can conduct the compatibility analysis of workflow processes (**Algorithm 2**). Therefore, here we have two steps: (1) Search and map the tasks of abstract-level virtual process in the transformed process tree, i.e., for each task $t \in VT$, check whether there exists an element in the process tree so that t is equivalent to tt by $prec(t) = prec(tt)$ and $eff(t) = eff(tt)$; (2) Compare the order of tasks in the abstract-level virtual process and the related task order in the process tree. Then, we can detect cases of incompatibility described in **Definition 6**.

Algorithm 2 Compatibility analysis of workflow processes

```
procedure Compatibility-Analysis-2($LPV, LPV'$)
 1: $IncompatibleFlag \leftarrow false$
 2: $TWF_0 \leftarrow$ Transformed Process from $WF_0$
 3: for all $t \in VT'$ do
 4:     /*$VT'$ is the virtual task set of $VWF'$ in $LPV'$.*/
 5:     $map[t] \leftarrow null$
 6:     for all $tt \in TT_0$ do
 7:         /*$TT_0$ is the task set of $TWF_0$ in $LPV$.*/
 8:         if $des(t) = des(tt)$ then
 9:             if $(prec(t)=prec(tt)) \wedge (eff(t)=eff(tt))$ then
10:                 $map[t] \leftarrow tt$
11:             else
12:                 $IncompatibleFlag \leftarrow true$
13:                 /* Incompatibility type: $Inconsistent\ Condition\ t$ */
14:             end if
15:             break
16:         end if
17:     end for
18:     if $map[t] = null$ then
19:         $IncompatibleFlag \leftarrow true$
20:         /* Incompatibility type: $Inexistent\ Task\ t$ */
21:     end if
22: end for
23: for all $(t_i, t_j) \in VF'^N$ do
24:     /*$VF'^N$ is the virtual flow set of $VWF'$ in $LPV'$.*/
25:     if $(map[t_j], map[t_i]) \in TF_0^N$ then
26:         /*$TF_0^N$ is the internal flow set of $TWF_0$ in $LPV$.*/
27:         $IncompatibleFlag \leftarrow true$
28:         /* Incompatibility type: $Disordered\ Tasks\ t_i, t_j$ */
29:     end if
30: end for
31: if $IncompatibleFlag = true$ then
32:     return Incompatible
33: else
34:     return Compatible
35: end if
```

4. Case Study

We use a case of collaborative software development process (the example in Fig. 3) to illustrate our proposed approach. After modeling the local process views of the two organizations, compatibility analysis should be conducted for coordination.

First, interaction protocols between two partners are compared. In the example shown in Fig. 3, *Partner A*'s local process view includes following five interaction events (message): (1) $send\ specification_{A \rightarrow B}$; (2) $send\ initial\ report_{B \rightarrow A}$; (3) $send\ modified\ specification_{A \rightarrow B}$; (4) $send\ test\ report_{B \rightarrow A}$; and (5) $send\ evaluation\ report_{A \rightarrow B}$. However, the interaction events included in *partner B*'s local process view are a little different. By using the compatibility analysis for interaction protocol coordination described in **Algorithm 1**, the incompatibilities among interaction protocols can be detected. Then, two partners try to negotiate with each other to reach an agreement on how to modify the interaction protocols for mutual compatibility until there is no incompatibility between the interaction.

Second, within each partner, the local process and its

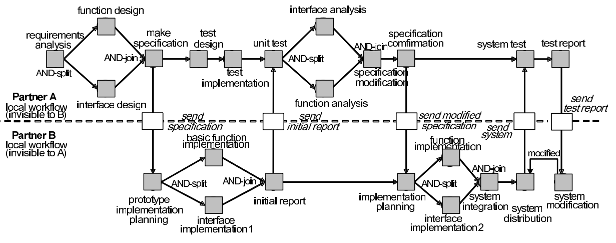

Fig. 6 Local processes of an interorganizational business process in Fig. 3 after coordination.

virtual process designed by the other partner are compared. To compare these processes, we first transform the local detailed process into a block-structured hierarchical process using the transformation rules. The hierarchical process is transformed from the local process and combined with subprocesses. For example, in *partner A*'s local process view, the hierarchical process combines three subprocesses: *requirement design*, *test preparation* and *specification analysis*. The parameters of precondition and effect for these atomic tasks and subprocesses are defined before the coordination process. Then, the virtual process in *partner B*'s local process view and the block-structured hierarchical process, which is transformed from the local workflow process in *partner A*'s local process view, could be compared. To compare these processes, we search and map the virtual tasks of the virtual process in the block-structured hierarchical process. Precondition and effect of tasks are compared during this process. By this means, incompatibilities of business processes designed by two partners can be detected using **Algorithm 2**. For example, there is a conflict that two partners have different opinions about who should be responsible for the task *system test* in the two initial local process views. Figure 6 shows the interaction protocols and local workflow processes of two partners after coordination and negotiation.

