A Formal Framework for Business Process Modeling

Roger Atsa Etoundi
Department of Computer Science
Faculty of Science
University of Yaounde I

Marcel Fouda Ndjodo
Department of Computer Science
Faculty of Science
University of Yaounde I

Ghislain Abessolo Alo'o
Department of Computer Science
Faculty of Science
University of Yaounde I

ABSTRACT
In this paper, we define a new business process modeling approach based on the abstraction of the execution environment which is presented as a set of observers. The triggering condition of a task is linked to the state of the environment rather than a predefined order between tasks as suggested by other modeling approaches. Relying on the environment and the tasks, a formal framework of a business process is given. The modeling proposal presented in this paper is used to address three non dominant perspectives: context, semantics and goal; along with the usual controlflow perspective.

The model defined in this paper does not deal with specific business processes and lack some analysis. This work can be extended by defining some case studies, carrying some analysis and developing a supporting tool. It will certainly allow enterprises to efficiently scope with the quality of service delivered to the customers and hence to deal with the competitive pressure of the network economy.

Keywords
Business Process Design, Environment State, Process Semantics, Execution Environment. Quality of service.

1. INTRODUCTION
Many techniques have been used to model business processes and workflows, from the very intuitive graph theory [22], to more sophisticated mathematical models such as Petri Nets [1] and, more recently Pi−Calculus [20].

Regardless of the modeling capacities of the available modeling frameworks, in practice, the control-flow perspective seems to have become the dominant perspective [2]. While we admit that encapsulating all the perspectives in one model is not realistic, it has been proven that it is possible to build a model that natively considers more than one perspective without extra-complexity, as shown in [21] for the data perspective, or in [11] for the quality of service perspective . This paper presents a modeling framework that is used to address three non dominant perspectives: context, semantics and goal; along with the usual control flow perspective. In this model, business process environments, i.e. the execution contexts of business processes - defined here as a set of boolean observers are modified by the tasks which have the capacity of changing the values of observers, thus modifying the states of the environments. The concept of environment used here is similar to the Leslie Lamport's environments [15]. The execution conditions of a task are linked to the environment rather than a predefined order between tasks. Relying on the environment and the tasks, a formal definition of a business process is given.

The core model supporting this approach is described in [5, 6], and it is supported by a number of papers [7, 8, 9] that illustrate its multi-perspective nature, giving room for further investigations.

The rest of the paper is structured as follows: section 2 presents some related works; the model is given in section 3 which is the main part of the work; section 4 illustrates the multiperspective nature of the model; and section 5 concludes the paper.

2. RELATED WORKS
This section roughly summarizes how some main modeling approaches deal with the multiperspective nature of business processes.

Graph theory has been widely used to model business processes. But, as it is expressed in [22] the initial weaknesses of this approach, such as the difficulties of reasoning about workflow properties, and the impossibility to express global dependencies has opened ways to enrich the initial model by adding constraints such as path constraint [22] and temporal constraints [10]. Despite all these improvements, it is hard to model other perspectives than the control-flow. This is probably one of the reasons which explain why this approach has not emerged as one of the most important for business process modeling.

The Workflow-nets approach[1], based on Petri Nets, models tasks by transitions, conditions by places, and cases by tokens. This approach mixes a formal semantics to the graphical and intuitive nature of Petri Nets [4]. Despite a number of extensions available for Petri nets such as coloured Petri nets, timed Petri nets and hierarchical Petri nets that suggest the handling of many perspectives, the full power of Workflow-nets (in terms of the abundance of analysis techniques and software tools made available for it by a very prolific research community) needs the abstraction from perspectives other than control-flow and data to be expressed [14].

However, in [21], workflow-nets are extended with data. Each task is associated with three sets that indicate which data elements the task reads, writes or destroys. This extension allows analysis techniques to check for errors such as deadlocks or livelocks without abstracting from the data perspective, and hence extending the detection power of such tools.

In [5, 6], a modeling framework is presented that natively addresses the context, control-flow, time and resource perspectives of business processes. The multiperspective nature of that approach has been illustrated by works focusing on the human resources [9], mobility [8] and security [7] aspects of business processes. This work falls within this approach.

3. THE CONTEXT-BASED MODELING FRAMEWORK
The modeling framework used in this paper has been described in [6]. The cornerstone of this approach is the concept of environment which formalizes the execution contexts of business processes defined here as a set of boolean observers. It is worthy to note that the notion of context used

here is different from the notion of context used in [18], where a context is an assignment of a status (wait or dead) to arcs linking two nodes of a process in order to manage the arrival of tokens. Here, a context is a"means to focus on aspects that are relevant in a particular situation while ignoring others" [19]. The relevant aspects are captured through the concept of observers. A business process environment is therefore the set of relevant observers required for the execution of that business process.

