

Zhao, X., Liu, C., Yang, Y., & Sadiq, W. (2007). Handling instance correspondence in inter-organisational workflows.

Originally published in *Proceedings of the 19th International Conference on Advanced Information Systems Engineering (CAiSE 2007), Trondheim, Norway, 11–15 June 2007.*

Lecture notes in computer science (Vol. 4495, pp. 51–65). Berlin: Springer.

Available from: http://dx.doi.org/10.1007/978-3-540-72988-4_5

Copyright © Springer-Verlag Berlin Heidelberg 2007.

This is the author's version of the work, posted here with the permission of the publisher for your personal use. No further distribution is permitted. You may also be able to access the published version from your library. The definitive version is available at <http://www.springerlink.com/>.

Handling Instance Correspondence in Inter-Organisational Workflows

Xiaohui Zhao¹, Chengfei Liu¹, Yun Yang¹, and Wasim Sadiq²

¹ Faculty of Information and Communication Technologies
Swinburne University of Technology
Melbourne, Victoria, Australia
{xzhao, cliu, yyang}@ict.swin.edu.au

² SAP Research
Brisbane, Australia
wasim.sadiq@sap.com

Abstract. As business collaboration involves multiple business processes from different participating organisations, it becomes a challenging issue to manage the complex correspondence between instances of these business processes. Yet very limited support has been provided by inter-organisational workflow research. In this paper, we develop a formal method to specify instance correspondence based on a novel correspondence Petri net model. In this method, cardinality parameters are defined to represent cardinality relationships between collaborating business processes at build time, while correlation structures are designed to characterise correspondence between collaborating business process instances at run time. Corresponding algorithms are also developed to generate the correspondence Petri nets for collaborative business processes, and to trace instance correlation on the fly using the generated Petri nets.

Key words. Inter-organisational workflow management, workflow instance correspondence, correspondence Petri net

1 Introduction

In recent years, organisations have been undergoing a thorough transformation towards highly flexible and agile collaborations [1]. Organisations are required to dynamically create and manage collaborative business processes to grasp market opportunities [2]. A collaborative business process involves multiple parties and their business processes, thus it inevitably brings new challenges to workflow choreography and orchestration. One of the most pressing issues in this context is the instance correspondence.

Complex instance correspondences may exist at both build time and run time. Here, we characterise instance correspondences in terms of cardinality and correlations. Thereby, we can define and represent static and dynamic correspondence when modelling and executing a collaborative business process.

Some research efforts were put in this field. Multiple workflow instantiation was discussed by Dumas and ter Hofstede [3], using UML activity diagrams. Later they extended their work to service interactions [4]. van der Aalst et al. [5] deployed coloured Petri nets to represent multiple workflow cases in workflow patterns, and implemented it in the YAWL system [6]. Guabtni and Charoy [7] extended the multiple instantiation patterns and classified multiple workflow instantiation into parallel and iterative instances. However, most of the above research focus on interaction patterns, and sidestep the instance correspondence issue in collaborative business processes. WS-BPEL (previously BPEL4WS) [8] uses its own correlation set to combine workflow instances, which have same values on specified message fields. However, WS-BPEL defines a business process in terms of a pivot organisation. This results in that a WS-BPEL business process only represents the interaction behaviours of the pivot organisation with its neighbouring organisations. This feature limits its application for complex business collaborations, which are likely to include interactions beyond neighbouring organisations.

Aiming to address this issue, this paper proposes a method to support instance correspondences from an organisation-oriented view. In our method, cardinality parameters are developed to characterise cardinality relationships between collaborating business processes at build time. Besides, a correlation structure is combined with each instance to trace dynamic workflow correlations at run time. In addition, we formalise this method with a novel correspondence Petri net to describe instance correspondence precisely.

The rest of this paper is organised as follows: Section 2 analyses the instance correspondence within collaborative business processes with a motivating example. In Section 3, we discuss workflow cardinality and correlation issues in business collaboration context. In Section 4, we establish a novel correspondence Petri net model with extensions on workflow cardinality and correlation as our formal method. In Section 5, we develop algorithms to illustrate how to model collaborative business processes and manage run time executions of business collaborations with these special Petri nets. Section 6 concludes this paper and indicates our future work.

2 Motivating Example

Figure 1 illustrates a business collaboration scenario, where a retailer may initiate a product-ordering process instance that orders products from a manufacturer. The manufacturer may use a production process instance to receive orders from retailers. Once it obtains enough orders, the production instance may start making goods in bulk. At the same time, the manufacturer may assign several shippers to handle goods delivery. These shippers arrange their goods transfer according to their transfer capability and route optimisation etc. Finally, these shipping instances send goods to the proper retailers according to these correlations.

From this collaboration scenario, we see that an instance of one business process is likely to interact with multiple instances of another business process. For example, one production instance may correspond to multiple product-ordering instances, and multiple shipping instances may correspond to multiple product-ordering instances. In contrast to such complex quantitative relationship, most current workflow modelling

approaches simply assume and support a one-to-one relationship between business process instances. Although such requirements are quite common in B2B collaborations, they are primarily supported by enterprise applications internally and not adequately by workflow management systems.

