International Journal of Computer Applications (0975 8887)
Volume 94 - No. 5, May 2014

A Context-based Approach to Semantic Correctness

Priso Essawe-Ndedi
Higher Teacher's Training College
University of Yaounde 1 - Cameroon

Marcel Fouda-Ndjodo
Higher Teacher's Training College
University of Yaounde 1 - Cameroon

ABSTRACT

Ly, Rinderle and Dadam [13] have worked on process semantic correctness and introduced the concept of semantic contraints (mutual exclusion and dependency) in order to introduce semantic concerns in process modeling. In this paper, we attempt to refine their work by proposing a solution to two kinds of problems generated by semantic constraints as they initially defined it: how to maintain the constraints set when a new task is introduced in a model? How to allow mutual exclusive tasks to coexist in the same process via the desactivation of the cause of the mutual exclusion by a third party. Our solution takes advantage of a context-based business process modeling approach presented in [11].

Keywords:

Business Process, Semantic correctness, Conflict, Dependency

1. INTRODUCTION

Research on business processes modeling has extensively studied the structural aspects of business processes. Most business process modeling languages are graph-based (Workflow nets, EPC, BPMN), and emphasis has been put on structural concerns such as verifying if all jobs terminate properly, if tasks are properly synchronized or if there are subprocedures that are never used[1]. Petri nets, and more precisely the sub-type called Workflow nets have been instrumental in getting these results, at least for the formal aspects. There is an extensive literature about the structural and the dynamic properties of business process models, particularly based on Petri nets. This sum of knowledge on Petri nets is in turn used to formally assess the properties of some other languages which are more connected to the industry (EPC[3], BPMN[9]).
However, less has been done to evaluate the semantic correctness of business processes. An attempt has been made by Ly et al [13], based on semantic constraints (mutual exclusion and dependency). Their work has made possible the expression of global constraints, overcoming the barrier imposed on Petri-net based token languages [7] by the locality principle which states that the firing of a transition only affects the status of its neighbor transitions. From this principle, it appears that if two non adjacent tasks of a process are in mutual exclusion, Petri-net techniques are ill-suited to model such a situation. As a solution to this problem, Ly et al. propose the explicit definition of the set of constraints involved in the process, a kind of constraint layer, which can serve for verification, processing and documentation purposes.
Another unsolved issue related to the semantics of business process tasks, is that of the definition of the role of a task in a business process. In [6], Davenport and Short defined a business process as a set of logically related tasks performed to achieve a defined business outcome that we call the goal of the business process. While this definition can be found consensual, the different business process modeling languages that have been designed later did not offer any means to specify the actual role of a task in a process. Does it directly contribute to the assigned goal? If yes, then what is exactly its action? If no, does it contribute indirectly by making possible the execution of another task? Without the answer to these questions, it is difficult to reliably replace a task by another, because the consequences of such a change are unpredictable.
In this paper, we use a context-based business process modeling approach (see [11] for details) to improve the concept of business process semantic correctness. We refine the notion of mutual exclusion expressed in [13] by proposing three kinds of such situations: conflict, incompatibility, and antagonism. Each of these properties reveals a special kind of mutual exclusion. The contribution of this paper is three-fold: we propose a simple mechanism to express semantic constraints where constraints are not hard-coded, and hence are more suitable for change handling. The second important contribution is the ability to use in the same process, two mutual exclusive tasks, provided that there exists a third task in-between which lifts the incompatibility, and so allows the second task to execute. Finally, we refine the dependency relations between tasks by offering a means to express what is precisely the contribution of a task t for the execution of a subsequent task t_k or for the satisfaction of a goal g instead of merely stating that t_k depends on t.
This paper is organized as follows: in section two, the notion of semantic correctness is recalled. In section 3, the context-based modeling approach we used is presented, and it is shown how it handles semantic contraints. In section 4, we present how this model improves the management of semantic constraints. The related works in section 5 are followed by the conclusion.

2. SEMANTIC CORRECTNESS

For more than a decade now, numerous business process modeling languages have been created, supported by several software platforms. Those platforms are very effective for structural correctness analysis. It is also crucial to be able to integrate semantic knowlegde, particularly to avoid semantic conflicts within the process change framework. But the fact is that very few support semantic correctness [13, 22]. Current approaches mainly focus on structural aspects or have a notion of semantic correctness turned towards graph reachability and graph coverage, to ensure that the final node of a process model is reachable according to the process definition, and that all the transitions of the process graph are used [1].