The mutations of business process environments are enforced through business process tasks which have the capacity of changing the values of observers, thus modifying the states of the environments. The business process tasks are ordered by a follow function in order to achieve an expected goal.

The rest of the section is organized as follows: section 3.1 defines the business process environments, business process tasks are defined in section 3.2, section 3.3 gives a formal model of business processes and, section 3.4 shows that this model captures some common routing constructions. In section 3.5, an example to illustrate the model is given.

3.1 Business Process Environments

The context of execution of a business process is an important modeling input that determines a number of actions. As it is not possible to capture the entire context, we restrict ourselves to the part of the real world that is of interest for a business process. We call it the environment. We define an environment as a set of different metrics whose value may change [6]. Every relevant characteristic of the real world is captured through boolean objects that we call observers.

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 describes the behaviour 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.

Definition 2. Observations

Let $\xi =< \theta,S,val >$ be an environment, an observation on ξ is a couple $<P,M>$ where P and M are disjoint sets of observers of θ.

The set of the observations on the environment ξ is denoted $O\xi$. When the context is clear, we write O for O_{ξ}.

Definition 3. Satisfaction of an observation

Given an environment $\xi =< \theta,S,val >$, the satisfaction of an observation obs $=<P,M> \in O$ in a state $s \in S$ is given by the function

$\Phi : S \times O \rightarrow Bool$ defined by:

$\Phi(s, obs) = (\forall o \in P, s(o)) \wedge (\forall o \in M, \neg s(o))$

We write s(obs)for $\Phi(s, obs)$.

Definition 4. Gap between states

Givenanenvironment $\xi =<\theta,S,val>$,two states $s_1,s_2 \in S$, the gap between s_1 and s_2 (denoted $s_1 \bullet s_2$) is defined by:

$$s_1 \bullet s_2 = \{o \in \theta: s_1(o) = s_2(o)\}.$$

The gap between two states s1 and s2 is the set of the observers that have different values in s_1 and s_2.

3.2 Business Process Tasks

The concept of task is not quite new in the modeling of workflows. All the workflow modeling approaches consider this concept [13]. In [5], a task is defined as a state transition function task: $S \rightarrow S$. This definition includes multipurpose tasks - that can produce different effects according to the initial state, and single-purpose tasks which produce the same effect (post-condition), whatever the initial state may be. Intuitively, any multipurpose task can be viewed as a"combination" of single purpose tasks with appropriate branching. Therefore, single-purpose tasks behave like atomic tasks. In this paper, we restrict ourselves to atomic tasks.

Definition 5. Tasks

Let $\xi =<\theta,S,val>$ be an environment, a task on ξ is a triple <t,ec,action> where *t* is the identifier of the task, *ec* is an observation specifying its precondition, and *action* is an observation specifying its post-condition. In the rest of the 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)).*

Definition 6. Conflicting Tasks

Let t_1, t_2 be two tasks on the environment $\xi =<\theta,S,val>$, we say that t_1 and t_2 are conflicting tasks if:

$(P(t1) \cap M(t2) = \emptyset) \vee (P(t2) \cap M(t1) = \emptyset)$, i.e there exist an observer on which t_1 and t_2 have opposite actions.

This notion can easily be extended to a set of tasks as follows: ts is a non conflicting set of tasks if: $\forall \{t_1,t_2\} \subseteq ts$, t_1 and t_2 are not conflicting tasks.

3.3 Formal Model for Business Processes

A business process is defined as "a set of logically related tasks performed to achieve a defined business outcome" [12] which is the goal of the business process. According to the section 3.2, any business process with multipurpose tasks can be modeled as a business process with single-purpose tasks. In this framework, business processes are therefore formalized as follows:

Definition 7. *Business Processes*

Given an environment ξ, a Business Process is a tuple BP $=<\theta, T, f, g>$ where:

- θ is a set of observers over ξ,

- *T* is a set of tasks on ξ,

- $g \in O$ is a distinguished observation called the goal of BP.

- f: $T \rightarrow 2^T$ is a function which, for every task, gives the names of the tasks that can be executed right after it.

We observe that in this definition, the follow function can be cyclic, allowing this framework to model loops. The execution model, (i.e. the operational semantics) of the business process is defined through the definition of the execution of a non conflicting set of tasks.