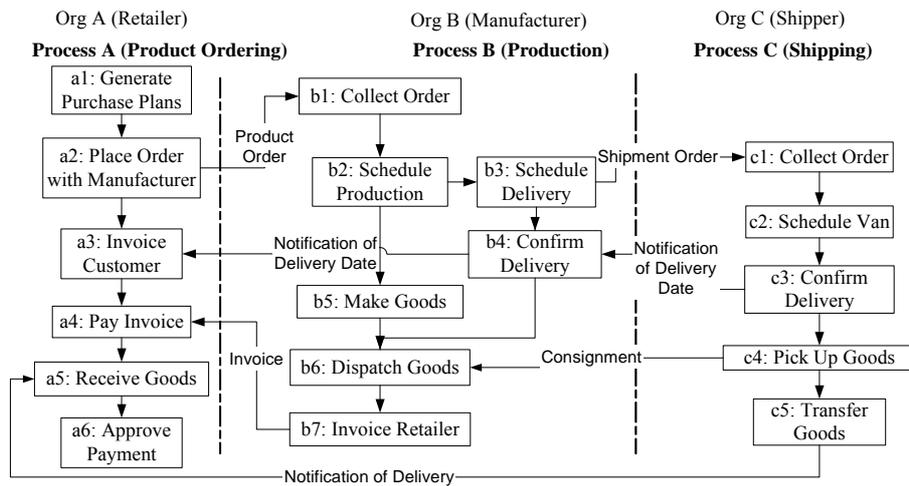


Fig. 1. Motivating example

At run time, instance correspondences are subject to the correlations between instances of different business processes. These correlations result from the underlying business semantics of interactions. In real cases, such correlations may be realised by real interactions (direct) or passing unique identifiers (indirect), such as order number. Sometimes, real interactions between instances may be triggered by time duration, external events etc. In this example, the manufacturer’s production instance is correlated with retailers’ product-ordering instances during the real interaction of receiving orders from retailers. Afterwards, the manufacturer contacts shippers for booking deliveries. At the same time, the manufacturer also passes the order numbers to proper shippers. With these order numbers, shippers’ shipping instances are indirectly correlated with retailers’ product-ordering instances. Following these correlations, shippers can pick up produced goods from the manufacturer, and then transfer them to proper retailers.

From the above discussion, we see that workflow correlations combine business interactions into a meaningful collaboration. Some existing approaches provide primitive support for correlation handling, such as message correlations in WS-BPEL. As discussed in Section 1, a WS-BPEL business process generated for a retailer cannot cover the interactions between the manufacturer and shippers, not to mention the correlations between their production and shipping instances.

3 Cardinality and Correlation Issues in Business Collaboration

In a collaborative business process, each participating organisation may play a specific role and only care about its own interests. For this reason, participating organisations do not wish, and may not be allowed to know the details of their partner organisations. Therefore, each participating organisation only has a partial and restricted view of the whole collaboration [9-12]. Due to diverse partnerships and authorities, different organisations may view the same collaboration differently.

3.1 Workflow Cardinality

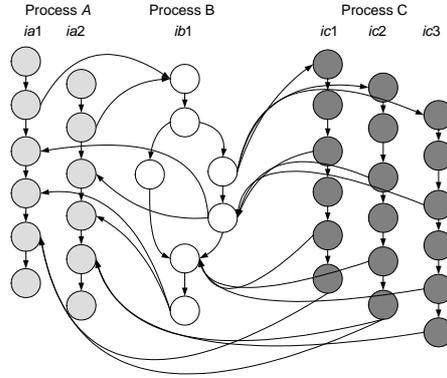


Fig. 2. Workflow cardinality of motivating example

Figure 2 shows a possible instance correspondence situation of the collaborative business process in the motivating example. In general, there are four possible cardinality relationships between a pair of interacting business processes, viz., single-to-single, single-to-many, many-to-single and many-to-many. In the organisation-oriented view, we substitute the four bilateral cardinality relationships with the pair of unidirectional cardinality relationships. For example, a single-to-many relationship between business processes p_B and p_C can be represented by a “to-many” relationship from p_B to p_C and a “to-one” relationship from p_C to p_B . A many-to-many relationship between p_A and p_C can be represented by a “to-many” relationship from p_A to p_C and a “to-many” relationship from p_C to p_A . In this paper, we define these two cardinality relationships with two *workflow cardinality parameters*,

- $[:1]$, denotes a *to-one* cardinality relationship;
- $[:n]$, denotes a *to-many* cardinality relationship.

As process interactions are implemented in the form of messaging behaviours, we incorporate these two cardinality parameters to message modelling. Conceptually, a message type can be defined as follows:

Definition message type. A message type m is defined as a tuple $(\rho, \alpha, \beta, f, \chi)$, where

- ρ is m 's messaging direction, 'in' or 'out'. These two values denote that m stands for an incoming message or an outgoing message, respectively.
- α is a task of a business process. α represents m 's source task, if m is an outgoing message; or it represents m 's target task.
- β is a set of tasks. This set of tasks represents m 's possible source tasks, if m stands for an incoming message; or it represents m 's possible target tasks. Each task in β is likely to send or receive an instance of m according to m 's direction.
- $f : \beta \rightarrow \{ [:1], [:n] \}$ is a mapping from β to the two discussed cardinality parameters.
- χ denotes the specification of the message body.

