MODELING TIME CONSTRAINTS IN INTER-ORGANIZATIONAL WORKFLOWS

Mouna Makni[1,3], Nejib Ben Hadj-Alouane[1], Moez Yeddes[2] and Samir Tata[3]
[1]*National Engineering School of Tunis, OASIS Laboratory, Tunis, Tunisia*
[2]*National School of Computer Sciences, CRISTAL Laboratory, Tunis, Tunisia*
[3]*Institut TELECOM, TELECOM SudParis, UMR CNRS Samovar, Évry, France*

Keywords: Inter-organizational, Workflows, Time constraints, Time Petri nets, Models.

Abstract: This paper deals with the integration of temporal constraints within the context of Inter-Organizational Workflows (IOWs). Obviously, expressing and satisfying time deadlines is important for modern business processes, and need to be optimized for efficiency and extreme competitiveness. In this paper, we propose a temporal extension to CoopFlow (Tata et al., 2008), an existing approach for designing and modeling IOWs, based on Time Petri Net models and tools. Methods are given, based on reachability analysis and model checking techniques, for verifying whether or not the added temporal requirements are satisfied, while maintaining the core advantage of CoopFlow; i.e. that each partner can keep the critical parts of its business process private.

1 INTRODUCTION

In recent years, companies have proven that collaboration plays a very significant role in facing industrial competition and pressures. In this new modern environment, it is important for business partners to rapidly join forces in order to create new and valuable expertise, with cheaper costs and within restrictive deadlines. For this purpose, companies have largely adopted workflow concepts and techniques. In order to promote a virtual environment within which companies can effectively engage each other, two important processes need to be properly designed: (1) workflow abstraction, and (2) workflow matching (Tata et al., 2008).

In fact, the basic supposition of a proper virtual environment is that the description of the business processes of the partners are well advertised and readily available in a common registry. A challenging issue is industrial secret preservation (Amirreza and Eder, 2008; Chebbi et al., 2006), especially in occasional collaboration where there are serious consequences for companies with regards to fully exposing their business knowledge. Thus, abstraction is a first step used prior to advertising, and consists of eliminating details that are not necessary for cooperation needs. The next step is matching, which consists of identifying and selecting the appropriate partner able to provide the required service. Within the above context, the CoopFlow approach is designed to support short-term ascending workflow cooperation within virtual enterprises, using the publish/subscribe paradigm. We have already presented the basic ideas of the CoopFlow approach and compared it with the existing approaches for workflows cooperation (Tata et al., 2008).

So far, in CoopFlow, the focus has been on the structural conformance of the processes of the partners; i.e., the partners which execute complementary tasks can interconnect within an Inter-Organizational Workflow (IOW). In this paper, we undertake the new concern of ensuring that the timing constraints and/or expectations of the involved partners are also compatible. With regards to functional requirements, workflow behaviors are typically closely tied to the timing requirements. Therefore, enterprises, which are generally looking for cost reduction, should strive to include timing requirements as part of their workflow matching process.

Our main purpose in this paper is to provide a time-oriented inter-organizational workflow modeling and management framework, within the context of CoopFlow. Such a framework will enable enterprises to specify temporal constraints and detect, early on, temporal contradictions that may constitute obstacles for their cooperation. For instance, deadlines constraints can be violated if all participating enterprises do not respect their allowed temporal bounds for delivering the invoked services. Given an abstracted version of its business process, a company should add temporal expectations for actions that will be performed by external parties i.e. time duration after which the supplied services cannot be performed anymore. On the other side, eventual partners will be able to capture temporal requirements as well as business behavior from the common registry, in order to execute temporal reasoning and verification. Thus, they can detect whether deadlines constraints will be satisfied or violated in the resulting IOW.

This paper is organized as follows. Section 2 presents the necessary background from Time Petri Nets, needed to understand our work. Section 3 presents an overview of the existing CoopFlow approach to illustrate our needs. Section 4 exposes our method, based on Time Petri Nets, for incorporating timed constraints. Section 5 presents a survey of the important articles in the literature related to this paper's topic, and dealing essentially with timed models for workflow.

