Detection of Anti-Patterns in the Control Flow of Collaborative Business Processes

Jorge Roa[1], Omar Chiotti[2], and Pablo Villarreal[1]

[1] CIDISI, Universidad Tecnológica Nacional - Facultad Regional Santa Fe, Santa Fe, Argentina
{jroa, pvillarr}@frsf.utn.edu.ar
[2] INGAR-CONICET, Santa Fe, Argentina
chiotti@santafe-conicet.gov.ar

Abstract.
The verification of the behavior of Collaborative Business Processes is an important aspect to consider when developing inter-organizational systems. In this work, a verification approach for the control flow of collaborative processes based on anti-patterns is proposed to improve the performance of verification. The approach supports the verification of complex constructs for advanced synchronization, multiple instances, and exception management. To this aim, 10 anti-patterns were defined from a repository of process models, and a tool which implements the anti-patterns was developed to evaluate the verification approach. Results indicate that, at worst, the verification time is less than half a millisecond, even for models with complex control flow constructs.

Keywords: Business Process, Verification, Soundness, Correctness Properties

1 Introduction

Nowadays, organizations are establishing dynamic and flexible collaborations with their partners. Collaboration implies that organizations agree on common business goals, coordinate their actions, exchange information, and jointly define and execute collaborative business processes [1].

A Collaborative Business Process (CBP) defines the global view of interactions of inter-organizational collaborations, as well as the way the inter-organizational systems will interact [1, 2]. There are different languages to define models and specifications of CBPs such as BPMN [3], WS-CDL [4], UMM [5], or UP-ColBPIP [2].

The verification of the control flow of CBPs is an important aspect to consider when developing inter-organizational systems, since the control flow defines the behavior of the collaboration. An incorrect definition of the behavior of a CBP could affect internal processes of organizations and cause a propagation of errors beyond their boundaries, generate a loss of trust and affect the efficiency between organizations, or frustrate the achievement of common goals agreed in the collaboration.

The verification of the control flow of business processes has been widely studied in the last decade achieving important results [6]. A well-known property to verify business processes is soundness [7, 8], which enables the detection of deadlocks and lacks of synchronizations in the control flow of business processes. In previous work, we proposed a verification method for the behavior of CBPs [9, 10]. The method proposes the Global Interaction soundness property as the main correctness criterion, which is based on the classical notion of soundness [7]. However, these methods rely on state space exploration, which may cause a negative impact on performance due to the state space explosion problem [11].

In order to improve the performance of verification methods, in this work, we propose an approach to verify the control flow of CBPs by means of behavioral anti-patterns. A *behavioral anti-pattern* of CBPs is a predefined and well-known situation of a deficient specification of the control flow of CBPs. It can be seen as a combination of control flow constructs which should be avoided when modeling CBPs. To this aim, we used the verification method based on GI-Nets we proposed in previous work [9, 10] to define 10 anti-patterns for the control flow of the UP-ColBPIP language, and developed a tool based on the Eclipse Platform [12] which supports the verification of UP-ColBPIP models by means of such anti-patterns.

The approach was evaluated with a repository of 880 UP-ColBPIP models. Each CBP model was verified with anti-patterns and with a formal verification method based on Hierarchical and Colored Petri nets we presented in previous work [9, 10, 22]. Results show that the verification based on anti-patterns is as accurate as the formal verification method, and it clearly improves the performance of the latter. This holds even for models with complex constructs and having more than 100 elements, where such models are intractable for the formal verification method.

This work is structured as follows. Section 2 describes the UP-ColBPIP language. Section 3 introduces the behavioral anti-patterns for CBPs. Section 4 presents the evaluation of the proposed anti-patterns. Section 5 describes related work. Finally, Section 6 presents conclusions and future work.

2 The UP-ColBPIP Language

The UP-ColBPIP language [1, 2] extends the UML2 semantics and encourages the modeling of technology-independent CBPs in a top-down approach. UP-ColBPIP supports the modeling of interaction protocols through UML2 Sequence Diagrams to represent the global view of interactions through a choreography of business messages between organizations playing different roles.

Partners and the Role they fulfill are represented through lifelines (Fig. 1.a). An interaction protocol is described by a temporal and ordered sequence of elements that cross the lifelines: business messages, terminations, protocol references, control flow segments and interaction paths.

A business message (Fig. 1.b) is the atomic building block of an interaction protocol and defines a one-way asynchronous interaction between two roles, a sender and a receiver. Its semantics is defined by the associated speech act that represents the intention the sender has with respect to the business document (information) exchanged in the message. A time constraint (Fig. 1.b) denotes a deadline or time-out on messages, control flow segments or protocols.