Here, α and β together represent the cardinality between business processes at type level. Two message types are said to be a pair if they have complementary source / target tasks and the same message body specification. The details of linking internal business processes into a collaborative business process via message types will be discussed in Section 5.

3.2 Workflow Correlation

Workflow correlation denotes the semantic relation between business process instances in the same business collaboration. Instances are directly correlated, when they “shake hands” during interactions. In addition, some instances may inherit pre-existing correlations from their counterparts during interactions. This correlation inheritance reflects the extending of business semantic relations.

In the scenario shown in Figure 2, firstly instances ia_1 and ia_2 are correlated with instance ib_1 , when ib_1 accepts orders from ia_1 and ia_2 ; Secondly, ib_1 contacts instances ic_1 , ic_2 and ic_3 for delivery booking. Here, suppose ib_1 assigns ic_1 and ic_2 to transfer products for ia_1 , and assigns ic_2 and ic_3 to transfer products for ia_2 . Thereby instances ic_1 , ic_2 and ic_3 are directly correlated with ib_1 , and they also inherit previous correlations from ib_1 . In this example, ic_1 and ic_2 inherit the correlation between ia_1 and ib_1 from ib_1 , while instances ic_2 and ic_3 inherit the correlation between ia_2 and ib_1 from ib_1 . This inheritance implies that shippers require consignees' information to arrange their shipping schedules. Corresponding shipping instances are therefore indirectly correlated with retailers' product-ordering instances. This inheritance is realised by passing retailers' order numbers from the manufacturer to shippers.

Based on these workflow correlations, we can derive a *logical instance* of a participating business process instance in the organisation-oriented view. From a business process instance ζ of organisation g , a so-called logical instance ξ consists of ζ and all its related instances of business processes belonging to other organisations through the instance correlations at run time. Here, organisation g is called *host organisation* of ξ , and ζ is called *base business process instance* of ξ . In terms of workflow correlations, we can define a logical instance as follows,

Definition *logical instance*. In the context of a collaborative business process A , the logical instance for a base business process instance ζ is defined as tuple (ζ, A, Δ) , where Δ is the set of business process instances that are correlated with ζ in the context of A .

The set of correlated business process instances evolves during the business collaboration. For example, if we start from instance ia_1 , the set of correlated business process instances for ia_1 contains no instances at the beginning; while it includes instance ib_1 right after ib_1 accepts its order, i.e., $\mathcal{A} = \{ ib_1 \}$; afterwards instances ic_1 and ic_2 may be added after ib_1 books delivery with ic_1 and ic_2 , then $\mathcal{A} = \{ ib_1, ic_1, ic_2 \}$.

4 Correspondence Representation Methodology

Petri nets were invented by Carl Petri in the sixties for modelling concurrent behaviours of a distributed system. A Petri net is a bipartite graph whose nodes can be distinguished in places and transitions, which are graphically represented by circles and rectangles, respectively. A Predicate / Transition or coloured Petri net can differentiate tokens with unique identifications or a set of colours. Each place can contain tokens of different identifications or colours at the same time. Each arc may be assigned with an expression to restrict what tokens and the number of tokens that can transfer through. Therefore, a Petri net can represent multiple process executions within one net. Now, Petri nets are widely applied in concurrency control and process simulation [13].

4.1 Correspondence Petri Nets

To support workflow cardinality and correlation, we extend traditional Petri nets with new parameters and functions together with special places and transitions.

1. Cardinality parameters

An auxiliary place is used to denote a message between two business processes, which may be represented by two sub nets. In Figure 3 (a), auxiliary place p is drawn as a shaded circle, while sub nets A and B are differentiated by white and striped circles.

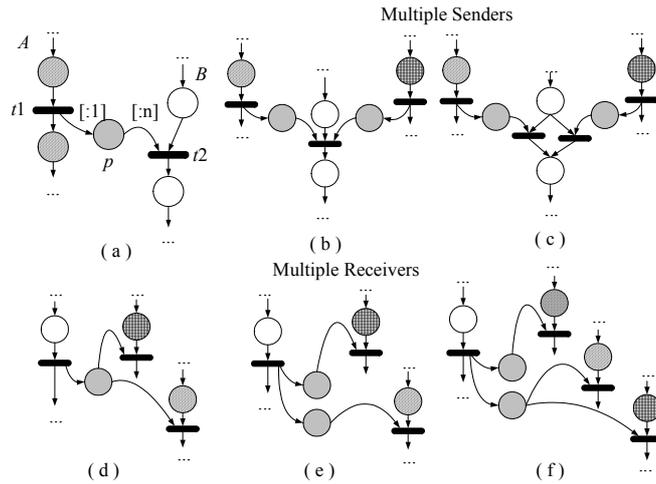


Fig. 3. Cardinality parameters

Transition t_1 of A is an *interaction requesting transition*, while transition t_2 of B is an *interaction responding transition*. Unidirectional cardinality parameter “[:1]” on the arc linking t_1 to p denotes that A views this interaction as a “to-one” cardinality, i.e., each token in A interacts with one token in B from A ’s view. Parameter “[:n]” on the arc linking p to t_2 denotes a “to-many” cardinality, i.e., each token in B corresponds multiple tokens in A from B ’s view. Therefore, we see that an auxiliary place separates the cardinality views from different perspectives.