2.1 Semantic constraints

In this paper, we focus on the definition of semantic correctness proposed by Ly et al. in [13]. They propose a context-based approach to semantic correctness. In order to express semantic correctness, each task is associated with constraints. A constraint can be of type mutual exclusion or dependency.
Formally, A being a set of activities, a semantic constraint c is defined as a tuple $(type,\ source,\ target,\ position, userDefined)$ whereas:
– $type \in Exclusion,\ Dependency$
– $source,\ target \in A,\ source \neq target$
– $position \in pre, post, notSpecified$
–$userDefined$ is a user-defined parameter
The parameter $type$ denotes whether the semantic constraint is a mutual exclusion constraint or a dependency constraint. The second parameter $source$ denotes the source activity the constraint refers to. The parameter $target$ denotes the target activity related to the source activity. Parameter $position$ specifies the order the source and target activity are to be related to each other within the process (e.g., the surgery depends on the preparation of blood bottles and the bottles have to be prepared before (pre) the surgery). The last parameter $userDefined$ can be used for several purposes, for instance for additionally describing the constraint.
In a medical context, we can express the fact that two drugs, $Aspirin$ and $Marcumar$ are incompatible with the following constraint:
$(Exclusion, TakeAspirin, TakeMarcumar,$
$notSpecified, "Sample1")$
This constraint expresses the fact that a patient who has been administered $Aspirin$ should not be given $Marcumar$. This constraint is not linked to the order in which those drugs are taken, reason why the position is set to $notSpecified.$
The second type of constraint is a dependency constraint. For example, before a surgery is performed, it may be necessary to prepare blood bottles. We get the following constraint:
$(Dependency, PerformSurgery, PrepareBloodBottles,$
$post, "Sample2")$
With this constraint, a process trace with the task $PerformSurgery$ without being preceded by the task$PrepareBloodBottles$ is semantically incorrect.
A business process is said to be semantically correct if all the constraints defined in the model are respected. Any time a change is enforced, the constraints should be verified to guarantee that the new process is still semantically correct. With the definition of constraints, it is possible to automatically check the semantic correctness of a process. In a semi-automated mode, it is also possible to warn the user of the possibility of a semantic problem.

2.2 Raised issues

The semantic constraints as described above contribute to the improvement of processes at the semantic level. Semantic constraints are no longer implicit in the mind of the designer, but explicit, documented and verifiable. Nevertheless, there are still some issues not solved by these semantic constraints as they are defined.

2.2.1 Hard-coded constraints. According to Ly et al., a constraint is established between two well-identified tasks. The reason why two tasks are mutually exclusive or dependent is not known with the given constraint presentation. As a consequence, deduction operation are very hazardous. A task t_1 is in mutual exclusion with a task t_2, and t_2 is in mutual exclusion with t_3. Is is possible to deduce something about t_1 and t_3? We are afraid the answer is no. We do not know anything about the reason underlying these exclusions. A human expertise is always necessary to create constraints. Even if two tasks $TakeMarcumar$ and $TakePhenprocoumon$ are identical ($Marcumar$ being the commercial name of $Phenprocoumon$), a computer system cannot assist in replicating the constraints involving one task on the other.
When adding a new task to the process, the elicitation of the constraints caused by the new task is very complex. It causes the whole process to be re-inspected. Each time a new task is to be added, all the pre-existing tasks should be inspected in order to verify if the new task is in mutual exclusion with another task. This has to be done manually according to the knowledge we have on the tasks of the process. It makes any change process very sensitive and very exhausting.

2.2.2 Impossibility to interleave mutual exclusion and dependency. According to Ly et al., when a mutual exclusion constraint is defined, the two tasks involved in the constraint should not appear together in an excution trace. Although it may seem normal, it can also appear in some situations as too restrictive. Let us consider their own example of drug incompatibility. A drug called Marcumar is not compatible with another drug called Aspirin. The mutual exclusion constraint fits well for this situation. Let us assume that another drug called AntiMar annihilates the effect of Marcumar which makes it incompatible with Aspirin. In such a case, it is safe to take Marcumar, then AntiMar, and finally Aspirin. Hence, the path Marcumar-AntiMar-Aspirin becomes semantically correct. However, according to the model proposed by Ly et al., this last option remains semantically incorrect, because any trace which contains Marcumar should not contain Aspirin.

