Proceedings of the 2011 International Conference on Industrial Engineering and Operations Management
Kuala Lumpur, Malaysia, January 22 – 24, 2011

Process Change Pattern Analysis in Collaborative Processes

Aekyung Kim and Jae-Yoon Jung
Department of Industrial and Management System Engineering
Kyung Hee University, Yongin-si, Gyeonggi-do, 446-701, Republic of Korea

Abstract

Traditional business process management systems provide verification tools of process models to deploy and automate the models. However, there are not so many studies on how to maintain systematically collaborative process models such as supply chain processes when companies are willing to change and update the collaborative process models. In this paper, we analyze change patterns of collaborative processes and declare 19 change patterns. In addition, we apply the change patterns to the process interoperability patterns in order to identify the change problems in case of independent process changes of collaborative processes. As a result, we devise an independency checking algorithm of process changes in collaborative processes

Keywords

Workflow patterns, process change patterns, verification, collaborative process

1. Introduction

Recently, as the competition among companies is extended from a single company to networked companies such as supply chain, a type of enterprise is changed a virtual enterprise based on collaborative network. Such collaboration of business network is performed on basis of information sharing among companies, electronic document exchange, and standardized process integration (i.e. ebXML [1] or RosettaNet [2]).

As business environment changes, companies participating in collaboration often want to change business processes easily and flexibly. For the purpose, business process management system (BPMS) can support to design and automate their collaborative processes.

Verification of business process, which is one of highlighted issues in business process modeling and analysis, is to check the integrity of such process models for automation in information systems. Traditional business process management systems provide verification tools of process models to deploy and automate the models.

However, there are not so many studies on how to maintain systematically collaborative process models such as supply chain processes when companies design and update the collaborative process models. If a method of analyzing and verifying changes of collaborative processes is developed, companies are able to respond quickly and flexibly to the changing business environment.

In this paper, we first classified change categories of collaborative processes models into 5 categories: ‘Activity split’, ‘Activity merge’, ‘Activity deletion’, ‘Activity insertion’, and ‘Structural change’. From the categories, we then proposed 19 change patterns of collaborative processes. In addition, we discovered three types of modification problems which 19 change patterns in collaborative processes may result in. Moreover, we devised an independency checking algorithm of process changes in collaborative processes. In general, companies need to often change business processes in the rapidly changing business environment. When a company changes business processes, if it is possible to find any problem of process changes based on the change patterns which are analyzed in this paper, it can maintain collaborative processes in easy and robust manners. In detail, the company can easily change and update its own process by decision whether the individual changes affect the overall collaborative process. Furthermore, by warning to users the possibility which independent process changes can influence their business partner, it helps companies quickly respond and prevent possible problems which can be occurred in future. Finally, a verification algorithm for changes of collaborative processes which is devised in this paper can be vastly used in business process modeling and analysis.

2. Related Work

Gamma et al. [3] first suggested 23 design patterns which are repeatedly occurred in object-oriented systems. van der Aalst et al. [4] became interested in workflow patterns which are appeared repeatedly in the workflow design

based on Petri-nets. The first deliverable of their research project was a set of twenty patterns describing the control-flow perspective of workflow systems: Basic control flow patterns, Advanced branching and synchronization patterns, Structural patterns, Multiple instance patterns, State-based patterns, and Cancellation patterns [5].
Kim and Kim [6] presented an overall framework which can support efficiently of business process improvement and maintenance in a dynamic environment. Besides, they first found change types which appear frequently in the course of a business process redesign based on workflow control patterns, and then presented patterns from the types. They classified 16 process change patterns into three main types: Activity split/change, Activity expansion/delete, and Activity combination/change types. Business process change patterns are able to support more systematically the steps of business process improvement in the lifecycle. Companies which manage business process change based on the process change patterns are able to manage more easily and efficiently various process changes, which may be often caused in dynamic environment.
Jung et al. [7] analyzed various types of interoperation between business processes and identified six primitive interoperability patterns as building blocks for expressing complex interactions. These primitives were extended from three interoperability models of Workflow Management Coalition (WfMC): chained, nested, and synchronized. In a chained model, one process triggers the creation and enactment of another process, but it takes no further interest in what happens next. This model subdivides into two types of patterns: Chained Substitutive (CS) and Chained Additive (CA). In a nested model, the invoking process takes execution results from the invoked process at a particular activity. The model is subdivided into three patterns: Nested Synchronous (NS), Nested Deferred (ND), and Nested Parallel (NP). The synchronized model follows only one pattern — parallel synchronized (PS) — in which two processes synchronize at a specific point. Only after both of them reach that point can they continue their execution.