2 BACKGROUND

In this section we present some preliminaries and definitions that are useful for our work. CoopFlow uses the Petri Net (PN) formalism to have powerful workflow analysis and validation techniques. In order to introduce time constraints, we adopt Time Petri Nets (TPN), which is one of Petri Nets time extensions proposed by Merlin (Merlin, 1974). As workfow systems are generally finite, they can be modeled by bounded Time Petri Nets, which offers a useful tool for workflow temporal behavior analysis.

Definition 1: Time Petri Nets. A TPN (Berthomieu and Diaz, 1991) is a tuple (P,T,B,F,M_0,S) where:

- P is a finite set of places;
- T is a finite set of transitions;
- $B : T \times P \to N$ is the backward incidence function;
- $F : T \times P \to \mathbb{N}$ is the forward incidence function;
- $M_0 : P \to \mathbb{N}$ is the initial marking function, and,
- $S : T \to \mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \infty)$ is a mapping called static interval.

Assuming that θ is the instant a transition t becomes enabled for the last time and $S(t) = [a,b]$ is the interval associated, then $[\theta+a, \theta+b]$ is the time range to fire the transition, except if t is disabled by firing another conflicting transition. A clock is associated to each validated transition to count elapsed time until its firing date. a and b are called the earlier (EFT) and last (LFT) firing times. Firing a transition is instantaneous and modifies the marking as classical Petri Nets. Moreover, some variations can be found in the literature concerning transition firing (Berthomieu and Diaz, 1991; Pezze and Young, 1999). We consider the Strong Time Semantics (STS) semantic, where an enabled transition must be fired at least when time reaches the LFT limit, unless it is disabled by another transition firing. In this case, temporal constraints are sufficient to resolve conflicts between transitions.

Definition 2: State Classes Graph. A state of the TPN can be defined by a couple (M,I) where:

- M is a net marking, and,
- I is a vector of intervals corresponding to firing intervals for each transition enabled by M.

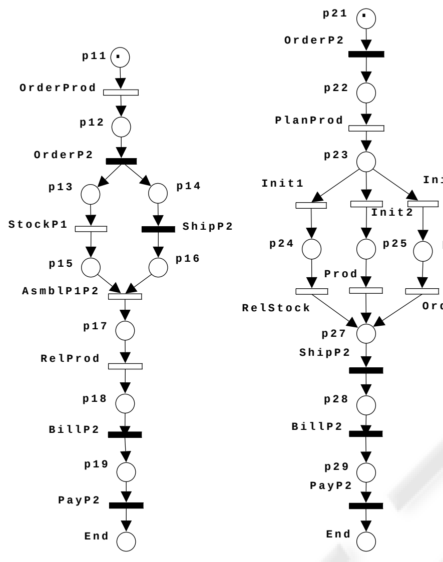
Obviously, we can see that reachable states from the initial marking is infinite because, in part, possible firing times for each transition are infinite. To analyze TPN behavior and state reachability, an enumerative technique is proposed in (Berthomieu and Menasche, 1983) based on state classes. This technique leads to a state class graph (SCG) generation which is an oriented graph that represents all possible sequences of firing values during each transition firings times. SCG generation is automated using the tool TINA (Berthomieu and Vernadat, 2006). Then, some properties such as accessibility, marking and temporal bounds can be checked on the graph.

3 THE CoopFlow APPROACH

In the context of short-term cooperation, enterprises with complementary skills are dynamically interconnected according to their needs. Our work consists of an extension of CoopFlow (Tata et al., 2008), a bottom-up approach that allows dynamic interconnection of a set of partners by providing a useful artifact for their privacy preservation. CoopFlow consists of three steps: (1) workflow abstraction and advertisement, (2) workflow matching and interconnection, and (3) workflow cooperation.

The goal of the first step is to allow partners to publish their business processes in a common registry without revealing their workflow's internal structures. This is done by the abstraction process and the resulting public workflow exposes only cooperative activities required for cooperation needs. The matching process in the second step consists of comparing the business behavior of the candidates i.e. of public workflows published in the registry, according to a given criteria. If the result is positive, workflows are then interconnected. We have used Petri Nets and symbolic observation graphs as theoretical foundations for workflow abstraction and matching (Klai et al., 2009; Chebbi and Tata, 2007). The third and last step consists of the inter-enterprises workflow cooperation platform (deployment, execution, management, etc...) (Chebbi and Tata, 2005).