3. A CONTEXT-BASED BUSINESS PROCESS MODEL

A business process modeling approach has been proposed in [11]. The essential feature of this approach is context-awarenes. We will use this model to propose a refinement of the notion of semantic correctness. The term context-awareness was initially coined by Schilit and Theimer [17] to describe the ability of applications to discover and react to changes in the environment they are situated in. Context is either paraphrased in the literature by means of synonyms, such as environment or situation [8].
The model we use is built around the notion of environment. The environment is defined in this model as a set of context variables. As it is not possible to capture the entire context of an organization [21], we restrict ourselves to the part of the real world that is of interest for the business processes to design. Every characteristic of the real word which is relevant for the processes to design is captured through context variables in the form of boolean objects that we call observers.
The environment provides us with a platform where all of the task components in a process model must share information with each other and work together in order to achieve the objective of the process [18].

DEFINITION 1. *Environments*
An environment ξ is a tuple $< \theta, S, val >$ where:
- θ is a non empty set whose elements are called observers;
- S is a non empty set whose elements are called states $(\theta \cap S = \emptyset)$;
- $val : \theta \rightarrow (S \rightarrow Bool)$ is a function which gives the values of observers in the different states.

When the context is clear, we write $s(o)$ for $val(o)(s)$ with the intuitive meaning that $s(o)$ is the value of the observer o in the state s. Given an environment ξ, an observation tells us if a condition

over a set of observers is satisfied or not. An observation therefore has a positive part and a negative part. The positive part of an observation is the set of the observers whose value is expected to be $true$, while the negative part is the set of observers whose value is expected to be $false$.
The environment depicts the whole context. At a given moment in time, we are focused on a subset of the environment. We make an observation which only concerns a few observers. On these few observers, we expect some to be $true$ and some others to be $false$. The observation is satisfied if our expectations are met. Formally, we get the following definition:

DEFINITION 2. *Observations*
Let $\xi =< \theta, S, val >$ be an environment. An observation on ξ is a couple $< A, B >$ where : $A \subseteq \theta$, $B \subseteq \theta$ and $A \cap B = \emptyset$.

A is the set of observers required to be $true$, and B is the set of observers required to be $false$. The observation $obs =< A, B >$ is satisfied in a state s iff $(\forall o \in A, s(o))$ and $(\forall o \in B, \neg s(o))$.

Let $obs =< A, B >$ be an observation, A (resp. B), also denoted $P(obs)$ (resp. $M(obs)$), is called the positive part (resp. negative part) of obs.

The environment constitutes the context, but this context is not static. It is in constant mutations, and we need processors to enforce these mutations. This role is played in our model by tasks. Tasks are the entities which have the capacity of modifying the context. A task modifies the value of one or many observers. The action of a task is explicit, and is limited to the observers it modifies. But for a task to act, its precondition has to satisfied first. In short, we define a task as an atomic unit of work which can be triggered under specific requirements (pre-condition) to produce a predictible result (post-condition).

DEFINITION 3. *Tasks*
Let $\xi =< \theta, S, val >$ be an environment. A task on ξ is a triple $< t, ec, action >$ where t is the name or identifier of the task, ec is an observation specifying its pre-condition, and $action$ is an observation specifying its post-condition.

In the remainder of this paper, the execution condition (resp. post-condition) of the task $< t, ec, action >$ is denoted $ec(t)$ (resp. $action(t)$). In the same vein, $P(action(t))$ (resp. $M(action(t))$) is denoted $P(t)$ (resp. $M(t)$).

The execution of t in the state s moves the environment ξ into the state s' such that:

$$\begin{cases} \forall o \in P(t), s'(o) \\ \forall o \in M(t), \neg s'(o) \\ \forall o \in \theta \setminus (P(t)) \cup M(t)), s'(o) = s(o) \end{cases}$$

In a business process, tasks are not isolated. According to Davenport and Short , a business process is as a set of logically-related tasks performed to achieve a defined business outcome.