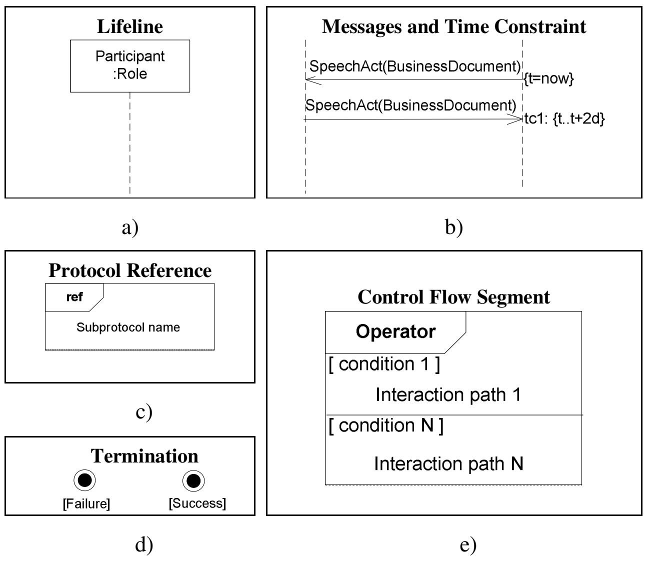


Fig. 1. Modeling constructs of UP-ColBPIP

A protocol reference (Fig. 1.c) represents a sub-protocol. Protocols have an implicit termination. A termination (see Fig. 1.d) defines an explicit end event of a protocol, which can be success or failure, to provide a logical indication of the end.

A control flow segment (CFS) represents complex message sequences (Fig. 1.e). It contains a control flow operator and one or more interaction paths. An interaction path contains an ordered sequence of elements: messages, protocol references, terminations and nested CFSs. An interaction path may have a condition that constrains its execution.

A CFS whose operator is *And* represents the parallel execution of paths to model concurrent interactions. They can be executed simultaneously or in any order as long as the order of the messages on each path is maintained. The *And* operator also represents the synchronization of the paths, i.e. the thread of control is passed to the next protocol element when all paths are completed.

A CFS whose operator is *Xor* represents two or more alternative paths but only one of them can be executed. The thread of control is passed to the next element when the selected path is completed.

A CFS whose operator is *Or* represents two or more alternative paths that can be executed according to the evaluation results of their conditions. Four types of path synchronization can be defined. (1) Synchronizing Merge (<<Sync-Merge>>): the thread of control is passed to the next protocol element when each enabled path is completed. (2) Discriminator (<<Disc>>): the control is passed to the next element when one path is completed. (3) N out of M Join (<<N out of M>>): the control is passed to the next element once N of the M paths are completed. The remaining paths in (2) and (3) are ignored but they are not cancelled. (4) Multi-merge (<<Multi-Merge>>): for each completed path a thread of control is passed to the next element.

A CFS whose operator is *Loop* contains one interaction path that can be executed several times while its condition is satisfied. A Loop Until has the condition (1, n) so that the path is executed at least once; a Loop While has the condition (0, n) and it means that the execution of the path is performed zero or more times.

A CFS whose operator is *Exception* defines the path to be followed after an exception takes place, which is identified at design time. A CFS with the *Exception* operator consists of one path that defines the scope of the exceptions (for all protocol element involved in the path) and other exception handler paths, one for each exception to be caught and managed. After an exception handler path is completed, the protocol continues with its normal execution.

A CFS whose operator is *Cancel* defines the path to be followed after an exception takes place. The difference between *Cancel* and *Exception* is that the former finalizes the execution of the protocol when the path that handles the exception is completed.

A CFS whose operator is *Multiple Instances* represents that its interaction path can be executed several times in parallel. Thus, a message or a sequence of protocol elements on this path will have multiple instances (MI) in parallel. Four types of multiple instances can be defined. (1) The MI at design-time (<<DT>>) and (2) the MI at run-time (<<RT>>) have an attribute N that contains the number of instances to be executed. In the former, the attribute is defined at design time and in the latter it takes the value of an attribute of a business document involved in the protocol. (3) The MI without a priori run-time knowledge (<<WRTK>>) requires a condition that enables the creation of new instances only if the condition evaluation is true. Otherwise, no new instances are created. The above types of MI imply that the multiple instances of the path are synchronized when they are completed. (4) Multiple instances without synchronization are denoted by the <<WS>> label.

Figure 2 shows a Collaborative Demand Forecast CBP modeled with UP-ColBPIP. This CBP is based on a real world case study of the application of a collaborative model for the supply chain management of desktop computers and notebooks. There are two organizations collaborating to agree on a demand forecast of final products. The customer sends a forecast request, which the supplier may agree or refuse. If the request is refused, the CBP ends with a failure. If the supplier agrees, the customer must send in parallel a forecast for each point of sales (POS) and a plan of programmed events. With this information, the supplier can generate the demand forecast and send it to the customer. The customer has at most two days to send the plan of programmed events. Otherwise the CBP must be canceled.

