A SIMULATION-BASED DIFFERENCE DETECTION TECHNIQUE FOR BOTTOM-UP PROCESS RECONCILIATION

Xi Chen and Paul W. H. Chung
Department of Computer Science, Loughborough University, Loughborough, Leicestershire, United Kingdom LE11 3TU

Keywords: Business process, reconciliation, simulation, B2B e-commerce.

Abstract: With the increasing dynamic and changing business environment, bottom-up approaches for business process collaboration is currently receiving a great deal of attention in the research community. Bottom-up approaches are seen to be more flexible then top-down approaches. However, none of the available techniques for process collaboration are suitable for process reconciliation, which is a common problem when different organisations have to work together. In order to address the issue in a bottom-up way, a simulation-based technique for detecting differences between any two given processes is proposed. It is based on the extended definitions of process compatibility for collaboration and is the core of the process reconciliation mechanism.

1 INTRODUCTION

In modern enterprises, workflow technology is commonly used for business process automation. Established business processes represent successful work practice and become a crucial part of corporate assets. In the era of Internet, business processes are unlikely to remain within the boundary of a single organisation. In response to the need for B2B e-commerce, the concept of business process collaboration is emerging.

However, before any two business partners can proceed in conducting B2B e-commerce transactions, their business processes that are involved in the transactions must be compatible with each other at business level (Yang and Papazoglou, 2000), i.e. they have a commonly agreed sequence of exchanging collaborative messages (e.g. a business object like a purchase order or a service invocation request). In general, there are two general approaches to achieve compatibility for process collaboration between two trading partners, namely, top-down and bottom-up. A top-down approach normally involves that people meet and discuss the collaboration, design the collaborative process and implement it locally. On the contrary, a bottom-up approach derives collaborative process from local processes, which is known a difficult task. However, because the top-down approach is labour intensive and expensive (Wombacher, 2005), it is necessary and worthwhile to explore the feasibility of the bottom-up approach in the face of an increasingly dynamic and changing business environment.

Current techniques for process collaboration are not able to provide sufficient computer assistance for bottom-up process reconciliation for a number of reasons. First, the definition of absolute compatibility, adopted by the top-down approach, is too limited. Other categories of compatibility will need to be identified. Secondly, a process reconciliation mechanism is required to consider all the relevant activities. Thirdly, as the core of the mechanism, a technique for process difference detection is needed to be able to address the differences encountered and guide the user towards a possible common process.

In section 2, definitions of process compatibility for collaboration are reviewed and new definitions proposed. On these definitions process reconciliation activities are based. Section 3 depicts the desired process reconciliation mechanism with emphasis on support for a unilateral decision-making process. In section 4, a simulation-based technique for detecting process differences is proposed and is explained by walking through an example. Conclusion is drawn in chapter 5 and future work is described.

The activity-based workflow modelling formalism (Bi and Zhao, 2004) is used in the rest of the paper as it is useful visual representation of

Chen X. and W. H. Chung P. (2007).
A SIMULATION-BASED DIFFERENCE DETECTION TECHNIQUE FOR BOTTOM-UP PROCESS RECONCILIATION.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - SAIC*, pages 72-77
DOI: 10.5220/0002394600720077

business processes. In this representation a vertical synchronisation bar is used as the symbol for an AND vertex; a circle with a cross inside is for an XOR vertex; a rectangle is for a normal activity vertex and an arrow is a directed arc. The only difference from the formalism is the type of split or join (AND or XOR) of a routing vertex is expressed by the pre-condition or post-condition of its neighbour activity vertex, which makes the routing vertex merged with its neighbour. Further more, when needed, an activity vertex explicitly shows its role in the collaboration as either a message sender (s) or receiver (r) (Chen and Chung, 2006) as a superscript.

2 PROCESS COMPATIBILITY FOR COLLABORATION

The purpose of reviewing the definitions of process compatibility for collaboration is to clarify the goal that process reconciliation needs to achieve if a bottom-up approach is followed. According to Hiltrop and Udall (1995), one of the essential principles of negotiation is to get what both sides want rather than to win at any cost. Apart from the unanimously agreed absolute compatibility, another two types of compatibility can be named as deadlock-free compatibility and reconcilable compatibility (Wombacher, 2005; Krukkert, 2003). These three types of compatibility for collaboration are defined below.

Definition 1: Absolute compatibility. Two abstract collaborative processes have the same set of activity vertices, routing vertices and arcs.