2. Multiple message senders / receivers

Particular structures are used to represent the scenarios where multiple possible senders or receivers are instances of different business processes. In regard to multiple senders, Figure 3 (b) shows an interaction receiving messages from two senders; while Figure 3 (c) shows an interaction receiving one message from two senders. In regard to multiple receivers, Figure 3 (d) shows an interaction in which one of two receivers is expected to receive the message; while Figure 3 (e) shows an interaction that a message is sent to both receivers. With these basic interaction schemes, we can represent more complicated ones. For example, Figure 3 (f) shows a scenario that a task sends a message to three receivers, and one of the three will receive it definitely, while only one of the other two is expected to do so.

3. Special transitions

In some cases, an interaction may result in generating new instances. For example, in the book-delivery interaction between a manufacturer and a shipper, the shipper may generate several new shipping instances to handle it. In Petri net context, this requires the corresponding transition to be capable of generating new tokens. In this paper, we classify such transitions as *token-generating transitions*. In this way, we represent the book-delivery interaction as the Petri net segment shown in Figure 4.

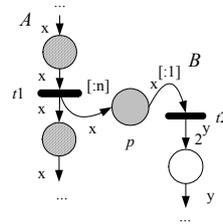


Fig. 4. Correlation function attached structures

In Figure 4, variable x or y ($x \neq y$) is labelled along an arc to denote the type of tokens that may go through this arc. For example, the token that flows from transition t_1 to place p is different from the token that flows out of transition t_2 . In addition, expression 2^y is labelled along the arc linking t_2 to the adjacent place, as t_2 is a token-generating transition. Thereby, this arc allows that more than one token representing instances of the same business process to pass through at one time.

4. Correlation structures

To record the run time workflow correlation, we combine a *correlation structure* with each token. A correlation structure is defined as follows:

Definition correlation structure. In a Petri net, the correlation structure for token ζ is defined as $r^\zeta = \{ \zeta, D_1, D_2, \dots, D_n, \mathcal{R} \}$, where

- each D_i ($1 \leq i \leq n$) denotes a set of tokens, which represent correlated instances of a business process. All tokens in D_1, D_2, \dots, D_n are correlated with ζ .
- \mathcal{R} is a binary relation defined between tokens in $\bigcup_{i=1}^n D_i$. Here, $d_x \mathcal{R} d_y$, ($d_x, d_y \in \bigcup_i D_i$), denotes that tokens d_x and d_y are correlated via token ζ .

ζ is called *base token* of this correlation structure. Token sets D_1, D_2, \dots, D_i may be dynamically updated during collaboration. For example, Figure 5 shows a part of the collaboration scenario mentioned in the motivating example using a Petri net. Each sub net stands for a business process, and is distinguished with different circles. The tiny circles within places denote tokens, and each token such as ia_1, ib_1, ic_1 stands for a business process instance. Each transition such as ta_2, tb_1, tc_1 stands for a task. When ia_1 and ia_2 flow to transition tb_1 via auxiliary place ap_1 , it means that the production instance accepts the orders from two retailers. Therefore, correlation structure r^{ib_1} at this moment is $\{ ib_1, \{ ia_1, ia_2 \}, \emptyset \}$. Tokens ia_1 and ia_2 may have correlation structures $r^{ia_1} = \{ ia_1, \{ ib_1 \}, \emptyset \}$ and $r^{ia_2} = \{ ia_2, \{ ib_1 \}, \emptyset \}$, respectively.

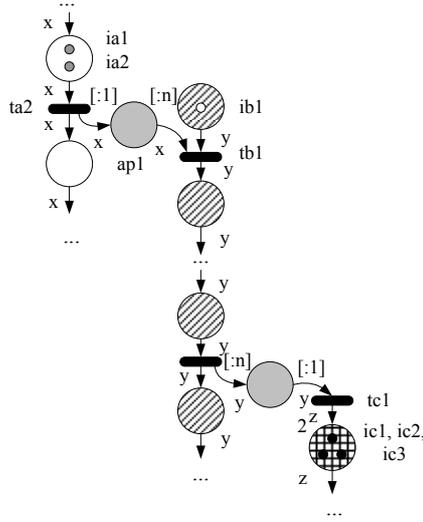


Fig. 5. Correlation scenario

This correlation structure accordingly evolves as the base token flows and interacts with other tokens. When ib_1 contacts ic_1, ic_2 and ic_3 to arrange the goods delivery for ia_1 and ia_2 , we suppose that ib_1 assigns ic_1 and ic_2 to serve ia_1 , while assigns ic_2 and ic_3 to serve ia_2 . Thus, r^{ib_1} will change to $\{ ib_1, \{ ia_1, ia_2 \}, \{ ic_1, ic_2, ic_3 \}, \{ (ia_1, ic_1), (ia_1, ic_2), (ia_2, ic_2), (ia_2, ic_3) \} \}$. Here, the last set denotes the correlated tokens via ib_1 . As the consignee information, the order numbers from ia_1 and ia_2 are passed to ic_1 and ic_2, ic_2 and ic_3 by ib_1 , respectively. Therefore, r^{ic_1} is set as $\{ ic_1, \{ ib_1 \}, \{ ia_1 \},$

\emptyset }, r^{ic2} is set as $\{ ic_2, \{ ib_1 \}, \{ ia_1, ia_2 \}, \emptyset \}$ and r^{ic3} is set as $\{ ic_3, \{ ib_1 \}, \{ ia_2 \}, \emptyset \}$.