We present a running example which will detail the abstraction and the matching processes.



(a) (Partner1) (b) (Partner2)

Figure 1: The client/provider example.

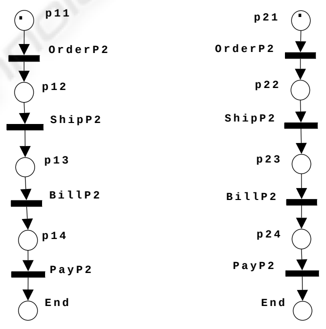
3.1 Running Example

Lets us consider a client/provider example in an industrial context, involving two candidates for a workflow cooperation: *Partner*1 and *Partner*2. *Partner*1 business process, shown in figure 1(a), supplies an industrial product, made up of two components *P*1 and *P*2. The product processing is launched when the corresponding order is received (*OrderProd*). While *P*1 is available in the stock, *P*2 should be requested to an external supplier (*OrderP*2). When the two components are available (*StockP*1 and *ShipP*2), they are assembled (*AsmblP*1*P*2) and the product is delivered to the client (*RelProd*). Finally, *Partner*1 receives the bill (*BillP*2) and makes the payment (*PayP*2).

On the other hand, the *Partner*2's workflow is illustrated in figure 1(b). After receiving an order to deliver component *P*2 (*OrderP*2), and as a result of customer specifications analysis (*PlanProd*), three cases are possible. *Partner*2 can either recover the order in the stock (*RelStock*), produce *P*2 locally (*Prod*), or supply it to a third participant (*Order*). When component *P*2 is built, it is shipped to the customer (*ShipP*2). Then, the bill is sent to *Partner*1 (*BillP*2), for the payment reception (*PayP*2). Filled transitions represent the cooperative activities, while others are internal activities and are not supposed to be visible to other partners.

3.2 The Abstraction Process

To support cooperation, one has to deal with the partners' privacy respect. In (Chebbi and Tata, 2007), we proposed a set of reduction rules as well as an abstraction algorithm. Given an initial workflow, the principle of the algorithm consists in removing all internal activities whose elimination doesn't affect the visible behavior of the initial workflows. Thus, we start by identifying the various reduction patterns and then applying them the suitable reduction rules. This procedure is repeated until the removing of all internal activities that don't play any direct role in the cooperation. The abstraction process applied to the private workflows of *Partner*1 and *Partner*2, generates the public workflows illustrated in Figure 2.



(a) (Partner1) (b) (Partner2)

Figure 2: The public workflow of the partners.

3.3 The Matching Process

Given public workflows advertised in a registry, the selection criteria making a choice of an effective partner is based on the observable behavior, i.e. the behavior on the cooperative transitions, which must match with the observable behavior of the eventual partner. Two cooperative transitions are considered equivalent according to their business semantics if they are described using references to equivalent concepts in a semantic model. If the matching result is positive, the workflows are then interconnected. In the example above, we notice the business behavior complementary of *Partner*1 and *Partner*2 workflows. As shown in Figure 3, interconnection can be held between the corresponding transitions *OrderP*2, *ShipP*2, *BillP*2 and *PayP*2 of *Partner*1 and *Partner*2.

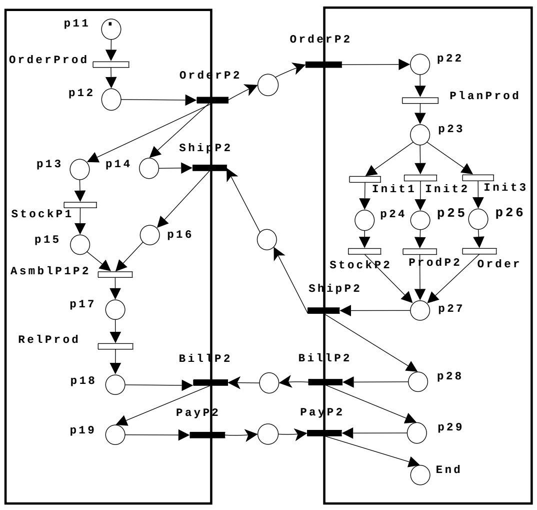


Figure 3: The inter-organizational workflow.