3. Process Change Patterns

In this paper, we redefine process change patterns to adopt for collaborative process models by referring to process change patterns proposed by Kim and Kim [6].
We first categorized the process changes into five types in Table 1. ‘Activity split’ type is that an activity is divided into multiple specialized activities. ‘Activity merge’ type is that multiple activities are merged into a generalized activity. ‘Activity deletion’ type is to delete an existing activity that is no longer necessary. ‘Activity addition’ type is to insert a new activity or add a branch to the original process. ‘Structural change’ type is to change control-flow while retaining the existing activities. The list of the redefined process change patterns and their characteristics are summarized in Table 1.

Table 1: Characteristic of process change types

Process change types	# of changed activities		Affected range		Process change patterns
	before	after	activity	scope	
Activity split	1	N	O		Serial split (C1), Parallel split (C2), Selective split (C3)
Activity merge	N	1		O	Merge of serial (C4), Merge of parallel (C5), Merge of selection (C6)
Activity deletion	1	0 or 1	O		Complete deletion (C7), Partial deletion (C8), Replacement (C9)
Activity addition	0	1 to N		O	Serial insertion (C10), Parallel addition (C11), Selective addition (C12)
Structural change	N	N		O	Reverse order (C13), Parallelization of serial (C14), Serialization of parallel (C15), Selective split of serial (C16), Serialization of selection (C17), Selective split of parallel (C18), Parallelization of selection (C19)

4. Analysis of Independent Process Change

4.1 Patterns Allowed to Independent Change

In this research, we analyzed 19 process change patterns by adapting to every activity in six process interoperability patterns [7] in order to develop the independency checking algorithm of independent changes in collaborative processes. In each case that a process change pattern was applied to a process interoperability pattern, we checked whether the change pattern causes any problem to whole collaborative process. As a result, C4, C5, C6, C10, C11, C15, C17, and C19 are patterns which are allowed to independent changes in every interoperability pattern.

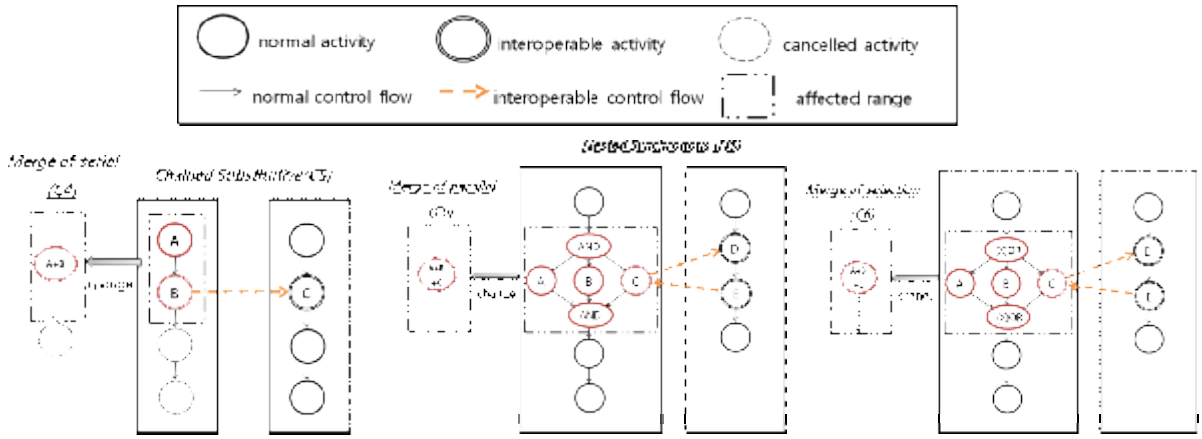

Figure 1: C4 and CS patterns

Figure 2: C5 and NS patterns

Figure 3: C6 and NS patterns

Figure 1 shows an example of a chained substitutive (CS) pattern that activity B sends activity C a message. If activity A and activity B in series are merged into activity 'A+B' in C4 (Merge of serial) pattern, the change will not occur any problem in the whole process only if the message is still sent to activity C. All the other interoperability patterns have the same results for the C4 pattern. In the same way, C5 (Merge of parallel) and C6 (Merge of selection) patterns do not occur any problem when they are applied to all activities in six interoperability patterns (see Figure 2 and 3).