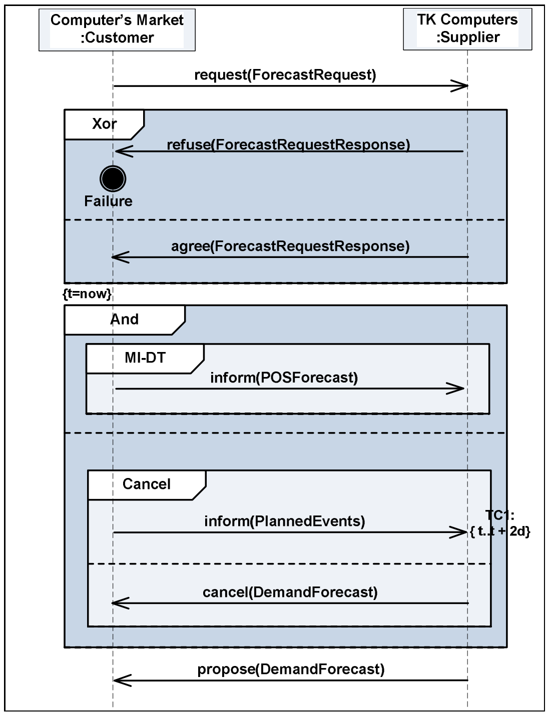


Fig. 2. Collaborative Demand Forecast CBP model defined with UP-ColBPIP

3 Behavioral Anti-Patterns for UP-ColBPIP

An anti-pattern must be defined as general as possible without considering any specific CBP. In order to discover and specify anti-patterns it is necessary to detect combinations of constructs that are repeated finitely often and always lead to the same behavioral problem. Hence, the repeated application of verification methods to CBPs enables the discovery and specification of anti-patterns.

To this aim, a repository of 880 UP-ColBPIP models was used. Details of this repository can be found in Section 4. Each model was verified with the formal method based on GI-Nets we proposed in previous work [9, 10]. This method determines if a CBP is sound (or unsound). If the behavior of a CBP model is free of deadlocks and lacks of synchronizations, then it is sound. Otherwise, it is unsound.

From the repository we selected unsound CBP models. Results returned by the verification method were analyzed to infer the anti-patterns from the unsound CBP models. In total, 10 behavioral anti-patterns for UP-ColBPIP were specified by following the approach proposed in [22].

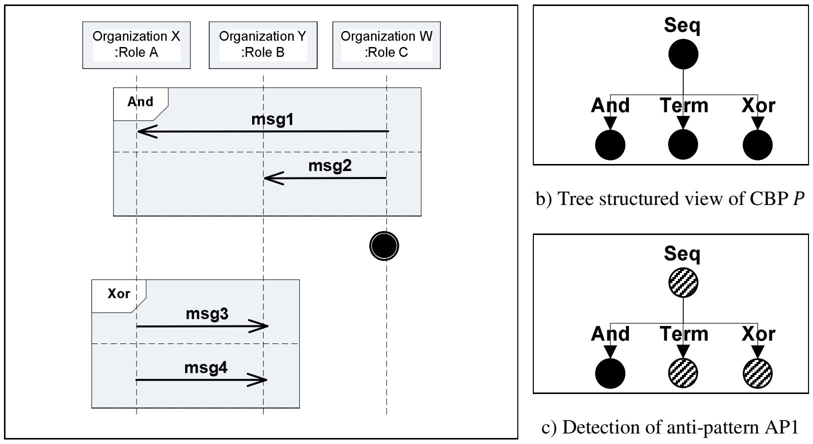
In the following sections each of these anti-patterns are described and exemplified. Since a model of UP-ColBPIP is essentially a tree of elements the anti-patterns make use of the function *Parent(e)*, which returns the parent in the tree structure of element *e*, and function *Ancestors(e)*, which returns a list of the ancestors of element *e*.

3.1 Deadlock in a Sequence

If a CBP is composed of an *Interaction Path* that has at least two elements: A and B, where element A is a *Termination,* and B is a direct successor of A, then the CBP is not sound. This leads to the anti-pattern AP1, which can be stated as follows:

Given a CBP *P* composed of the set of elements *E*, AP1 holds if there is a pair of elements $e_1, e_2 \in E$ where $e_1 \neq e_2$, $e_1 = Termination$, $Parent(e_1) = Parent(e_2) = Seq$ such that e_2 is a direct successor of e_1.

Figure 3.a shows an example of a CBP, where anti-pattern AP1 holds. Elements *And*, *Termination*, and *Xor* of this CBP are defined in sequence. This sequence of elements leads to a deadlock, since the CBP will terminate its execution before the element *Xor* can be executed. Figure 3.b shows the tree structured view of this CBP, which is a sequence with the elements *And*, *Term*, and *Xor*. Figure 3.c shows the detection of anti-pattern AP1 in the CBP, where nodes with oblique line texture are the elements that cause a deadlock.



b) Tree structured view of CBP *P*

c) Detection of anti-pattern AP1

a) CBP *P* defined with UP-ColBPIP

Fig. 3. Example of Anti-Pattern AP1

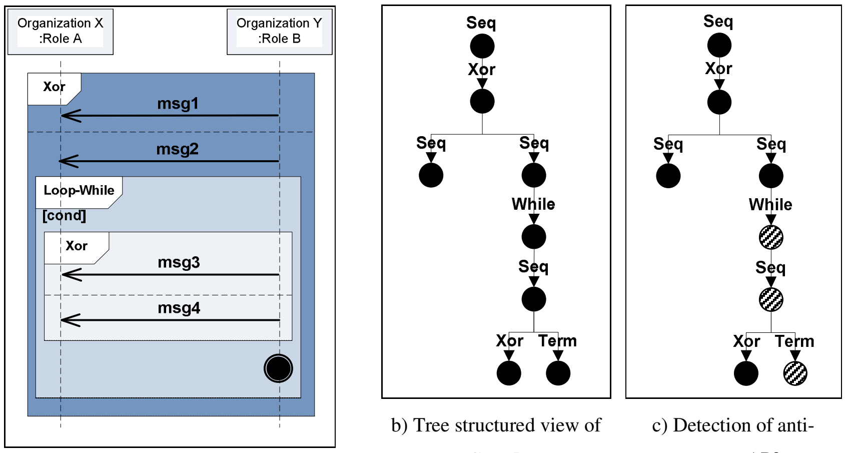
3.2 Deadlocks in Cycles

If a CBP is composed of a *Loop-While* or a *Loop-Until* and the interaction path which is part of such element contains a *Termination*, then the CBP is not sound. This leads to the anti-patterns AP2 and AP3, which can be stated as follows:

Given a CBP *P* composed of the set of elements *E*, AP2 holds if there are two elements $e_1,e_2 \in E$ where e_1*=Termination*, e_2*=Seq*, *Parent(*e_1*)=* e_2, and *Parent(*e_2*)=While*.

Given a CBP *P* composed of the set of elements *E*, AP3 holds if there are two elements $e_1,e_2 \in E$ where e_1*=Termination*, e_2*=Seq*, *Parent(*e_1*)=* e_2, and *Parent(*e_2*)=Loop-Until*.

Figure 4.a shows an example of a CBP, where anti-pattern AP2 holds. The *Loop-While* of this CBP is composed of the elements *Xor* and *Termination*. Since the CBP finishes its execution after the element *Termination* is reached, the element *Loop-While* will be executed at most once, which is not the expected behavior for a loop. Figure 4.b and 4.c show the tree structured view of this CBP and the detection of anti-pattern AP2.



a) CBP *P* defined with UP-ColBPIP

b) Tree structured view of CBP *P*

c) Detection of anti-pattern AP2

Fig. 4. Example of Anti-Pattern AP2

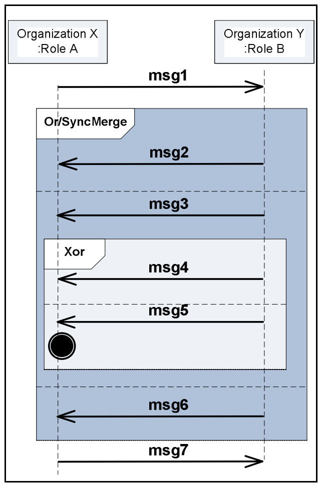
3.3 Deadlocks in Parallel Paths

If a CBP is composed of an element *And* or an *Or* and from any of the interaction paths which are part of such element it is possible to reach a *Termination*, then the CBP is not sound. This leads to the anti-patterns AP4 and AP5, which can be stated as follows:

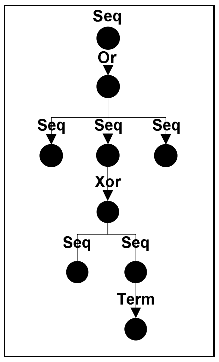
Given a CBP *P* composed of the set of elements *E*, AP4 holds if there are two elements $e_1,e_2 \in E$ where e_1*=Termination*, e_2*=Seq*, $e_2 \in$ *Ancestors(*e_1*)* and *Parent(*e_2*)=And*.

Given a CBP *P* composed of the set of elements *E*, AP5 holds if there are two elements $e_1,e_2 \in E$ where e_1*=Termination*, e_2*=Seq*, $e_2 \in$ *Ancestors(*e_1*)* and *Parent(*e_2*)=Or*.

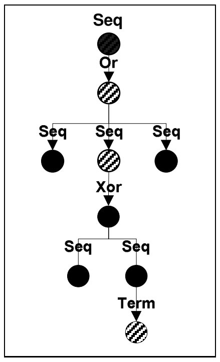
Figure 5.a shows an example of a CBP, where anti-pattern AP5 holds. In this CBP there is a *Xor* with a *Termination* in one of its interaction paths. The *Xor* is nested within an element *Or/SyncMerge*. If the execution of the CBP reaches the *Termination*, then the synchronization of the interaction paths of the element *Or* will not be reached. In addition, there could be business messages being sent even though the CBP has already finished its execution. Figure 5.b and 5.c show the tree structured view of this CBP and the detection of anti-pattern AP5.