4.2 Correspondence Petri Net

According to the above discussion, we establish a novel Petri net, called correspondence Petri net (CorPN), by extending the traditional Place / Transition Petri net. The definition of this CorPN is given below.

Definition *Correspondence Petri net.* A CorPN is represented as tuple $\Sigma = (\mathcal{P}, \mathcal{T}, \mathcal{F}, \mathcal{P}^\circ, \mathcal{F}^\circ, \mathcal{D}, \mathcal{V}, \mathcal{G}, \mathcal{E}, \mathcal{C}, \mathcal{Q}, \mathcal{I})$, where

(i) $(\mathcal{P}, \mathcal{T}, \mathcal{F})$ is a directed net, called the base net of Σ . Here, \mathcal{P} , \mathcal{T} and \mathcal{F} stand for the sets of places, transitions and arcs, respectively. $\mathcal{P} \cap \mathcal{T} = \emptyset$; $\mathcal{P} \cup \mathcal{T} \neq \emptyset$; $\mathcal{F} \subseteq \mathcal{P} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P}$;

(ii) $\mathcal{P}^\circ \subset \mathcal{P}$, is the set of auxiliary places, which represent the messaging relations between business processes of a collaborative business process.

(iii) $\mathcal{F}^\circ \subset \mathcal{F}$, is the set of arcs that connect auxiliary places, i.e., $\mathcal{F}^\circ \subseteq \mathcal{P}^\circ \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P}^\circ$.

(iv) \mathcal{D} is a set of tokens, each of which stands for a possible participating business process instance. Here, $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \dots \cup \mathcal{D}_n$, $\mathcal{D}_i \cap \mathcal{D}_j = \emptyset$, where $1 \leq i, j \leq n$ and $i \neq j$. Precisely, each \mathcal{D}_i denotes a *token group*, which includes instances of the same business process. n is the number of token groups.

(v) \mathcal{V} is a set of variables for token groups, and $\mathcal{V} = \{ v_1, v_2, \dots, v_n \}$. Actually, each element v_i of \mathcal{V} is defined on a token group, i.e., $v_i \in \mathcal{V}$. v_i is defined on \mathcal{D}_i , where $1 \leq i \leq n$ and n is the number of token groups.

(vi) $\mathcal{G}: \mathcal{P} \rightarrow \tau$, where each element τ_i of set τ is a set of possible tokens, i.e., $\tau_i \in \tau$ and $\tau_i \in 2^{\mathcal{D}}$.

(vii) $\mathcal{E}: \mathcal{F} \rightarrow \sigma$, where σ is a set of expressions defined on \mathcal{V} .

(viii) $\mathcal{C}: \mathcal{F}^\circ \rightarrow \varepsilon$, where ε is the set of cardinality parameters, i.e., $\varepsilon = \{ [:1], [:n] \}$.

(ix) $\mathcal{Q}: \mathcal{D} \rightarrow \lambda$, where λ is a set of correlation structures.

(x) $\mathcal{I}: \mathcal{P} \rightarrow \theta$, where θ is a set of possible composition of tokens defined in \mathcal{D} .

Explanation:

(1) $(\mathcal{P}, \mathcal{T}, \mathcal{F})$ determines the component net structures of this CorPN.

(2) \mathcal{P}° and \mathcal{F}° describe the messaging behaviours between the business processes of the underlying collaborative business process.

(3) The variables in \mathcal{V} are defined according to each token group, which represents the instances of a business process. Thus, the variables can be used to differentiate the instances of participating business processes and abstract the common behaviours of each business process.

(4) Mapping \mathcal{G} sets up the capacity of each place defined in \mathcal{P} .

(5) Mapping \mathcal{E} sets up the arc expressions to restrict the flowing of tokens.

(6) Mapping \mathcal{C} maps a cardinality parameter onto each arc that connects with an auxiliary place.

(7) Mapping Q combines a correlation structure to each token, and this evolving correlation structure is responsible for recording tokens that correlated with the combined token. Actually, the combined token is the base token of this correlation structure.

(8) Mapping I denotes the initial distribution of tokens.

5 Applying Correspondence Petri Nets

5.1 Generating Correspondence Petri Nets

To generate a CorPN, we first need to collect the participating business processes of this collaborative business process, as well as the messages to use. The conversion from a single business process to an individual Petri net encompasses the following steps:

- (1) Build up token set \mathcal{D} and variable set \mathcal{V} ;
- (2) Set up place capacity expression set \mathcal{G} and arc expression set \mathcal{E} to designate the flowing range of tokens;
- (3) In regard to token producible transitions, we mark a variable symbol 2^v to adjacent outgoing arcs to represent the possibility of all available tokens defined for this business process.
- (4) Initialise correlation set C .