DEFINITION 4. *Business process*
Given an environment ξ, a Business Process is a tuple $BP =< \theta, T, t_{init}, f, g >$ where:
- θ is a set of observers over ξ,
- T is a set of tasks on ξ,
- t_{init} is the initial task,
- g is a distinguished observation called the goal of the process,
- $f : T \to 2^T$ is a function which, for every task, gives the names of the tasks that can be executed right after it.

Our definition of a business process is conform to the idea of Davenport and Short, but it also context-oriented. The outcome of the business process (the goal) is an observation over the environment. In other words, we can observe on the environment if the business outcome is satisfied. The tasks are ordered by the follow function f, starting by the special task t_{init}. For each task, the follow function indicates which tasks can follow.

4. OUR IMPLEMENTATION OF SEMANTIC CONSTRAINTS

The model developped in [11] and presented above fits well with the semantic correctness defined by Ly et al[13]. In the following lines, we show how we use it to capture the different constraints proposed by Ly et al., and we propose some refinements to each type of constraint (mutual exclusion and dependency).

4.1 Mutual exclusion

A mutual exclusion constraint concerning two tasks t_1 and t_2 means that no execution trace should contain both t_1 and t_2. This can happen for several reasons. Both tasks may act on the same subject, but differently. For example, t_1 switches the light on, while t_2 switches it off. We call this kind of mutual exclusion conflict. It can also happen that t_1 needs the light to be on for its execution, while t_2 needs the same light to be off. We will say that t_1 and t_2 are incompatible. A third case is the situation where t_1 switches the light on, while t_2 requires that light to be off. In this case, we say that t_1 and t_2 are antagonist.
From the model described above, we derive a formalization of these three types of mutual exclusion.

Conflict. Two tasks t_1 and t_2 are in conflict if there is an observer o such that belongs to the positive part of the action of t_1, and to the negative part of the action of t_2. Formally, $o \in P(t_1) \cap M(t_2)$. In such a situation, t_1 and t_2 are in mutual exclusion on o. t_1 switches o to $true$ while t_2 switches the same observer o to $false$. If for example o tells if a loan is accepted, t_1 will say yes while t_2 says no.

Incompatibility. Two tasks t_1 and t_2 are incompatible if there is an observer o such that $o \in P(ec(t_1)) \cap M(ec(t_2))$.
In such a situation, t_1 and t_2 are in mutual exclusion on o. t_1 requires o to be $true$ while the same o is required to be $false$ by t_2. If for example o tells if a loan is accepted, t_1 requires the loan to be accepted, while t_2 requires the loan not to be accepted.

Antagonism. Two tasks t_1 and t_2 are antagonist if there is an observer o such that $o \in P(t_1) \cap M(ec(t_2))$.
In such a situation, t_1 and t_2 are in mutual exclusion on o. t_1 sets o to $true$ while the same o is required to be $false$ by t_2. If for example o tells if a loan is accepted, t_1 says that the loan to be accepted, while t_2 requires the loan not to be accepted.

4.2 Dependency

A dependency constraint concerning two tasks means that one task requires the other. For this constraint, we make a difference between two situations. A task t_1 may require another t_2 in order to have its execution condition satisfied. The action of t_2 contribute to the activation of t_1. This is a none side dependency, because in this case, t_2 does not depend on t_1. We call it an execution condition dependency because t_1 needs t_2 to be executed in other to have its execution condition satisfied.

Table 1. Tasks

Task name	Precondition		Postcondition	
	True	False	True	False
t_1			o_1	o_3
t_2				o_2

Table 2. Goal

OBSERVATION: g	
True	False
o_1	o_2
	o_3

A task t_1 may also require another task t_2 in other to achieve a goal g together, each task achieving just a part of the goal. In this second case, it is a mutual dependency. It would even be more accurate to say that the goal depends on t_1 and t_2. We call this dependency a goal satisfaction dependency.

Execution condition dependency. If two tasks t_1 and t_2 are such that there is an observer o such that $o \in P(ec(t_1)) \cap P(t_2)$, then o belongs to the execution condition of t_1, and it also belongs to the action of t_2. In other words, the action of t_2 makes it possible to execute t_1.
In such conditions, there is an execution condition dependency between t_1 and t_2. t_2 activates o for t_1, and allows t_1 to be executed, because if o is not set to $true$, then t_1 will not be able to execute.
In this situation, it is obvious that in an execution trace (or in a path), t_1 should appear after t_2. So according to Ly et al. terminology, t_1 will be the source, t_2 the target, with the position $post$, because in this case, t_1 appears after t_2.