a) CBP *P* defined with UP-ColBPIP



b) Tree structured view of CBP *P*



c) Detection of anti-pattern AP5

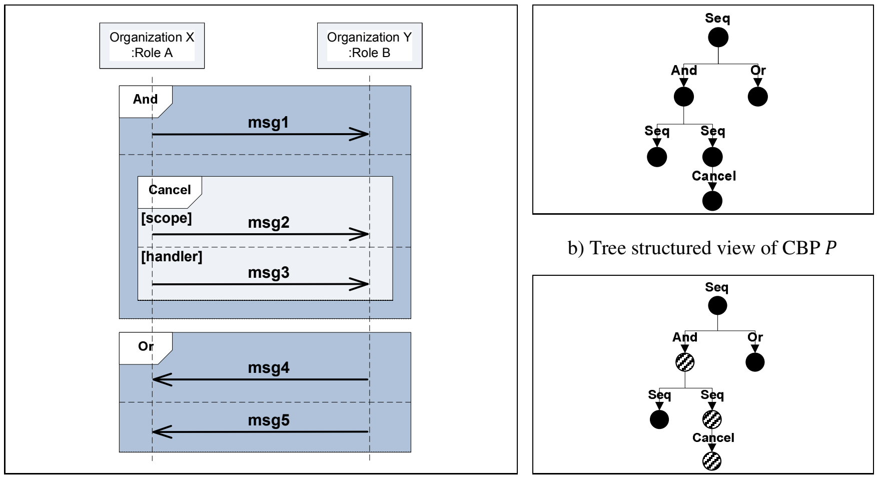
Fig. 5. Example of Anti-Pattern AP5

If a CBP is composed of an element *And* or an *Or* and from any of the interaction paths which are part of such element it is possible to reach a *Cancel*, then the CBP is not sound. This leads to the anti-patterns AP6 and AP7, which can be stated as follows:

Given a CBP *P* composed of the set of elements *E*, AP6 holds if there are two elements $e_1,e_2 \in E$ where e_1=*Cancel*, e_2=*Seq*, $e_2 \in$ *Ancestors*(e_1) and *Parent*(e_2)=*And*.

Given a CBP *P* composed of the set of elements *E*, AP7 holds if there are two elements $e_1,e_2 \in E$ where e_1= *Cancel*, e_2=*Seq*, $e_2 \in$ *Ancestors*(e_1) and *Parent*(e_2)=*Or*.

Figure 6.a shows an example of a CBP, where anti-pattern AP6 holds. In this CBP there is an *And* with a *Cancel* in one of its interaction paths. If the handler of the *Cancel* is executed, then the synchronization of the interaction paths of the element *And* will not be achieved. In addition, the business messages *msg1* could be sent even though the CBP has already finished its execution. Figure 6.b and 6.c show the tree structured view of this CBP and the detection of anti-pattern AP6.



b) Tree structured view of CBP *P*

a) CBP *P* defined with UP-ColBPIP c) Detection of anti-pattern AP6

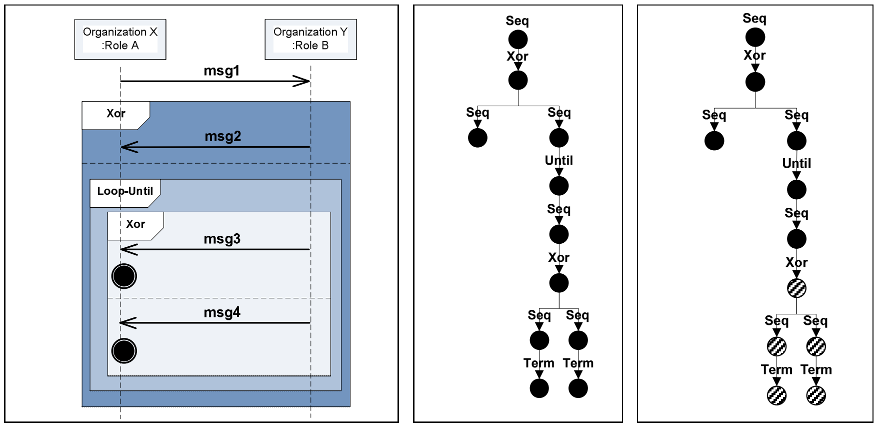
Fig. 6. Example of Anti-Pattern AP6

3.4 Deadlocks in Mutually Exclusive Paths

If a CBP is composed of an element *Xor*, and for each interaction path which can be reached from the *Xor* contains a *Termination*, then the CBP is not sound. This leads to the anti-pattern AP8, which can be stated as follows:

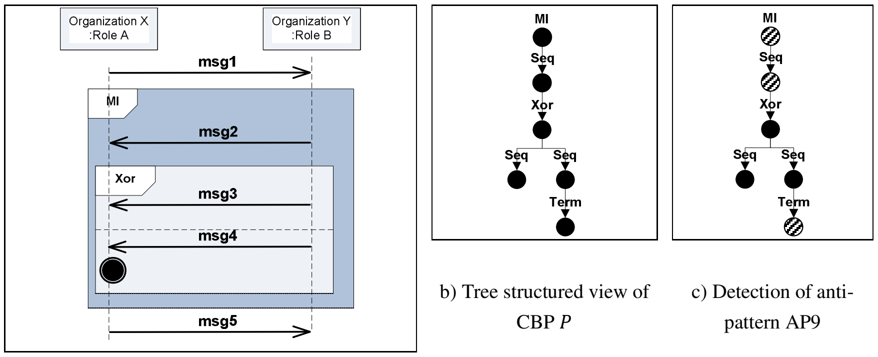
Given a CBP *P* composed of the set of elements *E*, AP8 holds if there is an element $e_1 \in E$ where $e_1 = Xor$, such that for each element s_i, where $e_1 \in Ancestors(s_i)$ and $s_i = Seq$, there is an element $e_i = Termination$ such that $Parent(e_i) = s_i$.