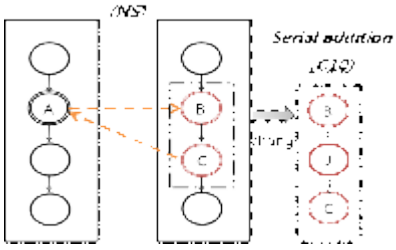

Figure 4: C10 and NS patterns

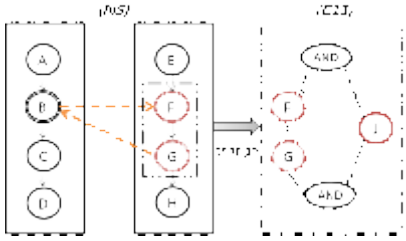

Figure 5: C11 and NS patterns

C10 (Serial insertion) and C11 (Parallel addition) patterns are allowed to independent changes because the inserted or added activities do not affect the message sending or receiving in the whole process. Note that C12 (Selective addition) pattern makes the branch of the sending or receiving activity be optionally executed so that the change is not allowed.

C15 (Serialization of parallel), C17 (Serialization of selection), and C19 (Parallelization of selection) patterns do not cause any problem because they guarantee the executions of all original activities after changes of serialization, as shown in Figure 6, 7, and 8. However, C14 (Parallelization of serial) is not allowed to independent changes because it may change the order of sending and receiving activities which were in serial.

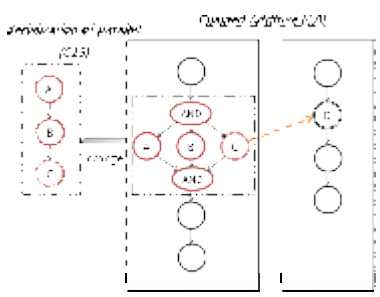

Figure 6: C15 and CA patterns

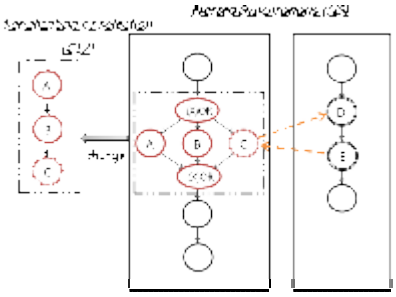

Figure 7: C17 and NS patterns

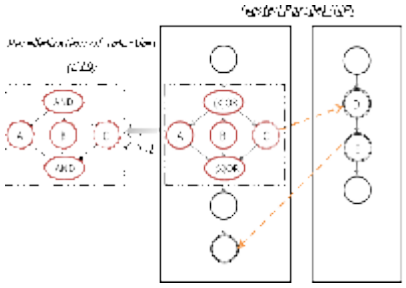

Figure 8: C19 and NP patterns

4.2 Problems of Independent Changes

The other patterns may cause several problems for the execution of the whole collaborative process. In this subsection, we analyzed the problems which can be occurred when some change patterns are applied to six process interoperability patterns. After that, we categorized three types of problems which were caused by process changes in a collaborative process: *non-execution*, *reverse execution*, and *partial restarting* problems.

The first type is *non-execution* problem that the sending or receiving activity does not guarantee its execution after changes such as C3 (Selective split) and C7 (Complete deletion) patterns. Figure 9 shows that when an activity B is changed in C3 pattern, activity B1 or B2 is executed and the receiving activity B3 is not. It will cause the activity A to fail the sending o fits message.