Goal satisfaction dependency. If a goal is such that no task can satisfy it alone, each task satisfying just a part of the goal, then we say that the tasks contributing to the satisfaction of the goal have mutual dependency constraints for the satisfaction of the goal.
Let us consider two tasks t_1 and t_2 such that t_1 sets an observer o_1 to $true$ and another observer o_3 to $false$, while t_2 sets an observer o_2 to $false$. The pre-condition of both tasks is void (see table 1).
We need to achieve a goal g which requires the observer o_1 to be $true$ and the observers o_2 and o_3 to be $false$ (see table 2).
We see that neither t_1 nor t_2 can satisfy the goal g. t_1 depends on t_2 just like t_2 depends on t_1 for the satisfaction of the goal g.
According to Ly et al. terminology, t_1 depends on t_2 with the position $NotSpecified$, because in this case, t_1 may appear either before or after t_2.

5. IMPROVEMENTS ON THE NOTION OF SEMANTIC CORRECTNESS

We have shown in the lines above that our model captures the notion of semantic constraint as defined in [13]. A process model is said to be semantically correct if all the semantic constraints are respected. In this section, we will show that mutual exclusion and dependency, as defined in [13], fail to capture some particular, but likely situations. Moreover, the constraints are ill-suited for change because they are hard-coded. Any change introduced in the model will require a complete re-inspection of the model to make sure it is still semantically correct.

5.1 Avoiding Hard-coded constraints

According to Ly et al., a constraint is established between two well-identified tasks, and existing constraints can merely be used to deduce new ones. With our model, the constraints are expressed using observers which are context variables. For each task, we define what it expects from the environment and what it produces on the environment. When a task is added or removed, its impact can be automatically calculated. Thus, the change process can be efficiently computer-assisted.
To illustrate this situation, let us consider a medical environment where a restricted set of drugs is used. The different constraints between those drugs are documented using Ly et al. constraints. When adding a new drug to this set, the interactions between the new drug and each of the existing ones must be carefully and manually investigated. Our knowledge of each existing drug should be used to decide if it is compatible with the new one. The result of this process is the definition of a new set of dependencies.
With our model, this investigation can be automated. A part of our knowledge on each drug is already documented in the form of preconditions and actions. Let us assume, for the sake of simplicity, that the new drug's effect is to increase the blood pressure. The action of the new drug will be setting an observer $lowBloodPressure$ to $false$. We can automatically look for incompatibilities by searching for any existing drug, which in its precondition requires $lowBloodPressure$ to be $true$.
Formally, $\{lowBloodPressure\} \in M(takeNewDrug)$, and the task $takeNewDrug$ is incompatible with any task t such that the $\{lowBloodPressure\} \in P(ec(t))$.
Given that the precondition and the action of all the tasks are completly defined, the updated set of constraints can be automatically obtained after a change by checking the execution condition and the action of all existing tasks against that of the new task.

5.2 Interleaving mutual exclusion and dependency constraints

According to Ly et al., when a mutual exclusion constraint is defined, the two tasks involved in the constraint should not appear together in an excution trace. This definition of mutual exclusion causes some correct processes to be marked as incorrect. Let us consider again the Marcumar and Aspirin incompatibility example. Let us assume that another drug called AntiMar annihilates the effect of Marcumar which makes it incompatible with Aspirin. In such a case, it is safe to take Marcumar, then AntiMar, and finally Aspirin. Hence, the path Marcumar-AntiMar-Aspirin becomes semantically correct. However, according to the model proposed by Ly et al., this last option remains semantically incorrect, because any trace which contains Marcumar should not contain Aspirin.
With our model, this limitation is lifted. Marcumar and Aspirin are antagonist (a particular form of mutual exclusion as explained above). Once Marcumar is taken, an observer o is set $true$, and that same observer is required to be $false$ before being allowed to take Aspirin (o is part of the execution condition of taking Aspirin). In this configuration, it is not possible to take Aspirin after Marcumar. But taking the drug AntiMar sets the observer o to $false$, making it possible again to take Aspirin. Our model clearly exhibits that the sequence $Marcumar - Aspirin$ is wrong, while $Marcumar - AntiMar - Aspirin$ is correct.