Figure 7.a shows an example of a CBP, where anti-pattern AP8 holds. In this CBP there is a *Xor* nested within a *Loop-Until*, where both interaction paths of the *Xor* have a *Termination*. In this example, the elements *Termination* cause a deadlock, and hence, the *Loop-Until* can be executed at most once, which is not the expected behavior for a loop. Figure 7.b and 7.c show the tree structured view of this CBP and the detection of anti-pattern AP8.



a) CBP *P* defined with UP-ColBPIP b) Tree structured view of CBP *P* c) Detection of anti-pattern AP8

Fig. 7. Example of Anti-Pattern AP8



b) Tree structured view of CBP *P* c) Detection of anti-pattern AP9

a) CBP *P* defined with UP-ColBPIP

Fig. 8. Example of Anti-Pattern AP9

3.5 Deadlocks in Paths with Multiple Instances

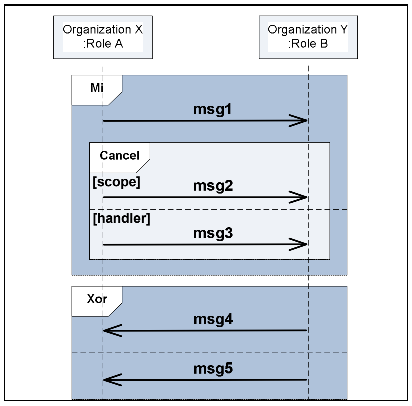
If a CBP is composed of an element *Multiple Instances* and from its interaction path it is possible to reach a *Termination* or a *Cancel*, then the CBP is not sound. This leads to the anti-patterns AP9 and AP10, which can be stated as follows:

Given a CBP *P* composed of the set of elements *E*, AP9 holds if there are two elements $e_1,e_2 \in E$ where e_1=*Termination*, e_2=*Seq*, $e_2 \in Ancestors(e_1)$, *Parent*(e_2)= *MultipleInstances*.

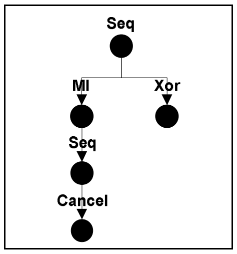
Given a CBP *P* composed of the set of elements *E*, AP10 holds if there are two elements $e_1,e_2 \in E$ where $e_1=Cancel$, $e_2=Seq$, $e_2 \in Ancestors(e_1)$, $Parent(e_2)= MultipleInstances$.

Figure 8.a shows an example of a CBP, where anti-pattern AP9 holds. In this CBP there is a *Xor* with a *Termination* in one of its interaction paths. The *Xor* is nested within an element *Multiple Instances*. If the execution of the CBP reaches the *Termination*, then the synchronization of the multiple instances will not be achieved. In addition, there could be running instances even though the CBP has already finished its execution. Figure 8.b and 8.c show the tree structured view of this CBP and the detection of anti-pattern AP9.

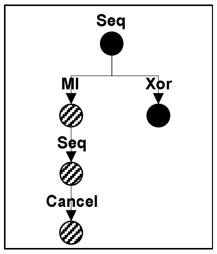
Figure 9.a shows an example of a CBP, where anti-pattern AP10 holds. In this CBP there is a *Multiple Instances* with a *Cancel* in its interaction path. If the handler of the *Cancel* is executed, then the synchronization of the multiple instances will not be achieved. In addition, there could be running instances even though the CBP has already finished its execution. Figure 9.b and 9.c show the tree structured view of this CBP and the detection of anti-pattern AP10.



a) CBP *P* defined with UP-ColBPIP



b) Tree structured view of CBP *P*



c) Detection of anti-pattern AP10

Fig. 9. Example of Anti-Pattern AP10

3.6 Case Study

As a case study, the Collaborative Demand Forecast CBP model of Figure 2 was verified with the proposed anti-patterns. The results returned by the tool determined that the process has a deadlock. The problem is that the behavioral semantics of the *And* establishes that all of the parallel paths must be synchronized. However, if a time event occurs within the scope of the *Cancel*, an exception will be raised and the synchronization of the *And* will not be achieved. Since the paths of the *And* are executed in parallel, there could be interactions in execution even though the CBP has reached an end state through the element *Cancel*. This problem is detected by anti-pattern

AP6. The solution to this problem is presented in Figure 10, where the element *And* is defined within the scope of the *Cancel*. This way, if an exception is raised, the scope of the *Cancel* will cover all of the paths of the *And*, avoiding a deadlock.

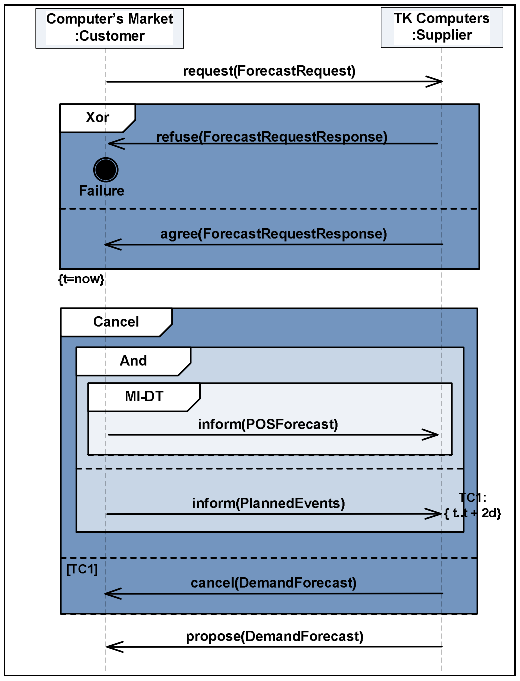


Fig. 10. Collaborative Demand Forecast CBP model free of deadlocks

4 Evaluation of UP-ColBPIP Anti-Patterns

In order to evaluate the verification approach based on anti-patterns, anti-patterns were implemented as an Eclipse plug-in [13] in a case tool for modeling CBPs. The tool can be accessed from http://code.google.com/p/upcolbpip-verification/. In addition, a repository of 880 UP-ColBPIP models was used.