Definition 2: Deadlock-free compatibility. If the difference between two processes are only XOR activities on the receiving process and the corresponding sending activities do not split into XOR branches then the two processes are deadlock-free compatible. For example, in Figure 1, process A and B are different but are deadlock-free compatible. When B informs A the only available option by sending a message from $B.b^s$ to $A.b^r$, a deadlock situation will not arise. Therefore, no adjustment is required for both sides.

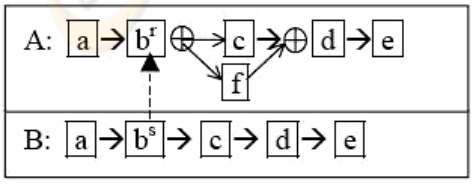

Figure 1: Deadlock-free compatibility.

Definition 3: Reconcilable compatibility. Two processes may appear differently but are reconcilably compatible if they have the same set of activity vertices and the maximum intersection of the sets of possible paths contains at least one path that leads to success. A path denotes a possible execution sequence of all the activities that can be reached based on the current process definition.

As illustrated in Figure 2, process A is different from process B but there exists process C that can be successfully traced through both A and B. Thus, process C can be adopted, which meets the requirements of A and B, and collaboration can proceed.

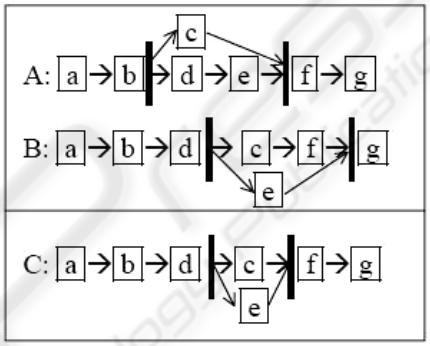

Figure 2: Reconcilable compatibility.

These definitions collectively form a set of acceptance criteria when considering in a bottom-up manner whether any two given processes can be simply adjusted to be compatible for collaboration.

For processes that are not compatible, i.e. they do not meet any one of above definitions, then differences between them must be detected and appropriate changes must be made by either or both partners in order to bring about collaboration.

3 PROCESS RECONCILIATION MECHANISM

Bilateral negotiation is an effective way of reconciling differences in a distributed manner (Li et al., 2003). According to Li et al., such a negotiation comprises a series of unilateral decisions within the control of an underlying negotiation protocol. Since it is common that more than one discrepancy exist between two processes, partners involved are very likely to negotiate and make decisions on multiple issues, which makes the bilateral negotiation a multi-attribute (or multi-issue) one (Fershtman, 1990). Whether to apply a simultaneous or

sequential protocol for a multi-attribute negotiation depends on the problem itself. This is because discrepancies between processes are often interdependent, in order to prevent simultaneous controversial decisions from being made, the negotiation protocol can only be set as a sequential one in the form of alternate proposal of counteroffer after the initial offer. Therefore, within a bilateral negotiation process are many unilateral decision-making steps that take place on both sides.

In the light of the requirement of privacy and the complexity of the decision making process, it is assumed that within a distributed B2B environment, the unilateral decisions are made by people on both sides of the negotiation as shown in Figure 3. This process is repeated until a modified collaborative process is completely formed or collaboration is abandoned.

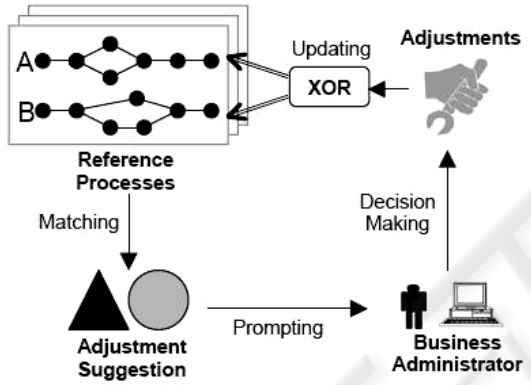

Figure 3: Unilateral decision-making process.

Most current techniques for process matching take a static view towards the differences between two processes. Few have considered process reconciliation when mismatches are encountered. If these techniques were applied directly to process reconciliation, they either confine themselves to minor passive adaptations (Krukkert, 2003; Du et al., 2005; Wombacher, 2005) or handle drastic changes without giving user any selection right (Yeoh et al., 2004). Juan (2006) proposes a string comparison approach to tackling process logic differences however the preliminary case study shows that the approach is restrictive because it requires to express the process in digraph form into strings that is on sequential level. Demanded by the interactive and repetitive nature of the process reconciliation task, discrepancies must be identified according to the progress of reconciliation. Also, the current reference process needs to be continuously updated to reflect the user's decision regarding the previous discrepancy. Such dynamism can also be seen from Figure 3.