3.4 Dealing with Time Constraints

In the context of cooperation, partners generally want to run a workflow within a specific period of time. In the previous example, it is obvious that *Partner*1 needs information concerning *P*2 delivery due date, because the cooperation can be achieved only if *P*2 delivery is properly attended on time. We propose to use the TPN formalism for modeling temporal behavior of the given workflows. Figure 4 illustrates the temporal version of the internal workflow of *Partner*2. We can see that the necessary duration for *P*2 delivery depends on the case executed:

- if *P*2 is available on the stock, the processing takes at least 2 time units, at most 4 times units;
- if *P*2 is produced by *Partner*2, the processing takes at least 4 time units, at most 6 times units, and,
- if *P*2 is supplied to an external provider, the processing takes at least 5 time units, at most 7 times units.

Let us consider a situation in which *Partner*1 specifies that *P*2 delivery should not exceed 3 time units since the order is sent. *Partner*2 is able to satisfy *Partner*1's constraint only if *P*2 is available on the stock. Thus, even if the structural conformance of the candidates is validated by the matching process, their corresponding public workflows do not contain enough information to ensure that timing constraints of the involved partners are also compatible. Specifying deadlines becomes a base requirement within

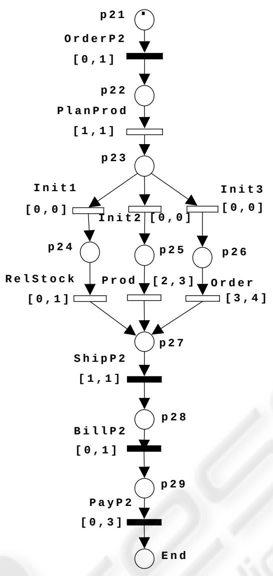


Figure 4: The temporal private workflow of *Partner*2.

IOWs, especially in a critical business context (financial, medical, etc.), where some parties want to pay more but also imposes stricter services delivery times. If the deadline is violated, they can ask for compensation. Thus, some temporal information have to be added as part of to the advertised public workflows in order to allow eventual partners to verify that deadlines are respected, which will lead to a successful execution of the resulting IOW.

4 TEMPORAL CONSTRAINTS IN COOPFLOW

4.1 Our Approach

As explained above, according to CoopFlow approach, each partner holds locally a Petri Net modeling its business process. The abstraction process generates the corresponding public process which will be published in the common registry. In order to provide a CoopFlow temporal extension, we assume that:

- each partner, given its business process modeled by a PN, also holds a timed version modeled by a TPN, called ”Private temporal workflow”, and,
- given time intervals associated with activities, a partner is able to specify the estimated execution times of its critical supplied tasks. These deadline constraints are called ”Timing constraints”.

We propose in this section a method for modeling deadlines between critical cooperative activities.

Our approach, illustrated in Figure 5, is based on the existence of a public workflow generated following the abstraction process. It consists of adding to this abstract representation deadline constraints between two cooperative activities, interacting with the same company, and referring to a duration limit between an input and an output messages. Several deadlines can be specified by a partner. Attached constraints will be advertised in the common registry and propagated to external parties as part of the resulting ”Public temporal workflow”.

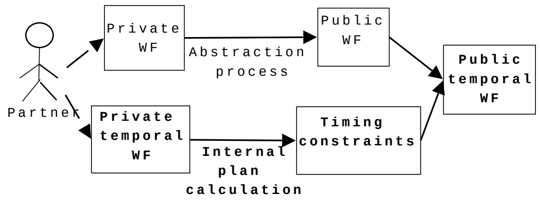


Figure 5: A CoopFlow timed extension.

Even if the internal business process is not changing, the deadlines specified by a partner may vary and can be dynamically managed while computing the estimated execution times. Thus, depending on its internal plan execution and its available resources, a partner is able to advertise several ”Public temporal workflow” versions, in order achieve different cooperation needs, depending on clients requirements.