The repository was automatically and systematically generated through an algorithm implemented in Java. The algorithm includes a set of generation rules, which considers all of the possible combinations of constructs that can be defined from the UP-ColBPIP's metamodel, in such a way to cover all the sound and unsound combinations of elements. Generation criteria consider number of elements per sequence, alternative parallel paths, exception handlers, and the size of models. Models of this repository are composed of the combination of any of the following constructs: *And*, *Xor*, *Loop-While, Loop-Until*, *Termination*, *Or/SyncMerge*, *Cancel*, *Exception* y *Multiple Instances*. Figure 11 shows information about the number of elements and interaction paths of the models. The number of elements of each model ranges from 1 to 120, whereas the number of interaction paths ranges from 0 to 100.

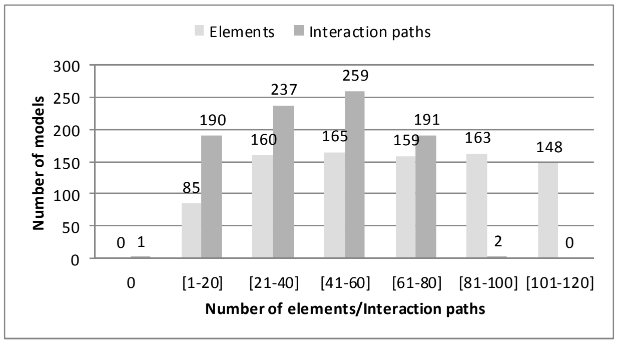


Fig. 11. Number of elements and interaction paths of CBPs in the repository

Based on the repository, an experiment was conducted to determine minimum, maximum, and average analysis time of the verification based on anti-patterns and the verification based on GI-Nets proposed in previous work [9, 10]. The experiment was performed in a PC with a processor Intel Core 2 duo 2.8 GHz and 8 GB of RAM.

A summary of obtained results are shown in Table 1. For the verification based on anti-patterns all models are tractable, whereas for the verification based on GI-Nets only 49,5% of models are tractable. Intractable models were not considered in the study. Response time of anti-patterns varies between 0,009 ms and 0,386 ms, with an average of 0,072 ms, whereas response time of GI-Nets varies between 1091 ms and 4993 ms, with an average of 2807 ms. No inconsistencies were found in the comparison of results obtained from both verification methods.

Table 1. Experimental results of the verification based on anti-patterns

		Anti-Patterns	GI-Nets
Tractable models (< 5000 [ms])		100 %	49,5%
Analysis time [ms]	Min.	0,009	1091
	Max.	0,386	4993
	Avg.	0,072	2807

The verification based on anti-patterns obtained faster response times than the method based on GI-Nets. This is essentially due to the method based on GI-Nets carries out an exploration of the state space of the GI-Net that formalizes a CBP model, whereas the method based on anti-patterns is focused on finding a given set of combinations of constructs within a CBP model. Results also show that the method based on anti-patterns obtained the same set of results than the method based on GI-Nets.

5 Related Work

Different approaches were proposed to verify the behavior of CBPs [14, 15, 16]. However, most of these approaches rely on state space exploration which may lead to the state space explosion problem [11], which negatively affects the performance. In addition, these approaches do not support complex constructs for advanced synchronization, multiple instances, and exception management. In this work, we proposed anti-patterns for simple and complex control flow constructs achieving response times of less than half a millisecond, which results much faster in comparison to results obtained in a verification proposed in previous work [9, 10].

In the context of intra-organizational processes different approaches were proposed to verify business processes based on anti-patterns [17, 18, 19]. However, none of them were applied to CBPs. In [17], authors extracted typical modeling errors from process repositories, and generalized them into anti-patterns. In [18], authors proposed an approach for detecting control flow errors in business processes based on BPMN-Q, which is a visual query language for defining anti-patterns. In [19], authors presented an approach to detect anti-patterns in EPC models. However, the approach may return erroneous results for some type of models. According to experimental results, the anti-patterns we proposed in this work do not return erroneous results.

6 Conclusions and Future Work

In this work, an approach for the verification of the behavior of CBPs was presented. The approach is based on behavioral anti-patterns and enables the detection of deadlocks in the control flow of CBPs which may have simple and complex control flow constructs. To this aim, 10 anti-patterns for CBPs were defined considering the behavioral semantics of simple and complex control flow constructs. For each anti-pattern a rule defining the anti-pattern and an example were provided.