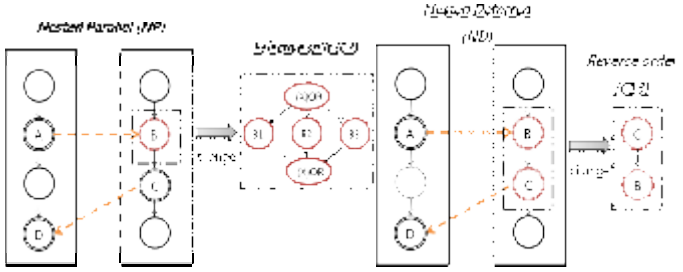

Figure 9: 'non-execution' problem

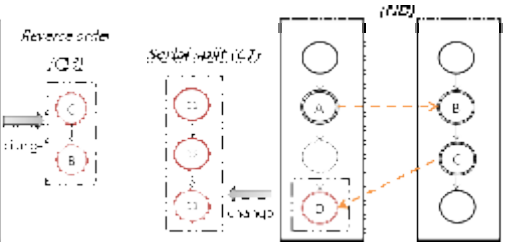

Figure 10: 'reverse execution' problem

Figure 11: 'partial restarting' problem

The second is *reverse execution* problem that the order of sending and receiving activities is changed to cause the problem of message changes. Figure 11 illustrates that the process change makes the order of sending activity B and receiving activity C be reversed.

The final is *partial restarting* problem that all activities are not restarted after changes. This type of problems occurs only in ND (Nested Deferred) interoperability pattern. In the original model of Figure 11, activity D receives a message from activity C and then restarts. If activity D is split into activities D1, D2, and D3 in series and an activity D3 receives the message, activities D1 and D2 will be ignored when D3 restarts. Note that in *partial restarting* problem, the whole problem will continue to execute with some activities (i.e. D1 and D2) not executed, while in *non-execution* problem, the execution of the whole process.

4.3 Analysis Results of Independent Process Change

Table 2 shows the results obtained by applying 19 change patterns to six interoperability patterns. It summarizes which problem each of 19 process change patterns may cause in interoperability patterns.

Table 2: Problems caused by process change patterns

Process changes		Problems of independent changes		
Types	patterns	non-execution	reverse execution	partial restarting
Activity Split	- Serial split (C1)			O
	- Parallel split (C2)			O
	- Selective split (C3)	O		O
Activity Merge	- Merge of serial (C4)			
	- Merge of parallel (C5)			
	- Merge of selection (C6)			
Activity Deletion	- Complete deletion (C7)	O		O
	- Partial deletion (C8)	O		O
	- Replacement (C9)	O		O
Activity Insertion	- Serial insertion (C10)			
	- Parallel addition (C11)			O
	- Selective addition (C12)	O		O
Structural change	- Reverse order (C13)		O	
	- Parallelization of serial (C14)		O	
	- Serialization of parallel (C15)			
	- Selective split of serial (C16)	O	O	
	- Serialization of selection (C17)			
	- Selective split of parallel (C18)	O		
	- Parallelization of selection (C19)			

5. Independency Checking of Process Changes

We devised an independency checking algorithm of process changes in collaborative processes. The algorithm can be used to verify independent process changes when a company wants to change a part of the collaborative processes as shown in an example of Figure 12.

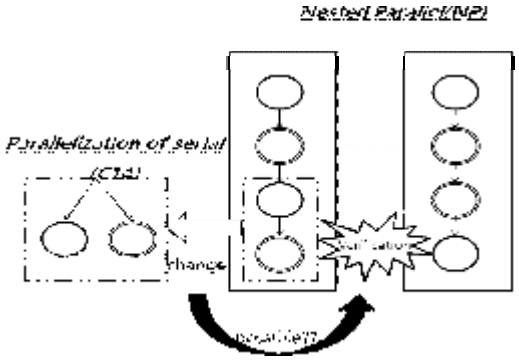

Figure 12: Example of independency checking of process changes

The algorithm is developed based on the results of analysis described in Table 2. It is drawn as a flow chart in Figure 13. The algorithm first starts with a process change pattern p detected from a given process change. If the pattern p is belonged to those are allowed to independent process change (P^A) or p does not include any sending (a^S) or receiving (a^R) activity in the scope of the patterns $Scope(p)$, the algorithm decides to allow the given change because it does not affect the execution of the whole collaborative process. If p includes a^S or a^R and it has non-execution problem ($p \in P^{NE}$), the change is not allowed regardless of interoperability patterns. If p does not have non-execution problem, but reverse execution ($p \in P^{RE}$), p should be checked to lie in ND or NP pattern. If it is true and both a^S or a^R are in the scope of ND or NP, the change is not allowed because of the reverse execution problem. Finally, partial restarting problem is checked. Because the problem occurs only in ND interoperability pattern, p is not allowed if it lies in ND and the restart activity is in the scope of the change pattern p. Otherwise, it is allowed with the termination of the algorithm.