4.2 Modeling Deadline Constraints

While TPN absolute time information is not preserved through local clocks in TPN, a simple alternative, introduced by (Toussaint et al., 1997), consists of modeling a parameter in the corresponding TPN which describes the allowed execution time between two events. The parameter includes a set of places and transitions representing a non-intrusive observer, and transforms the problem into a reachability analysis of a specific state. This approach, detailed in (Godary, 2008), is illustrated in figure 6, in which an observer is introduced to supervise the maximum duration between transitions *Begin* and *End*.

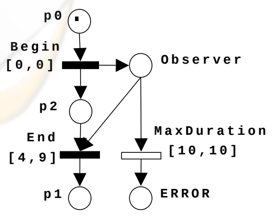


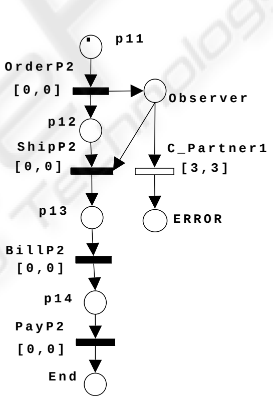
Figure 6: A deadline constraint modeled by an observer.

When the transition *MaxDuration* is validated (by the existence of a token in the place *Observer*), the associated clock is enabled. The token will be used either by firing transition *End* if the constraint is satisfied or transition *MaxDuration* if the given time limit is reached. Checking the maximum duration becomes an analysis of the reachability of place *ERROR*, which is marked if the execution time is longer than the specified duration.

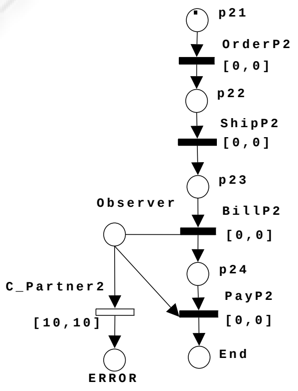
4.3 Incorporating Timing Constraints and Advertisement

Let us consider the following constraints for the previous example:

- between the instant *Partner*1 orders for component *P*2 (*OrderP*2) and the time a reception notification is received (*ShipP*2), he waits at most 3 time units;
- between the instant *Partner*2 sends the bill for *P*2 delivery (*BillP*2) and the instant the payment is received (*PayP*2), he waits at most 10 time units.



(a) (Partner1)



(b) (Partner2)

Figure 7: The public temporal workflow of the partners.

To advertise these properties, partners would have to translate these informal requirements into the temporal constraint model. Figure 7 illustrates such an advertisement for the two previous constraints: an observer is added in the corresponding public workflow between the critical cooperative transitions, showing the constraint as the associated temporal interval to *C_Partner*1 and *C_Partner*2 transitions.

4.4 Workflow Matching and Interconnection

Public temporal workflows contain enough information to ensure the structural and temporal suitability of the eventual partners. First, the business complementarity behavior is validated by the algorithms defined in CoopFlow. If this first step is successful, an analysis is done to validate the temporal behavior conformance of the eventual partners as follows.

4.4.1 Reproducing Temporal Constraints into Private Temporal Workflows

When a partner receives a time constraints advertisement, he should execute locally a temporal reasoning and verification to validate that the deadline will be satisfied in the IOW. This means that a partial IOW is generated using its private temporal workflow and the received public temporal workflow which includes the observer. The deadline respect is equivalent to the analysis of the reachability of place *ERROR*. Figure 8 and 9 illustrate private temporal workflows of *Partner*1 and *Partner*2, combined with the corresponding public temporal workflows advertised.

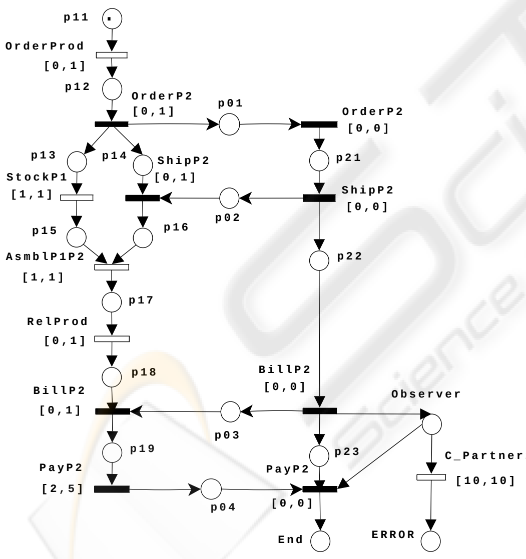


Figure 8: The timed workflow verification for *Partner*1.