5. Related Work

Many interorganizational business process approaches have been proposed, including the perspective of view [3], workflow interoperability [1], [7], agreement and contracts between organizations [2], process specification languages [20] and so on. In this section, we compare our work with previous work on different features, which is summarized in Table 1.

Privacy preservation (PP). In modeling interorganizational business process, most of approaches consider privacy preservation, where local workflow processes of organizations are not entirely open to public. In some approaches, organizations provide part of its workflow as the public part by using workflow abstraction [2], [3]. In other approaches, organizations design local processes privately based on a common accepted abstract interorganizational workflow [7]. In our approach, organizations do not open its local workflows, instead they design virtual workflows for other organizations based on their own understandings.

Table 1 Comparison of our approach and related work.

Approach	Comparison Items					
	PP	FS	PR	VA	CE	MV
Chiu *et al.* [20]			√	√		
Grefen *et al.* [2]	√	√				
van der Aalst [7]	√	√		√	√	
Chebbi *et al.* [3]	√	√	√			
Our approach	√	√	√		√	√

Flexibility support (FS). Flexibility in interorganizational business process managements covers many aspects, e.g., local workflow design, dynamic workflow change, exception handling, workflow system usage and so on. For example, some previous work focuses on the flexibility of runtime workflow change [2]. In our approach, we mainly concentrate of flexibility of local workflow design of organizations, which is similar to some previous approaches [3], [7]. Moreover, since each organization has its own local process view in our approach, it is flexible for organizations to design the whole interorganizational workflow view based on their understanding.

Process reuse (PR). In the context of collaborative business, it is efficient and economical for business partners to find some existing processes from business process repositories for coordination. Some approaches enable organizations to preserve their local workflows as services [3], [20]. However, local workflows need cooperation and composition mechanisms to establish an interorganizational workflow. In our approach, each organization has a local process view including not only local workflows but also interaction protocols that can be preserved as a workflow service.

Verification analysis (VA). Some previous researches use specification languages or mathematical models to represent workflow processes and therefore provide verification functions [7], [20]. Currently, the issue of verification analysis is not considered in our approach, but it is important to address this issue by extending our current interorganizational workflow model.

Coordination efforts (CE). Chebbi *et al.* [3] propose the idea that organizations coordinate their private workflow through third party. However it is not clear what is required for organizations to conduct for coordination. In the approach of van der Aalst [7], organizations try to coordinate to get a common part for interorganizational workflow, and design their own private local workflow processes based on the common part. However, organizations might have conflict with each other if there is no coordination between local workflows. In our approach, the coordination efforts involve both the interaction part and the local workflows.

Multiple views (MV). In this paper, we not only present the important issue of interaction protocols between organizations, but also consider the different process views of organizations, which is rarely discussed in previous work.

6. Conclusion

In this paper, we propose an interorganizational business process approach to model collaborative business based on their own considerations of the whole process, and coordinate their unique views with partners to reach mutual understanding and compatibility. The contribution of this research covers two issues. First, we propose an interorganizational business process collaboration model, taking different process modeling views of different organizations into consideration. In this model based on local process views, organizations do not share a pre-defined global interorganizational business process, but have their own understandings of how to model the whole business process with their unique backgrounds and knowledge. Second, we propose the coordination mechanism for different local process views to detect conflicts among organizations, which is realized by conducting compatibility analysis of local process views of different organizations. The compatibility analysis is divided into two main phases: coordination of interaction protocols, and coordination of workflow processes. During this process, incompatibilities are expected to be detected and eliminated before business process execution, and therefore a compatible interorganizational business process that supports multiple local process views can be established. The proposed interorganizational business process model and coordination approach is demonstrated by a case study of collaborative software development collaboration.

Acknowledgements

This research was partially supported by Service Science, Solutions and Foundation Integrated Research Program from JST RISTEX, and a Grant-in-Aid for Scientific Research (S) (24220002) from JSPS.