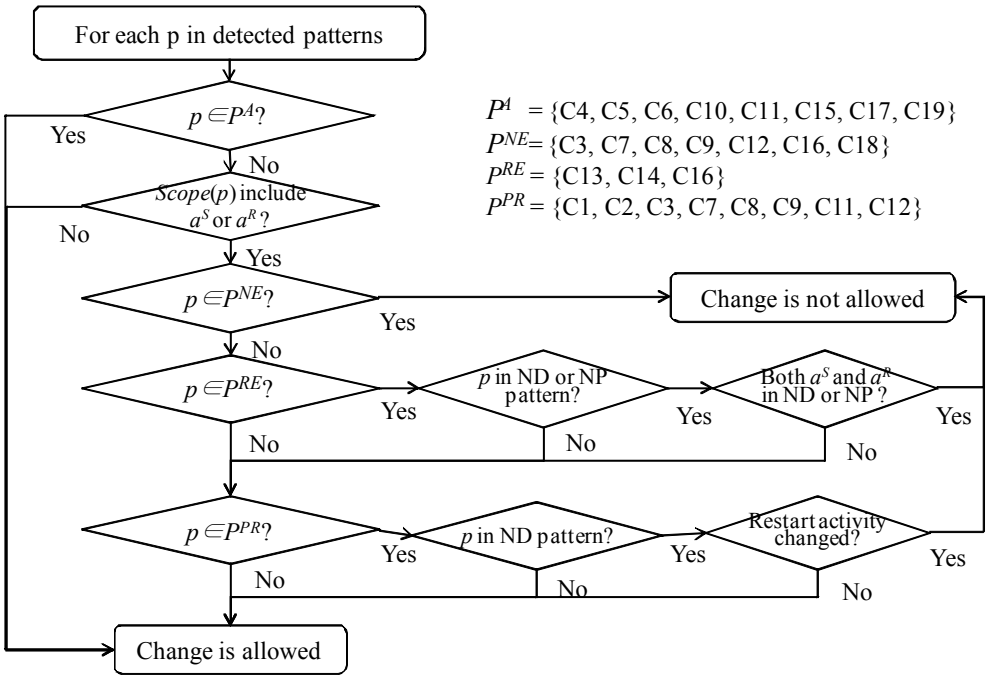

Figure 13: Independency Checking Algorithm for Process Changes

6. Conclusion

In this paper, we analyzed process changes in collaborative processes and declared 19 process change patterns. Next, we discovered the possible problems of the process changes that may affect the right execution of the whole process. Finally, we devised an independency checking algorithm for process changes, which can be used for automatically verifying independent process changes in collaborative processes.

Companies participating in supply chain want to often change business processes. In their updates and changes, arbitrary changes may cause many problems to the execution of whole collaborative processes. Even if an importance of design and maintenance of collaboration is awarded, most of the studies focus on mainly technical problems, not collaborative process design.

From the reason, we devised an algorithm that can easily and quickly check whether partial changes of collaborative processes is allowed to independent change. The process change patterns and the independency checking algorithm proposed in the research will provide a useful means which companies can efficiently and quickly design and maintain the process models in their collaboration.

Acknowledgements

This work was supported by the i-manufacturing program funded by the Ministry of Knowledge Economy (MKE, Korea).

References

1. ebXML web site. http://www.ebxml.org/.
2. RosettaNet web site. http://www.rosettanet.org/.
3. Gamma, E., Helm, R., Johnson, R., and Vlissides, J., 1995, Design patterns: Elements of Reusable Object-Oriented Software; Reading, Addison-Wesley, Massachusetts.
4. Workflow patterns homepage, http://www.workflowpatterns.com/.
5. van der Aalst, W.M.P, ter Hofstede, A.H.M., Kiepuszewski, B., and Barros, A.P., 2003, "Workflow Patterns," Distributed and Parallel Databases, 14(1), 5-51.
6. Kim, D., and Kim, M., 2007, "Business Process Change Patterns," Spring Conference on KIIE/KORMS, Jinju, Korea.
7. Jung, J.-Y., Hur, W., Kang, S.-H., and Kim, H., 2004, "Business Process Choreography for B2B Collaboration," IEEE Internet Computing, 8(1), 37-45.