Definition 8. *Execution of non conflicting tasks*

Let ξ be an environment, BP =<θ,T,f,g> a business process over ξ, *ts* a non conflicting set of tasks, and *s* a state of ξ.

The execution of the set of tasks *ts* in the state *s* moves the environment into the state *s'*, and activates the set of taks *ts'*such that:

- *s'• s =(∪ t∈tsP(t)) ∪ (∪ t∈tsM(t))* (the gap between *s* and *s'* is the set of observers modified by tasks of *ts*)
- *ts'= {t ∈∪ t∈tsf(t): Φ(s',P(ec(t))) ∧ Φ(s',M(ec(t)))}* (the followers of the tasks of *ts* whose execution conditions are satisfied in s')

We write *exec(ts, s) = (ts', s').*

The intuitive idea is that all the tasks whose execution conditions are satisfied in a state are concurrently executed, unless they are conflicting.

Definition 9. *Implementations of Business Processes*

Let ξ be an environment, BP =< θ,T,f,g > a business process over ξ.An implementation of BP is a list of couples < (ts_0,s_0), ...(ts_n,s_n) > where:

– s_i is a state,

– ts_i is a non conflicting set of tasks,

– $(ts_{k+1}, sk_{+1}) = exec\ (sk,sk)$

– *g* is satisfied in s_n.

The complete investigation of the operational semantics is not the main purpose of this paper. A substantial amount of this investigation can be found in [5].

3.4 Common Routing Constructions Modeling

A model pretending to describe a workflow process shall implement some basic routing constructions that will guide the flow of work. Along with the sequential execution which is part of our business process definition and implemented by the *f* (follow) function, it should be possible to address parallel execution, switching and synchronization.

This section shows that execution conditions and the follow function are very powerful routing tools that will enable one to route work in all cases.

Definition 10. Sequential execution

The routing"t_2 is executed after t_1" is formalized as $f(t_1)= \{t_2\}$.

Definition 11. Parallel execution

The routing"t_1 can be executed in parallel with t_2 after the task *t*" is formalized as

$$\begin{cases} ec\ t_1 = ec(t_2 \\ t_1, t_2 \in f\ t \\ t_1 \text{ and } t_2 \text{ are not conflicting} \end{cases}$$

Definition 12. Switch

The routing"after *t*,either $t_1$or t_2 will execute, not both" is formalized as:

$$\begin{cases} (P\ ec\ t_1 \cap M\ ec\ t_2 = \varnothing) \vee \\ (M\ ec\ t_1 \cap P\ ec\ t_2 = \varnothing) \\ t_1, t_2 \in f\ t \end{cases}$$

This formalization says that at most one of the tasks t_1 and t_2 can be executed after *t* because their execution conditions are conflictual.

Definition 13. Synchronization

The routing"*t* will be executed only when both t_1 and t_2 have been executed" is formalized as:

$$\begin{cases} 1\ (P\ ec\ t \subset (P\ t_1 \cup P\ t_2) \wedge \\ P\ ec\ t\ \ P\ t_1 \wedge P\ ec\ t\ \ P\ t_2 \\ 2\ (M\ ec\ t \subset (M\ t_1 \cup M\ t_2) \wedge \\ M\ ec\ t\ \ M\ t_1 \wedge M\ ec\ t\ \ M\ t_2 \end{cases}$$

Whose meaning is that *t* can only be executed if both *t1*and *t2* have been executed. None of them alone suffices for *t* to be executed.

4. BUSINESS PROCESS PERSPECTIVES

The definition of a business process as a set of logically related tasks directly suggests the control flow perspective of a business process. A coherent theory about business processes cannot ignore this perpective. But this perspective should not over-shadow the others whose importance have been stressed a long time ago [3].
In this section, we show how this model captures three usually uncovered perspectives. The first one is the context perpective which describes the environment of the business process. The second one is the goal perspective which defines the expected outcome. Finally, we show how this model allows the designer to include semantics in his business process definition.

4.1 Context Perspective

In [17], the context is defined as a perspective that"provides an overview perspective of the process and describes major business process characteristics" for"people who do not know or do not need to know the process in detail". In this case, the context is some kind of summary perspective of the overall

complex process. This is not the sense given to the word context in this paper. Rather, we agree with Aalst et al. in [3] where"the context describes the environment for which a process model has been designed".

In this model, the context is captured by the notion of environment (defined as a set of observers) and the set of tasks. So the context is the set of tools available in a given environment.