After the four steps, we can obtain the tuple sets for a business process. By incorporating all the obtained tuple sets of all business processes participating in a business collaboration, we may obtain the tuple of a pre-processed CorPN, Σ , for the corresponding collaborative business process. Due to the page limit, we do not discuss this issue intensively.

Algorithm 1 formalises the procedure of assembling these individual Petri nets into a CorPN via message types for the underlying collaborative business process. As this CorPN is created at process level rather than instance level, messages types are therefore used in this algorithm instead of message instances.

In Algorithm 1, function $transition(\Sigma, t)$ returns the transition that stands for task t in CorPN Σ ; function $link(t/p, p/t)$ creates an arc linking transition t to place p , or place p to transition t , and t or p can also be set null to denote an undetermined transition or place; function $priorP/posteriorP(\Sigma, tr)$ returns the prior/posterior place of transition tr in CorPN Σ ; function $priorA/posteriorA(\Sigma, tr)$ returns the prior/posterior arc of transition tr in CorPN Σ ; function $relink(\Sigma, a/p, p/a)$ adjusts a half-determined arc a to connect to/from place p in CorPN Σ .

Algorithm 1. Assembling Petri nets

Input: MSG : the set of unidirectional message types used by business processes in WP .
 Σ : the tuple of the pre-processed CorPN.
Output: Σ' : the CorPN tuple that is updated with auxiliary places, corresponding arcs etc.

1. set $\Sigma' = \Sigma$; $\Pi = \text{null}$; $\Omega = \text{null}$; $sendingArcs = \emptyset$;

```

2. for each  $m \in MSG$ 
3.   if  $m.\rho = 'out'$  then // handling for outgoing message types
4.      $tempT = \emptyset$ ; // create a half-determined arc for each outgoing message type
5.      $k = link( transition(\Sigma', m.\alpha), null )$ ;
6.      $\Sigma'.\mathcal{F}^o \leftarrow k$ ;
7.     for each  $t' \in m.\beta$ 
8.        $\Sigma'.\mathcal{C}^o \leftarrow (k \rightarrow m.f(t'))$ ;  $tempT \leftarrow transition(\Sigma', t')$ ;  $sendingArcs \leftarrow k$ ;
9.     end for
10.     $\Pi \leftarrow (k \rightarrow tempT)$ ;
11.  else // handling for incoming message types
12.     $tempA = \emptyset$ ;
13.    for each task  $t' \in m.\beta$  // decompose the message-receiving transition into a
14.      create transition  $tr$ ; // series of transitions, please refer to Figure 6 (b)
15.       $k = link( priorP( transition(\Sigma', m.\alpha), tr )$ ;
16.       $\Sigma'.\mathcal{F}^o \leftarrow k$ ;
17.       $b = link( tr, posteriorP( transition(m.\alpha) )$ ;
18.       $c = link( null, tr )$ ;  $\Sigma'.\mathcal{F}^o \leftarrow c$ ;  $\Sigma'$ .
19.       $\mathcal{C}^o \leftarrow (c \rightarrow m.f(t'))$ ;
        /* create a half-determined arc for each potential incoming route of this
        message type */
20.       $\Omega \leftarrow ( transition(\Sigma', t') \rightarrow c )$ ;
21.    end for
22.     $\Sigma'.\mathcal{T} = \Sigma'.\mathcal{T} - \{ transition(\Sigma', m.\alpha) \}$ ;
23.     $\Sigma'.\mathcal{F} = \Sigma'.\mathcal{F} - \{ priorA(\Sigma', transition(m.\alpha)), posteriorA(\Sigma', transition(m.\alpha)) \}$ ;
24.  end if
25. end for
26. for each  $k \in sendingArcs$  // link half-determined arcs with proper auxiliary places
27.   create auxiliary place  $px$ ;
28.    $relink(\Sigma', k, px)$ ;
29.   for each transition  $tr \in \Pi(a)$ 
30.      $b = \Omega(tr)$ ;
31.      $\Pi \leftarrow (k \rightarrow (\Pi(k) - \{k\}))$ ;
32.      $\Omega \leftarrow (tr \rightarrow (\Omega(tr) - \{b\}))$ ;
33.      $relink(\Sigma', px, b)$ ;
34.   end for
35. end for

```

In this algorithm, line 4 to line 10 first generates arcs for outgoing message types, and line 12 to line 23 generates arcs for incoming message types. At this stage, these generated arcs are half-determined ones, because we only designate one end of an arc while leave the other end open. To keep the information of multiple receivers or senders of a message, two mapping functions, Π and Ω , are used to record the correspondence between the interaction participating transitions and the generated half-determined arcs. Based on these two mappings, line 26 to line 35 generates auxiliary places and re-links the open ends of those half-determined arcs to proper auxiliary places. In this way, we can connect the individual Petri nets together according to the messaging behaviours between participating business processes.

Following this algorithms, we can generate a CorPN as shown in Figure 6 for the collaborative business process of the motivating example. The sub nets for different

business processes are distinguished with different circles, and the auxiliary places are marked as shaded circles.

Because this CorPN simulates the interaction between multiple business processes, it may own more than one starting place and ending place.