4 SIMULATION-BASED PROCESS DIFFERENCE DETECTION TECHNIQUE

Since the unilateral decision-making is based on the discrepancy currently identified, it is required to detect and prompt the discrepancy to the user appropriately. Focusing on the process diagram or the corresponding adjacent matrix would not contribute further to the desired manner of difference detection. What these techniques can reveal are merely structural differences between two digraphs. From process logic's point of view, such differences are trivial. Therefore, the desired technique should be able to reveal process logic information.

In theoretical computer science, a simulation pre-order describes a relation between two state transition systems that one system behaves in the same way as the other or one simulates the other.

Although being used only to match two processes by identifying whether any possible common paths exist, the simulation-based technique proposed by Krukkert (2003) does suggest another view on the problem of process difference detection. According to Krukkert, an activity-based process diagram can be converted to a state transition system (STS) if several prerequisites are met. Related conversion algorithms are also provided.

Therefore, on the one hand, a simulation-based technique can be used to identify common paths that exist between two processes if there are any, which meets the need of matching for reconcilable compatibility. On the other hand, even if no such common path exists, it can be used to reveal to what extend common states exist as well as from which point difference occurs. The differences encountered in the way can be further compared, analysed and prompted to the user to support the decision-making task. Since the simulation-based technique is only valid for processes with the same number of vertices, a pre-treatment and a post-treatment are required to deal with unmatched vertices. The algorithm is constructed as in Table 1.

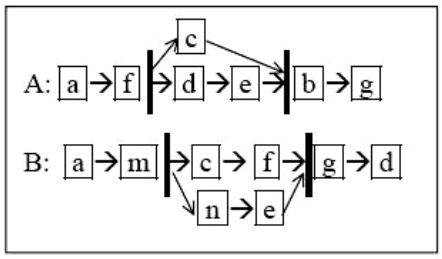

Figure 4: Example processes A, B.

The example illustrated in Figure 4 is used to explain the algorithm.

After a bipartite matching, a, c, d, e, f and g are identified as matching vertices. 'b' is uniquely possessed by A whilst 'm' and 'n' by B. No isomorphic sub-structure is identified. After the matching vertices being deposited, process A_1 and B_1 are the results (illustrated in Figure 5), which are converted to state transition system graphs as shown in Figure 6.

Table 1: Simulation-based difference detection algorithm.

Pre-treatment

```
Conduct bipartite matching of vertices of both process diagrams, extract only
the matched vertices, record the unmatched vertices and their causal relations
with the matched ones of both sides;
Check for isomorphic sub-structure and replace them with single dummy vertex;
```

Core part of the difference detection algorithm

```
WHILE maxCommonPathFlag == FALSE AND terminationFlag == FALSE
    STSD_A = activityDtoSTSD(activityD_currRefA);
    STSD_B = activityDtoSTSD(activityD_currRefB);
    FOR (currLayer = 1; currLayer <= matchedVertexNum; currLayer++)
        commonState = StateComparison(currLayer);
        IF(commonState)
            maxCommonPath = maxCommonPath + commonState;
            verifyMaxCommonPath(maxCommonPath);
            IF(currLayer == matchedVertexNum)
                maxCommonPathFlag = TRUE;
                break;
        ELSE
            dispFwd = getDispFwd();
            dispBkwd = getDispBkwd();
            adjustmentSuggestion = getAdjustmentSuggestion(dispFwd, dispBkwd);
            tempActivityD = adjust(adjustmentSuggestion, activityD_currRefA);
            association = checkAssociation(adjustmentSuggestion);
            IF(association)
                promptUser(theUnmatchedVertices);
                decision_unmatched = getUserDecision();
                recordDecision(decision_unmatched);
            promptUser(adjustmentSuggestion, tempActivityD);
            decision = getUserDecision();
            adjust(decision, activityD_compromising);
    END-FOR
END-WHILE
```

Post-treatment

```
adjust(decision_unmatched, activityD_compromising);
Prompt the remaining unmatched vertices and unique exclusive OR branches (if
any) as discrepancies to the user;
decision_unmatched = getUserDecision();
adjust(decision_unmatched, activityD_compromising);
```

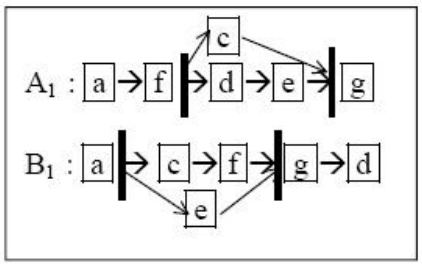

Figure 5: Result of matched vertices' extraction.