For example, in a context of disaster such as after an earthquake, a medical doctor will not request an X-ray when he suspects a fracture, because there is probably no electricity, and even no radiologist around. The context determines the set of actions that can be taken. We can compare the process of treating a common tibia fracture in a normal context (BP1) and in an emergency context (BP2).

BP1 =<θ1, T1, f1, g1> and BP2 =<θ2, T2, f2, g2 > where:

θ1= {hasFracture Symptoms,
isFracture Diagnosed,isTibiaBroken,
isXRayDone, isFracture Confirmed,
isCastApplied, isFracture T reated}

T_1= {DiagnoseFracture ,MakeXRay,

ConfirmFracture, ApplyCast}

θ_2= {hasFracture Symptoms,

isFracture Diagnosed,

isT ibiaBroken, isT ibiaImmobilized,

isFracture T reated}

T_2= {DiagnoseFracture, ImmobilizeT ibia}

In a normal context (BP_1), when a fracture is diagnosed, an X-Ray will be done to confirm the fracture before a cast is applied to treat the fracture. In an emergency context (BP_2), the task *ApplyCast* is replaced by *ImmobilizeT ibia* which can be done with a cast (if available) or any other means (like a piece of wood and a string to tie the leg). Also, the tasks *MakeXRay* and *ConfirmFracture* do not exist in the emergency context.

4.2 Semantics Perspective

When designing a business process, it is crucial to be able to also integrate semantic knowledge within the process [16]. The goal of integrating application knowledge is to enable the system to perform process checks at the semantic level.

This central preoccupation is tackled in this model by the notion of observation that enables one to define conditions. Each observer of a business process environment is a semantic unit. Having it true or false in a state yields a non ambiguous understanding that is captured by the notion of observation that we used to define the pre and post-condition of any business process task. Using our task model, two semantic problems expressed in [16] are easily solved: mutual exclusion and dependency constraints. Mutual exclusion constraints express that two activities are not compatible and should not be executed together. Mutual exclusion constraints are symmetric. For instance administering two incompatible drugs (Marcumar and Aspirineas as in [16]). We do not want a patient to take Aspirin after Marcumar.

Let us define a business process BP =<θ,T,f,g>, where :

θ = {tookAspirin, tookMarcumar, o, ...}

T = {AdministerMarcumar,AdministerAspirin, t1,t2,t3,t4}

AdministerMarcumar :

Pre-condition:

P(ec(AdministerMarcumar)) = {o};
M(ec(AdministerMarcumar)) = {tookAspirin};(we want to make sure the patient did not take Aspirin before)

Post-condition:

P(AdministerMarcumar)= {tookMarcumar};
M(AdministerMarcumar)= {o};

AdministerAspirin :

Pre-condition:

P(ec(AdministerAspirin)) = Ø;
M(ec(AdministerAspirin)) = {tookMarcumar, o};
(we want to make sure the patient did not take Marcumar before)

Post-condition:

P(AdministerAspirin)= {tookAspirin};
M(AdministerAspirin)= Ø;

Tasks t1, t2, t3, t4 :

Pre-condition:

$P(ec(t_1)) = Ø; P(ec(t_2)) = Ø; P(ec(t_3)) = Ø; P(ec(t_4)) = Ø;$
$M(ec(t_1)) = Ø; M(ec(t_2)) = Ø; M(ec(t_3)) = Ø; M(ec(t_4)) = Ø;$

Post-condition:

$P(t_1)= \{o_1\}; P(t_2)= \{o_2\}; P(t_3)= \{o_3\}; P(t_4)= \{o_4\};$
$M(t_1)= Ø; M(t_2)= Ø; M(t_3)= Ø; M(t_4)= Ø;$

Follow function

$f(t_1)$= {AdministerAspirin,AdministerMarcumar}

With such a business process definition, we are assured that the patient will get either Marcumar or Aspirin, according to the value of the observer o that belongs to the pre-conditions of both tasks, but with opposite values. He will never get the two (fig. 1). This solution is the one found in [16].