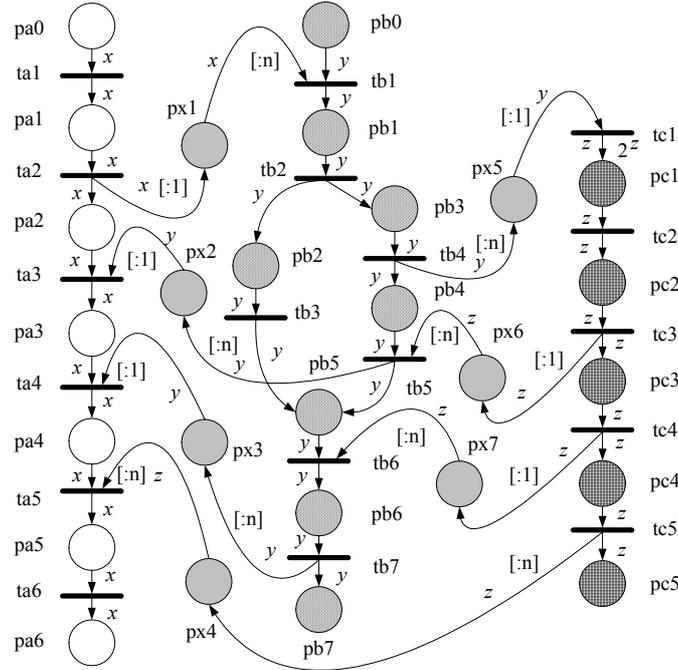


Fig. 6. a CorPN for a collaborative business processes

5.2 Run Time Execution

As discussed in Section 3, workflow correlations occur when business process instances interact. During interactions, a participating instance may inherit pre-existing workflow correlations from its counterparts in case that this interaction has some relation with previous correlations. To update these correlations, each business process instance needs to modify its correlation structure every time after ‘shaking hands’ with partner business process instances.

For example, when the manufacturer contacts shippers for delivery, the manufacturer’s production instance may update its correlation structure with the correlations between retailers’ product-ordering instances and shippers’ assigned shipping instances. In the meantime, these shipping instances also update their correlation structures with the production instance and retailers’ product-ordering instances that are to be served. As for retailers’ product-ordering instances, they may not know these new correlations until the manufacturer notifies them of the delivery date after booking deliveries. Actually, to timely update their correlation structures, retailers need to proactively trace such potential correlations rather than passively

wait for feedbacks. Thus, the correlation handling comprises two procedures, i.e., to generate correlations after interactions and to trace existing correlations through coupled instances.

Algorithm 2 details the procedure of updating correlation structures after collaborating business process instances ‘shake hands’. Following the organisation-oriented view, we classify the tokens representing the host organisation’s involved instances as local tokens, and the ones representing the involved instances of partner business processes as foreign tokens. In this algorithm, function $TYPE(setTK)$ returns which token group that tokens in $setTK$ belong to; function $relatedTK(tk, setTK, \psi)$ returns the set of tokens correlated with token tk from set $setTK$ during interaction ψ ; function $update(tk, setTk)$ updates the content of token tk ’s correlation structure with tokens in $setTk$. The details of function $update$ are given at the end of Algorithm 2.

Algorithm 2. Updating correlation structures

Input: Σ : a CorPN.
 ψ : a real interaction.
 $localTK$: the set of participating local tokens during interaction ψ .
 $foreignTK$: the set of participating foreign tokens during interaction ψ .

Output: Σ' : the updated CorPN.

1. **set** $f = \text{null}$;
2. **set** $\Sigma' = \Sigma$;
3. **for** each $tk \in localTK$
4. $setTK' = relatedTK(tk, foreignTK, \psi)$;
5. $update(tk, setTK')$; // update the correlation structures of local tokens.
6. **for** each $tk^\circ \in setTK'$
7. $f \leftarrow (tk^\circ, tk)$;
8. **end for**
9. **for** each $tk^\circ \in foreignTK$
10. $update(tk^\circ, f(tk^\circ))$; // update the correlation structures of foreign tokens.

// function $update$ is given below
 $update(tk, setTK)$

- u-1. $r^{tk} = \Sigma'.Q(tk)$;
- u-2. **if** $\exists D_i, D_i \in r^{tk} (TYPE(D_i) = TYPE(setTK))$ **then** $r^{tk}.D_i \leftarrow setTK$;
- u-3. **else** $r^{tk}.D_i \leftarrow \{setTK\}$;
- u-4. **for** each $tk_1 \in \bigcup_i r^{tk}.D_i, tk_2 \in setTK$
- u-5. **if** tk_1 is coupled with tk_2 via tk **then** $r^{tk}.R \leftarrow (tk_1, tk_2)$;

Once an interaction occurs, each participating business process instance needs to run Algorithm 2 to update its correlation structure. For each local token, this algorithm searches all participated tokens for the correlated ones with this local token. This job is done by line 3 to line 8. Line 9 and line 10 call function $update$ to update these correlated tokens in the correlation structures of local tokens. In addition, function $update$ also generates proper tuples in relation R of each participated local token, if there exist tokens that are correlated via this local token.