A_1		B_1		
start | a		start | a		
a		a		
a | f		a | ce		dispFwd
f		c	e	
af | cd		ac | ef	ae | c	
c	d	ec	f-	Common States Identified
afc | d	afd | ce	ace | f	acf | e	
dc	-e	fe		
afcd | e	afde | c	acef | g		
ec		g		
afcde | g		acefg | d		
g		d		dispBkwd
acdefg | end		acefg | end		
A_1.dispFwd:	ce → f	A_1.dispBkwd	g → d	
Adjustment Suggestion:	ce → f @A_1	If rejected:	f → ce @B_1	

Figure 6: 1st round comparison between STS graphs of A_1 and B_1.

Common states are examined in a forward direction. When discrepancy is encountered, it is recorded as forward discrepancy (dispFwd) and a backward examination is carried out with the encountered backward discrepancy (dispBkwd) recorded. Since the vertices involved in A_1.dispFwd and A_1.dispBkwd are not the same, no swap operation is required. The A_1.dispFwd is by default selected as the current adjustment suggestion and is prompted to the user as "'ce' should be moved immediately in front of 'f' in process A_1" together with a corresponding activity diagram representation.

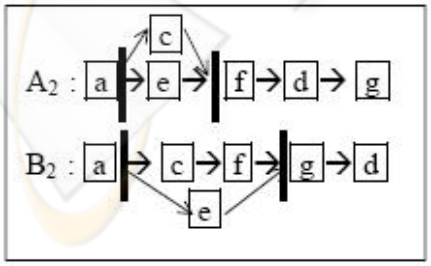

Figure 7: Activity diagrams of A_2 and B_2.

Assuming that user A accepts this suggestion and therefore process A_1 needs to be adjusted accordingly to A_2 whilst B_2 remains the same as B_1, whose activity diagrams and corresponding STS graphs are shown in Figure 7 and Figure 8.

A_2		B_2		
start | a		start | a		
a		a		
a | ce		a | ce		
c	e	c	e	
ac | e	ae | c	ac | ef	ae | c	Common States Identified
ec		ec	f-	
ace | f		ace | f	acf | e	
f		fe		
acef | d		acef | g		dispFwd
d		g		
acefd | g		acefg | d		
g		d		dispBkwd
acdefg | end		acefg | end		
A_2.dispFwd:	g → d	A_2.dispBkwd:	g → d	
Adjustment Suggestion:	swap g,d @A_2	If rejected:	swap g,d @B_2	

Figure 8: 2nd round comparison between STS graphs of A_2 and B_2.

During the second round comparison, four more common states are identified, which are 'a | ce', 'ac | e', 'ae | c' and 'ace | f'. Following on from this, A_2.dispFwd and A_2.dispBkwd are identified and the involving vertices are evaluated as the same, which implies the adjustment suggestion should be a vertices swap operation between 'g' and 'd' in process A_2. After being prompted both in words and graphically, assuming that user A rejects the discrepancy this time, process B_2 is expected to make concession by being adjusted instead, i.e. swapping 'g' and 'd' in B_2 to form B_3 whilst, A_3 remains the same as A_2. Shown in Figure 9 are the resulting activity diagrams of A_3 and B_3. After the third round comparison between STS graphs, a common path is found, whose corresponding activity diagram C is shown in Figure 10.

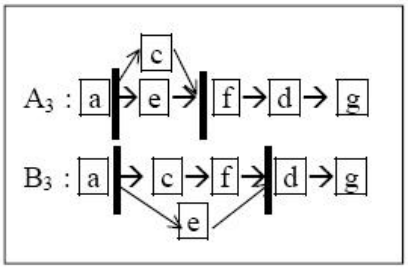

Figure 9: Activity diagrams of A_3 and B_3.

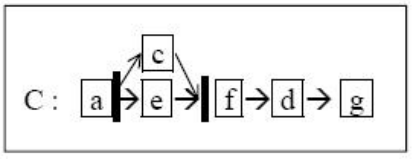

Figure 10: Common path identified between A_3 and B_3.