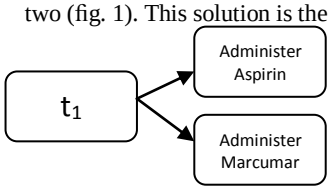

Fig. 1. Switch

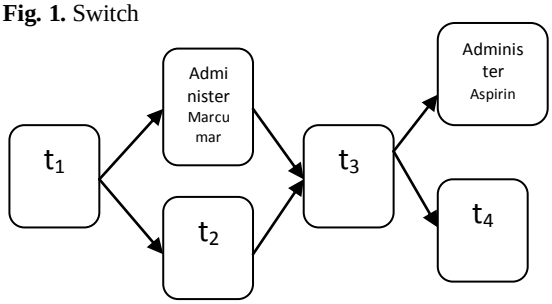

Fig. 2. Global dependency

Even with another follow function defined as:

f(t1)={t2,AdministerMarcumar}; f(AdministerMarcumar)= {t3};

f(t2} = {t3}; f(t3)= {AdministerAspirin, t4}

If *AdministerMarcumar* is the only task of the environment that modifies the observer *tookMarcumar*, we are assured that

if *AdministerMarcumar* has been executed, *AdministerAspirin* will not be executed. This is because *AdministerMarcumar* sets the observer *tookMarcumar* to true, and no other task will change its value. It is then guaranteed that *AdministerAspirin* cannot be executed as it needs the observer *tookMarcumar* to be false as its precondition. This illustrates the global dependency property of the model. Two tasks do not need to be adjacent to enforce a dependency constraint.

This second solution (illustrated in fig. 2) were labeled conflictual in [16] since the conflict there was dependent of process structure, not on the semantics of tasks as it is the case here.

4.3 Goal Perspective

In this model, the notion of goal is explicit. In a business process BP = <θ,T,f,g >, the goal g is an observation that has to be satisfied at the end of the process. Beyond the disambiguating role of the goal in the business process definition, it increases the reliability potential of the model. Each task can be evaluated according to its input in the satisfaction of the goal (like counting the number of observers that the task sets for the goal). We then have two categories of useful tasks: those that directly impact the goal, and those that contribute to the satisfaction of the execution conditions of another task. For example, with the goal definition

g: P(g)= {isFractureTreated};M(g)= ∅,

The execution of the task ApplyCast is enough to satisfy the goal; but to execute the task; its pre-condition requires that the observers *isFractureDiagnosed* and *isFractureConfirmed* be set to the value true. This is only possible if the tasks DiagnoseFracture (to set the observer *isFractureDiagnosed*) and *ConfirmFracture* (to set the observer *isFractureConfirmed*) have been executed. The task ConfirmFracture itself needs the task *MakeXRay* to be executed to set the observer isXRayDone to true.

The notion of goal also enables the setting of a kind of process quality of service. The quality of service here does not refer to performance. It refers to the nature of the process. Using the previous section example, we can define goals with different quality of service.

g: P(g)= {isFracture T reated};M(g)= ∅

The execution of the task *ApplyCast* is enough to satisfy the goal; but to execute the task; its pre-condition requires that the observers *isFractureDiagnosed* and *isFractureConfirmed* be set to the value true. This is only possible if the tasks *DiagnoseFracture* (to set the observer *isFractureDiagnosed*) and *ConfirmFracture* (to set the observer *isFractureConfirmed*) have been executed. The task *ConfirmFracture* itself needs the task *MakeXRay* to be executed to set the observer *isXRayDone* to true.

The notion of goal also enables the setting of a kind of process quality of service. The quality of service here does not refer to performance. It refers to the nature of the process. Using the previous section example, we can define goals with different quality of service.

g: P(g)= {isFracture T reated};M(g)= ∅,

g': P(g')= {isFracture T reated, isCastApplied}:

M (g') = ∅,

It appears that g' suggest a better quality of service than g, because unlike g, g' requires that the process designed respects particular care (applying a cast) that is considered essential for the quality of the final result.

5. CONCLUSION

In this paper, a business process modeling approach based on the concept of environment is presented, that allows one to precisely describe the execution context of business processes by using a set of observers. The model presented in this paper is used to address three perspectives: the context perspective, the semantic perspective and the goal perspective.

A task concept has been defined, where the action of every task is deterministic and the execution condition of a task is linked to the environment rather than a predefined and rigid order between tasks. Finally, given the environment and the tasks, a formal definition of a business process is given, that associates the environment and the tasks to a follow function in order to achieve a given goal. By expressing constraints over tasks using conditions over the environment rather than a strong task ordering scheme, tasks behave like independent components.

In ongoing researches, we are looking forward to enrich this model with business process generation and verification capacity given that semantics is easily expressed.

6. REFERENCES

[1] van der Aalst, W.M.P.: The application of Petri nets to workflow management. The Journal of Circuits, Systems and Computers 8(1), 21–66 (1998).