5.3 Summary of the improvements

We summarize the improvements on the notion of semantic correctness introduced by our implementation of semantic constraints in the table 3:

Table 3. Summary of the improvements

	Ly's semantic constraints	Our implementation
Precision	The causes of the existence of constraints are not known	The constraints originate from the pre/post conditions of tasks
Change handling	After a change (adding/removing a task), the tasks should be manually re-inspected to decide if new constraints should be defined or if existing some existing constraints should be removed	After a change, the set of constraints can be generated from the pre and post conditions of tasks
Flexibility	when a mutual exclusion constraint is defined, the two tasks involved in the constraint should not appear together in an excution trace	two tasks in mutual exclusion can appear in the same execution trace if another task executed in-between disables the cause of the mutual exclusion

6. RELATED WORKS

This paper is fundamentally based on two previous ones: [11]where the formal model we use here has been defined, and [13] where we picked the notion of semantic correctness.
In [11], a formal business process model is defined. This model incorporates the notion of environment which was first used for the definition of business processes in [5]. It refines it by proposing a more computable task model. While in [5] a task was any operator that changed the state of the environment, in [11], it is required in the definition of a task to indicate on which observers the task actually has an effect. As a consequence, the action of a task is predictible, regardless of the state of the environment. The modeling driver for both [5] and [11] is the emphasis put on context. The term context-awareness was initially coined by Schilit and Theimer [17] to describe the ability of applications to discover and react to changes in the environment they are situated in. Context is either paraphrased by means of synonyms, such as environment or situation [8]. The term environment has been used in [5] and kept in [11]. The emphasis on context can be very helpful to handle change situations. In [15], Regev et al. defined business process flexibility as “the capability to implement changes in the business process type and instances by changing only those parts that need to be changed and keeping other parts stable”[15]. This understanding implies that business processes are aware of the environment they are designed for, i.e. it has to be known what parts of the process have to be changed and which parts are to be kept stable, and it has to be noticed when a change is necessary. This in turn suggests taking the environment, where a change occurs, into account when designing business processes [12].
The second milestone of this paper is the work of Ly et al.[13] who proposed a system of semantic constraints for a business process model. As we mentionned in the above sections, they propose two types of constraints: mutual exclusion and dependency constraints. A process is said to be semantically correct if all the listed constraints are respected. This constitutes an improvement on previous business process modeling approaches where constraints are implicit, and if enforced, difficult to maintain. The term semantic correctness is also used in [4] to denote the notion of soundness as defined in [1], by opposition to syntactic correctness. Syntactic errors are all errors that can be detected without any knowledge of the application domain, i.e. all constructs violating universal, domain independent requirements. A process is said to be sound if it has no deadlocks and terminates properly. Proper termination means that the state is reached with a single token in the sink place and after termination there are no dangling references [2]. This definition of semantic correctness associated to soundness has different goal than ours, and the term semantic correctness used in this paper is not linked to their definition, but to Ly's one. In [14], semantic correctness is also studied. Semantic concerns are added on top of workflow nets to obtain semantic workflow nets. A process is said to be semantically conformant to a process specification if the precondition of the process is always fullled in the precondition of the specification, if the effect of the process fulfills the effect of the specification, and if each activity in the process is actually invokable when it can be invoked. Schaffner et al.[16] also confirm the link between semantic correctness and the satisfaction of preconditions. They add that each task of the process should be relevant (its outputs is consumed by a successor operation) and should not be redundant. These needs are explicitly captured in our model with observers that are set by a task in a value required later in the precondition of another task.

7. CONCLUSION

This work has been an attempt to refine the notion of semantic correctness defined by Ly, Rinderle and Dadam[13], using the business process modeling approach defined in [11], where tasks are defined by their effect on the environment (defined as a set of boolean context variables called observers). A process is said to be semantically correct if all the mutual exclusion constraints and dependency constraints are satisfied. We introduced three kinds of mutual exclusions: conflict, incompatibility, and antagonism. Two tasks are conflicting if they produce different effect on the same observer. They are incompatible if for their execution, they require opposite values from the same observer. They are antagonist if the action of one task sets an observer to a value opposed to what is required by the other.
As a result of our approach, we have been able to express constraints on the environment, rather than expresing them directly between tasks. By so-doing, we have shown that the constraint set can be automatically updated after a change (adding, modifying or deleting a task). Another result is the possibility to use in the same process, two mutual exclusive tasks, provided that there exists a third task in-between which lifts the incompatibility, and so allows the second task to execute.
The results obtained in this paper can be used to improve the change handling process and move towards systems which can automatically adapt to changes.