Vertices that have no match (b, m and n) are highlighted to user A for further decisions. C_1 is shown in Figure 11 as one of the possible resulting common collaborative process following organisation A's unilateral decisions.

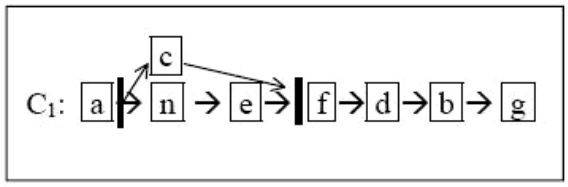

Figure 11: One possible resulting common collaborative process following organisation A's unilateral decisions.

When process C_1, as the counteroffer, is passed to organisation B, the same procedure is followed by user B to carry out their own decision-making. Other issues, e.g. strategy of unilateral decision-making, negotiation termination condition, are also important but beyond the scope of this paper.

5 CONCLUSIONS

The simulation-based process difference detection technique is proposed to provide support during distributed process compatibility negotiation by helping users concentrate on a series of adjustment suggestions to agree on a common path as soon as possible. With the support of such a technique, human efforts are saved from the labour-intensive task and corporate assets in terms of business processes are preserved and put into good use. Also, the ability to start from two predefined process logics enables the technique to be used from bottom up, which makes it possible to replace the expensive top-down approach to cross-organisational process reconciliation. Furthermore, the technique can also be applied in the area of process compliance (Cheung, 2003) as well as process benchmarking (Juan and Ou-Yang, 2005; Juan, 2006), in which customer defined processes are checked for compliance issues against certain standard or best-of-breed process.

In addition to preliminary case studies having been carried out, the effectiveness of the technique needs to be further evaluated through a full range of real life business processes. It is also envisioned that a comprehensive business process collaboration framework is needed to take full advantage of such a technique, within which the execution components are mentioned in Chen and Chung (2006).

REFERENCES

Bi, H., Zhao, J., 2004. Applying Propositional Logic to Workflow Verification. *Information Technology and Management* 5, 293–318, 2004.

Chen, X., Chung, P., 2006. Cross-Organisational Workflow Enactment Via Progressive Linking by Run-Time Agents. In *Ali, M. and Dapoigny, R. (Eds.): Proceedings of 19th International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2006*, LNAI 4031, pages 54-59, 2006.

Cheung, Y.C., 2003. *COMPLIANCE FLOW - AN INTELLIGENT WORKFLOW MANAGEMENT SYSTEM TO SUPPORT ENGINEERING PROCESSES*, Ph.D. Thesis, Loughborough University, 2003.

Du, Z., Huai, J., Liu, Y., Hu, C., and Lei., 2005. IPR: Automated Interaction Process Reconciliation. In *Proceedings of IEEE/ACM International Conference on Web Intelligence (WI)*, 2005.

Fershman, C. The importance of the agenda in bargaining. *Games and Economic Behavior*, 2(224-238), 1990.

Hiltrop, J., Udall, S., 1995. *The Essence of Negotiation*, Prentice Hall, London 1995.

Juan, Y.C. and Ou-Yang, C., 2005. A process logic comparison approach to support business process benchmarking, *International Journal of Advanced Manufacturing Technology*, Vol. 26, pages 191-210, 2005.

Juan, Y.C., 2006. A String Comparison Approach to Process Logic Differences between Business Process Models. In *Proceedings of the Joint Conference on Information Sciences 2006*, Atlantis Press, October 2006. doi:10.2991/jcis.2006.23

Krukkert, D., 2003. *Matchmaking of ebXML business processes*, Technical Report IST-28584-OX_D2.3_v.2.0, openXchange Project, Oct 2003.

Li, C., Giampapa, J.A., and Sycara, K., 2003. *A Review of Research Literature on Bilateral Negotiations*, Tech. Report CMU-RI-TR-03-41, Robotics Institute, Carnegie Mellon University, November, 2003.

Wombacher, A., 2005. *Decentralized establishment of consistent, multi-lateral collaborations*, PhD Thesis at Technical University Darmstadt, Facultiy of Informatics, 2005.

Yang, J., Papazoglou, M., 2000. Interoperation Support for Electronic Business. *COMMUNICATIONS OF THE ACM* June Vol. 43, No. 6, 39-47, 2000.

Yeoh, M.L., Chung, P.W.H., Anumba, C.J., El-Hamalawi, A., Motawa, I.A., 2004. Process change identification using workflow specification matching. In *Proceedings of the Tenth Americas Conference on Information Systems*, New York, New York, August 2004.