Algorithm 3 describes the procedure of tracing potentially correlated tokens. An organisation may use this algorithm to proactively detect correlated business process instances for its own business process instance. In this algorithm, function $update(tk, setTk)$ is the same with the one in Algorithm 2.

Algorithm 3. Tracing correlated tokens

Input: tk° : the original token to update correlation structure.
 Σ : the CorPN.
Output: Σ' : the updated CorPN.

1. **set** $\Sigma' = \Sigma$;
2. $List = \emptyset$;
3. $oldList = \emptyset$;
4. $r^{tk^\circ} = \Sigma.Q(tk^\circ)$;
5. $List \leftarrow \bigcup_i r^{tk^\circ}.D_i$; // *List is used to store the tokens to check.*
6. **do while** $List \neq \emptyset$
7. **select** $tk \in List$; **remove** tk from $List$;
8. $oldList \leftarrow tk$; // *oldList is used to store the checked tokens.*
9. **for each** $tk' \in \bigcup_i r^{tk}.D_i$
10. **if** $\exists (tk^\circ, tk') \in r^{tk}.R \wedge tk' \notin oldList$ **then** $List \leftarrow tk'$;
11. **end while**
12. $update(tk^\circ, oldList)$;

This tracing procedure, from line 6 to line 11, follows a depth-first strategy to search for correlated tokens. After finding correlated tokens, the host organisation updates the retrieved tokens to its correlation structure by invoking function $update$. This correlation structure determines the logical instance of the specified business process instance. This procedure may be called upon request by the host organisation, for example, at a point that a retailer wants to know shippers' details while waiting for goods delivered by several shippers. Therefore, we do not have to derive this correlation structure for all instances involved in a collaborative business process.

6 Discussion and Conclusion

This paper looked into the problem of instance correspondence in an inter-organisational setting, which is of great importance to business process management yet has not been extensively studied in the literature. By establishing a CorPN model, we proposed a novel method to specify instance correspondences among collaborating business processes. This method captures the dynamics and diversity of business collaboration in terms of workflow cardinality and correlation. With this method, an organisation can clearly track its involvement over a collaborative business process. The detailed contributions of this paper are as follows:

- (1) Unidirectional cardinality parameters and correlation structures to characterise instance correspondence at build time and run time, respectively;
- (2) A correspondence Petri net model with proposed cardinality parameters and

correlation structures etc., for inter-organisational workflow monitoring;

(3) An algorithm for assembling individual business processes into a collaborative process;

(4) Algorithms for specifying workflow correlations and tracing workflow correlations on the fly.

Our future work is to incorporate the proposed method into Business Process Management Notation (BPMN) or BPEL languages, and combine it with our existing relative workflow framework [9]. This future work is expected to provide a comprehensive solution for collaborative business process applications.

Acknowledgements

The work reported in this paper is partly supported by the Australian Research Council discovery project LP0669660.

References

1. Bussler, C.: B2B Integration. New York. Springer-Verlag (2003).
2. Chen, Q. and Hsu, M.: Inter-Enterprise Collaborative Business Process Management. In Proceedings of the 17th International Conference on Data Engineering. Heidelberg, Germany (2001) 253-260.
3. Dumas, M. and ter Hofstede, A.H.M.: UML Activity Diagrams as a Workflow Specification Language. In Proceedings of 4th International Conference on the Unified Modeling Language, Modeling Languages, Concepts, and Tools. Toronto, Canada (2001) 76-90.
4. Barros, A.P., Dumas, M., and ter Hofstede, A.H.M.: Service Interaction Patterns. In Proceedings of Proceedings of the 3rd International Conference on Business Process Management (BPM 2005). Nancy, France (2005) 302-318.
5. van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., and Barros, A.P.: Workflow Patterns. Distributed and Parallel Databases, 14(1) (2003) 5-51.
6. van der Aalst, W.M.P. and ter Hofstede, A.H.M.: YAWL: Yet Another Workflow Language. Information Systems, 30(4) (2005) 245-275.
7. Guabtini, A. and Charoy, F.: Multiple Instantiation in a Dynamic Workflow Environment. In Proceedings of 16th International Conference on Advanced Information Systems Engineering (CAiSE 2004). Riga, Latvia (2004) 175-188.
8. Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., Weerawarana, S.: Business Process Execution Language for Web Services (2003)
9. Zhao, X., Liu, C., and Yang, Y.: An Organisational Perspective on Collaborative Business Processes. In Proceedings of the 3rd International Conference on Business Process Management. Nancy, France (2005) 17-31.
10. Schulz, K. and Orłowska, M.: Facilitating Cross-organisational Workflows with a Workflow View Approach. Data & Knowledge Engineering, 51(1) (2004) 109-147.
11. Chiu, D.K.W., Karlapalem, K., Li, Q., and Kafeza, E.: Workflow View Based E-Contracts in a Cross-Organizational E-Services Environment. Distributed and Parallel Databases, 12(2-3) (2002) 193-216.
12. Zhao, X. and Liu, C.: Tracking over Collaborative Business Processes. In Proceedings of the 4th International Conference on Business Process Management. Vienna, Austria (2006) 33-48.
13. Reisig, W.: A Primer in Petri Net Design. Berlin. Springer (1992).