Anti-patterns were implemented as an Eclipse plug-in in a case tool for modeling CBPs. This tool, together with a repository of 880 UP-ColBPIP models, was used to evaluate the proposed anti-patterns. With the use of anti-patterns, the verification of CBPs is simplified to a pattern matching technique, which leads to an important performance improvement. Results show that the verification of these models can be performed, in the worst case, in 0,386 milliseconds. This means that CBPs can be verified in less than half a millisecond, even those with complex control flow constructs. Results also show that the verification based on anti-patterns is as accurate as the verification based on GI-Nets, and clearly improves the performance of the latter. This holds even for models with complex constructs and having more than 120 elements, where such models are intractable for the verification method based on GI-Nets.

However, at the moment there is no systematic method to discover every possible behavioral anti-pattern of a given language. Currently, anti-patterns are discovered and specified from well-known problems described in current literature, or from the verification and inspection of repositories. This implies that a verification method using such anti-patterns may not be able to detect, for instance, every possible situation leading to a deadlock or lack of synchronization in CBPs. Future work is concerned with defining an approach to systematically discover all of the anti-patterns of a process language.

References

1. Villarreal, P., Salomone, H., Chiotti, O.: "Modeling and specifications of collaborative business processes using a MDA approach and a UML profile". Enterprise modeling and computing with UML. Idea Group Inc (2007) 13-45.
2. Villarreal, P.D., Lazarte, I., Roa, J., Chiotti, O.: "A Modeling Approach for Collaborative Business Processes Based on the UP-ColBPIP Language". In: Business Process Management Workshops. Volume 43. Springer (2010) 318-329
3. Business Process Model and Notation (BPMN) version 2.0, Object Management Group, Inc. (OMG), <www.omg.org/spec/BPMN/2.0/>.
4. Web Services Choreography Description Language Version 1.0, <http://www.w3.org/TR/ws-cdl-10/>.
5. Huemer, C, Liegl, P, Motal, T, Schuster, R & Zapletal, M 2008, "The development process of the UN/CEFACT modeling methodology", Proceedings of the 10th international conference on Electronic commerce, ACM, Innsbruck, Austria.
6. Fahland, D., Favre, C., Jobstmann, B., Koehler, J., Lohmann, N., Völzer, H., Wolf, K.: "Instantaneous Soundness Checking of Industrial Business Process Models". In Dayal, U., Eder, J., Koehler, J., Reijers, H., eds. : Business Process Management, vol. 5701, pp.278-293 (2009)
7. van der Aalst, W 1998, "The Application of Petri Nets to Workflow Management", Journal of Circuits, Systems and Computers, vol 08, no. 01, pp. 21-66
8. van der Aalst, W., van Hee, K., ter Hofstede, A., Sidorova, N., Verbeek, H., Voorhoeve, M., Wynn, M.: "Soundness of workflow nets: classiffication, decidability, and analysis". Formal Aspects of Computing, 1-31, (2010)
9. Roa, J., Chiotti, O., Villarreal, P.: "A verification method for collaborative business processes", in: BPM 2011 Workshops, Part I, Springer, Lecture Notes in Business Information Processing, vol. 099, (2012), 293-305.
10. Roa, J., Chiotti, O., Villarreal, P. "Behavior Alignment and Control Flow Verification of Process and Service Choreographies". Journal of Universal Comp. Science, vol. 18, no. 17, pp. 2383-2406, 2012.
11. Valmari, A 1998, "The state explosion problem", in W Reisig, G Rozenberg (eds.), Lectures on Petri Nets I: Basic Models, Springer Berlin Heidelberg, Dagstuhl, Germany.
12. Eclipse Platform, accessed April 2014, <http://www.eclipse.org>.
13. Steinberg, D, Budinsky, F, Paternostro, M & Merks, E 2008, EMF: Eclipse Modeling Framework, 2nd edn, Addison-Wesley Professional.
14. Van der Aalst, WMP 1998, "Modeling and analyzing interorganizational workflows", Proceedings of the 1998 Int. Conf. on Application of Conc. to System Design, IEEE Computer Society, Fukushima.
15. Stuit, M & Szirbik, N 2009, "Towards agent-based modeling and verification of collaborative business processes: an approach centered on interactions and behaviors". International journal of cooperative information systems, vol 18, no. 03n04, pp. 423-479.
16. Norta, A & Eshuis, R 2010, "Specification and verification of harmonized business-process collaborations". Information Systems Frontiers, vol 12, no. 4, pp. 457-479.
17. Koehler, J & Vanhatalo, J 2007, "Process anti-patterns: How to avoid the common traps of business process modeling", IBM WebSphere Developer Technical Journal, vol 10, no. 2.
18. Laue, R & Awad, A 2010, "Visualization of Business Process Modeling Anti Patterns", Electronic Communications of the EASST, vol 25.
19. Han, Z, Gong, P, Zhang, L, Ling, J & Huang, W 2013, "Definition and Detection of Control-Flow Anti-patterns in Process Models", IEEE 37th Annual Computer Software and Applications Conference Workshops (COMPSACW), 2013 , IEEE, Kyoto.
20. Roa, J. "Verificación y alineación de procesos de negocio colaborativos". Tesis doctoral. Universidad Tecnológica Nacional, Facultad Regional Santa Fe. 2014.