4.4.2 Timed Verification

Introducing an observer transforms the problem into a reachability analysis of place *ERROR*. This can be done using the tool TINA (Berthomieu and Vernadat, 2006), which automatically generates the state classes graph. Three cases can be found:

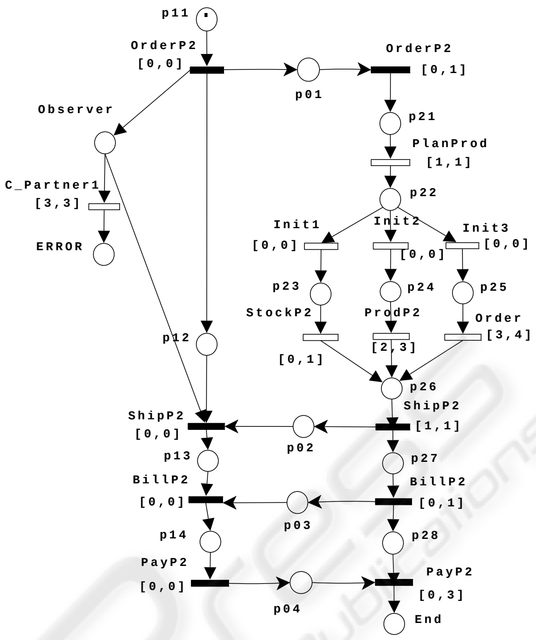


Figure 9: The timed workflow verification for *Partner*2.

- a state including the place *ERROR* in its marking is never reachable, while the place *End* is accessible; this means that all the firing sequences and times verify the required deadline;
- the place *ERROR* is reachable but the place *End* is never accessible; this means that the desired property is never satisfied;
- the two specific places *ERROR* and *End* appear in the corresponding state classes graph; this means that the deadline constraint is not always satisfied. We need a further investigation to detect restrictions to apply allowing the partner to satisfy deadline constraint.

A preliminary verification leads to the identification of the two cases in which the cooperation can always or can not be positive. Following this result, and if the matching is positif, cooperation move to the next step which is workflow interconnection.

- **Timed Verification for** *Partner*1**.** Let us consider reachable state classes, generated by TINA, for the partial Inter-OW illustrated in figure 8. The interval associated to the transition *C_Partner*2 is 10 time units. Table 1 shows that the place *ERROR* is never reachable (i.e. *C_Partner*2 is a dead transition). This means that, while the reachable states are infinite because of the temporal interval associated to each transition, the deadline constraint is always respected.
- **Timed Verification for** *Partner*2**.** *Partner*2 should also verify the temporal constraint adver-

Table 1: The *Partner*1 state classes for the interval associated to $C\_Partner2[10,10]$.

C_Partner2 [10,10]
state 0 = p11
state 1 = p12
state 2 = p01 p13 p14
state 3 = p13 p14 p21
state 4 = p02 p13 p14 p22
state 5 = Observer p02 p03 p13 p14 p23
state 6 = Observer p03 p13 p16 p23
state7 = Observer p03 p15 p16 p23
state 8 = Observer p03 p17 p23
state 9 = Observer p03 p18 p23
state 10 = Observer p19 p23
state 11 = Observer p04 p23
state 12 = **End**
state 13 = Observer p02 p03 p14 p15 p23
state 14 = p13 p16 p22

tised by *Partner*1, which is a maximum of 3 time units between transitions *OrderP*2 and *ShipP*2. The state classes generated is illustrated in Table 2 below. We can see that the cooperation can not always meet the temporal constraint, because the two places *ERROR* and *End* are reachable. This means that the temporal conformance can be achieved under some additional restrictions.

Table 2: The *Partner*2 state classes for the interval associated to $C\_Partner1[3,3]$.