8. REFERENCES

[1] W.M.P. van der Aalst, The application of Petri nets to workflow management, The Journal of Circuits, Systems and Computers (1998), pp. 21-66

[2] van der Aalst, W. M., & Jablonski, S. (2000). Dealing with workflow change: identification of issues and solutions. Computer systems science and engineering, 15(5), 267-276

[3] W.M.P. van der Aalst and J. Desel and E. Kindler, On the semantics of EPCs: A vicious circle (2002)

[4] W.M.P. van der Aalst and M. Dumas and F. Gottschalk and A.H. ter Hofstede and M. La Rosa and J. Mendling, Preserv-

ing correctness during business process model configuration, Formal Aspects of Computing (2010), pp. 459-482

[5] R. Atsa-Etoundi, A domain engineering approach for multi-perspectives Workflow modelling, University of Yaounde I - Cameroon (2004)

[6] T. Davenport and J.E. Short. The new industrial engineering: information technology and business process redesign. Sloan Management Review, 31(4), pp.11–27, 1990

[7] J. Desel and G. Juhs, “What Is a Petri Net?” Informal Answers for the Informed Reader, In Unifying Petri Nets pp. 1-25, Springer Berlin Heidelberg, 2001

[8] A.K. Dey, Understanding and using context. Personal and ubiquitous computing, 5(1), 4-7, 2001

[9] R. Dijkman and M. Dumas and C. Ouyang, Semantics and analysis of business process models in BPMN, Information and Software Technology (2008), pp. 1281-1294

[10] S. Evangelista, High level petri nets analysis with Helena, In Applications and Theory of Petri Nets 2005, pp. 455-464, Springer Berlin Heidelberg, 2005

[11] M. Fouda-Ndjodo, P. Essawe-Ndedi, R. Atsa-Etoundi, An Interperspective-Oriented Business Process Modeling Approach, LNBIP 50, Springer Heidelberg, 2010, pp. 145–156

[12] I. Krschel (2010). On the Notion of Context for Business Process Use. In ISSS/BPSC (pp. 288-297).

[13] L.T. Ly and S. Rinderle and P. Dadam, Semantic correctness in adaptive process management systems, Lecture Notes in Computer Science (2006), pp. 193-208

[14] H. Meyer, Calculating the semantic conformance of processes., In : Business Process Management Workshops. Springer Berlin Heidelberg, 2008. pp. 473-483.

[15] Regev, G.; Soffer, P.; Schmidt, R.: Taxonomy of flexibility in business processes. In: Proc. Seventh Workshop on Business Process Modeling, Development,and Support (BPMDS). Requirements for flexibility and the ways to achieve it. Luxemburg, 2006

[16] J. Schaffner, H. Meyer and M. Weske . A formal model for mixed initiative service composition. In IEEE International Conference on Services Computing, 2007. SCC 2007. (pp. 443-450). IEEE.

[17] Schilit, B., Theimer, M.: Disseminating active map information to mobile hosts. IEEE Network8(5):22-32, 1994

[18] J.J. Shi and D. Lee and E. Kuruku, Task-based modeling method for construction business process modeling and automation, Automation in Construction (2008), pp. 633-640

[19] Soffer, P.:Onthe notion of flexibility in business processes. Proceedings of the CAiSE’05 Workshop, 2005; pp. 35-42

[20] H.M.W. Verbeek and T. Basten and W.M.P. van der Aalst, Diagnosing workflowprocesses using woflan, The Computer Journal (2001), pp. 246-279

[21] F.B. Vernadat, Enterprise Modeling And Integration (EMI): Current Status And Research Perspectives, Annual Reviews in Control (2002), pp. 15-25

[22] M. Vervuurt, Modeling Business Process Variability: A search for innovative solutions to business process variability modeling problems, University of Twente (2007)