References

[1] W.M.P. van der Aalst, "Loosely coupled interorganizational workflows: Modeling and analyzing workflows crossing organizational boundaries," Information & Management, vol.37, no.2, pp.67–75, 2000.

[2] 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, vol.24, no.1, pp.52–57, 2001.

[3] I. Chebbi, S. Dustdar, and S. Tata, "The view-based approach to dynamic inter-organizational workflow cooperation," Data Knowl. Eng., vol.56, no.2, pp.139–173, 2006.

[4] G. Hofstede, Culture's Consequences: Comparing Values, Behaviors, Institutions, and Organizations across Nations, Sage Pubns, 2001.

[5] J. Noll, S. Beecham, and I. Richardson, "Global software development and collaboration: Barriers and solutions," ACM Inroads, vol.1, no.3, pp.66–78, 2010.

[6] F.Q. da Silva, C. Costa, A.C.C. França, and R. Prikladinicki, "Challenges and solutions in distributed software development project management: A systematic literature review," 5th IEEE International Conference on Global Software Engineering (ICGSE2010), pp.87–96, 2010.

[7] W.M.P. van der Aalst, "Inheritance of interorganizational workflows to enable business-to-business e-commerce," Electronic Commerce Research, vol.2, no.3, pp.195–231, 2002.

[8] A. Dogac, Workflow Management Systems and Interoperability, Springer, 1998.

[9] W.M.P. van der Aalst, T. Basten, H. Verbeek, P. Verkoulen, and M. Voorhoeve, Adaptive Workflow: On the Interplay between Flexibility and Support, Enterprise Information Systems, 2000.

[10] W.M.P. van der Aalst and K. van Hee, Workflow Management: Models, Methods, and Systems, MIT Press, 2002.

[11] W.M.P. van der Aalst, A. Hirnschall, and H. Verbeek, "An alternative way to analyze workflow graphs," Proc. 14th International Conference on Advanced Information Systems Engineering (CAiSE2002), vol.2348, pp.535–552, 2002.

[12] J. Vanhatalo, H. Volzer, and F. Leymann, "Faster and more focused control-flow analysis for business process models through SESE decomposition," 5th ICSOC Conference, LNCS, pp.43–55, 2007.

[13] S. Mauw and M.A. Reniers, "An algebraic semantics of basic message sequence charts," Computer Journal, vol.37, no.4, pp.269–277, 1994.

[14] W.M.P. van der Aalst, "Process-oriented architectures for electronic commerce and interorganizational workflow," Inf. Syst., vol.24, no.9, pp.639–671, 1999.

[15] D. Lin, H. Sheng, and T. Ishida, "Interorganizational workflow execution based on process agents and ECA rules," IEICE Trans. Inf. & Syst., vol.E90-D, no.9, pp.1335–1342, Sept. 2007.

[16] S. Krishna, S. Sahay, and G. Walsham, "Managing cross-cultural issues in global software outsourcing," Commun. ACM, vol.47, no.4, pp.62–66, 2004.

[17] J. Mendling, K. Lassen, and U. Zund, "Transformation strategies between block-oriented and graph-oriented process modelling languages," Tech. Rep. JM-200510-10, WU Vienna, Oct. 2005.

[18] J. Bae, H. Bae, S. Kang, and Y. Kim, "Automatic control of workflow processes using ECA rules," IEEE Trans. Knowl. Data Eng., pp.1010–1023, 2004.

[19] W. Sadiq and M. Orlowska, "Analyzing process models using graph reduction techniques," Inf. Syst., vol.25, no.2, pp.117–134, 2000.

[20] D. Chiu, S. Cheung, S. Till, K. Karlapalem, Q. Li, and E. Kafeza, "Workflow view driven cross-organizational interoperability in a Web service environment," Information Technology and Management, vol.5, no.3, pp.221–250, 2004.

Donghui Lin was born in 1982. He received his Ph.D. degree in social informatics at Kyoto University in 2008. He is an assistant professor of social informatics at Kyoto University. His research interests include services computing, business process management and intercultural collaboration.

Toru Ishida was born in 1953. He is a professor of social informatics at Kyoto University. He has been working on Digital City and Language Grid projects. His research interests include autonomous agent and multiagent systems, semantic Web service and intercultural collaboration. He is a fellow of IEICE, IPSJ, and IEEE.