[2] van der Aalst, W.M.P., van Hee, K.M.: Workflow management: models, methods, and systems, p. 267. MIT Press, Cambridge (2004).

[3] van der Aalst, W.M.P., Rosemann, M., Dumas, M.: Deadline-based Escalation in Process-Aware Information Systems. Decision Support Systems 43(2), 492–511 (2007)156.

[4] van der Aalst, W.M.P.: Three Good Reasons for Using a Petri-net-based Workflow Management System. In: Navathe, S., Wakayama, T. (eds.) IPIC 1996, Cambridge, Massachusetts, pp. 179–201 (1996).

[5] Atsa Etoundi, R.: A Domain Engineering approach for multiperspectives Workflow modelling, Ph.D Thesis, University of Yaounde I - Cameroon (2004).

[6] Atsa, R., Fouda, M.: An Abstract Model for Workflows and Business Processes. In: CARI 2002, pp. 239–247 (2002).

[7] Atsa, R., Fouda, M.: Security Based Approach to Data Consistency in a Virtual Enterprise. In: ACS-IEEE International Conference on Computer Systems and Applications (2003), IEEE Catalog Number: 03EX722, ISBN: 0-7803-7983-7.

[8] Atsa, R., Fouda, M.: Mobile-Based support for Business Processes: Feasibility and Correctness. In: ACS-IEEE International Conference on Computer Systems and Applications (2003), IEEE Catalog Number: 03EX722, ISBN: 0-7803-7983-7.

[9] Atsa, R., Fouda, M.: Human Resource Constraints driven Virtual Workflow Specification. In: Proceeding of the

International Conference on Signal-Image technology & Internet-based Systems, pp. 176–182 (2005), 2-9525435-0 c IEEE SITIS 2005.

[10] Attie, P., Singh, M., Sheth, A., Rusinkiewicz, M.: Specifying and enforcing intertask dependencies. In: Proceedings of the 19th VLDB Conference (1993).

[11] Cardoso, J., Sheth, A.,Miller, J., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. Journal of Web Semantics 1(3), 281–308 (2004).

[12] Davenport,T., Short, J.E.:TheNewIndustrialEngineering: InformationTechnology and Business Process Redesign. Sloan Management Review, 11–27 (Summer 1990).

[13] Dehnert, J., Freiheit, J., Zimmermann, A.: Modelling and evaluation of time aspects in business processes. Journal of the Operational Research Society 53, 1038–1047 (2002).

[14] Kiepuszewski, P., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Fundamentals of Control Flow in Workflows. Acta Informatica 39(3), 143–209 (2003).

[15] Lamport, L.: Specifying Concurrent Program Modules. ACM Transactions on Programming Languages and Systems 5(2), 190–222 (1983).

[16] Ly, L.T., Rinderle, S., Dadam, P.: Semantic correctness in adaptive process management systems. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 193–208. Springer, Heidelberg (2006).

[17] List, B., Korherr, B.: An Evaluation of Conceptual Business Process Modelling Languages. In: Proceedings of the 21st ACM Symposium on Applied Computing (SAC 2006). ACM Press, New York (2006).

[18] Mendling, J., van der Aalst, W.M.P.: Towards EPC Semantics based on State and Context. In: Proceedings of the 5th EPC Workshop EPK 2006, CEUR 2006 Workshop Proceedings, pp. 25–48 (2006).

[19] Motschnig-Pitrik, M.: Contexts as means to decompose Information Bases and represent relativized Information. In: Proc. CHI Workshop #11: The Who, What, Where, When, Why and How of Context-Awareness. Hague, Netherlands (2000)

[20] Puhlmann, F.: Why do we actually need the Pi-Calculus for Business Process Management? In: Abramowicz, W., Mayr, H. (eds.) 9th International Conference on Business Information Systems (BIS 2006). Bonn, Gesellschaft fur Informatik. LNI, vol. P-85, pp. 77–89 (2006).

[21] Trcka, N., van der Aalst, W.M.P., Sidorova, N.: Data-Flow anti-patterns: Discovering data-flow errors in Workflows. In: van Eck, P., Gordijn, J., Wieringa, R. (eds.) CAiSE 2009. LNCS, vol. 5565, pp. 425–439. Springer, Heidelberg (2009).

[22] Wenfei, F., Weinstein, S, (1999).: Specifying and Reasoning About Workflows with Path Constraint. In: Hui, L.C.K., Lee, D.-L. (eds.) ICSC 1999. LNCS, vol. 1749, pp. 226–235. Springer, Heidelberg.