C_Partner1 [3,3]
state 0 = p11
state 1 = Observer p01 p12
state 2 = Observer p12 p21
state 3 = Observer p12 p22
state 4 = Observer p12 p23
state 5 = **ERROR** p12 p23
state 6 = **ERROR** p12 p26
state 7 = **ERROR** p02 p12 p27
state 8 = **ERROR** p02 p03 p12 p28
state 9 = Observer p12 p26
state 10 = Observer p02 p12 p27
state 11 = Observer p02 p03 p12 p28
state 12 = p03 p13 p28
state 13 = p14 p28
state 14 = p04 p28
state 15 = **End**
state 16 = p13 p27
state 17 = Observer p12 p24
state 18 = **ERotROR** p12 p24
state 19 = Observer p12 p25
state 20 = **ERROR** p12 p25

4.4.3 Timed Verification for Workflow Negotiation

Besides analyzing whether a deadline requirement is respected or n, it is also important to determine maximal bound under which the constraint is guaranteed to be satisfied. In order to obtain the maximum limit allowed, the tool LPT (Little Parametric Tool) (Godary, 2008) proposes an automatic analysis of the TPN with added observer, while parameterizing the interval associated to the transition modeling the deadline constraint. This reachability analysis uses the state classes graph generated automatically using the tool TINA (Berthomieu and Vernadat, 2006).

In the previous example, *Partner*2 can not meet *Partner*1's temporal requirement. On our future work, we will focus on model checking techniques to determine necessary constraints to apply on the TPN in order to meet the deadline. A communication protocol will be defined between the two eventual partners for exchanging necessary information. Even if *Partner*2 is unable to modify its internal process and add other timing constraints, he can send to *Partner*1 the necessary duration to deliver component *P*2. The code below illustrates a part of LPT result for searching the corresponding temporal constraint. It is shown that cooperation can always be held between *Partner*1 and *Partner*2 if the specified deadline between the two activities *OrderP*2 and *ShipP*2 is greater than 7 time units.

```
--------------------------------------------
 INITIAL PETRI NET : Partner2.ndr
 ANALYSIS OF THE WORST DURATION BETWEEN
 TRANSITIONS OrderP2 AND ShipP2
--------------------------------------------
 THE BOUND VALUE IS : 7
 Remark : this value is the upper bound,
 it is never reached
--------------------------------------------
 GRAPH SIZE : 17 classe(s), 19 transition(s)
--------------------------------------------
 CALCULATION TIME : 1 seconds
--------------------------------------------
```

5 RELATED WORK

Time management in workflow-based processes has been studied among several aspects: activities duration control, scheduling and prioritization, resources management, etc... The first important issue to deal with is time modeling. In (Eder et al., 2000) and (Eder and Panagos, 2000), time constraints are expressed using a timed workflow graph, which extends worklow graphs by adding some temporal constraints to each activity. In (Amirreza and Eder, 2008), the authors adapted their approach for checking temporal consistency of inter-organizational workflows using workflow views. They present a timed workflow graph approach in order to express the upper and lower bound constraints of task execution. The technique enable partners to execute the IOW without violating temporal constraints such as explicitly assigned deadlines.

Temporal constraints modeling have also been studied in web service compositions research field. In (Diaz et al., 2007), temporal expectations are expressed using goal-oriented engineering. An extension of KAOS, an approach for goal oriented formulation, is proposed and allow formal specification of timing requirements and automatic generation of counterexamples using model checking techniques. In (Kazhamiakin et al., 2006), authors propose an extension of timed automata formalism to specify global timing aspects of web service compositions, called Web Service Timed State Transition Systems (WSTTS). Complex timed requirements can be specified for modeling time intervals between events, bounds or combinations of them. An algorithm is proposed to compute the interval limits allowing to meet the timing constraint. In (Benatallah et al., 2005), business protocols are modeled as deterministic finite state machines, and temporal abstractions of business protocols are specified using timed transitions.

Our work is based on the CoopFlow approach and guarantee that each partner can keep the critical part of its business process private. Activities duration are not fixed. We plan to specify a communication protocol allowing partners to negotiate the specified deadlines according to their needs and their available resources.

6 CONCLUSIONS

In this paper, we presented a time-oriented framework for incorporating and verification deadlines constraints in the context of Inter-Organizational Workflows (IOWs). Even if the business behavior complementarity of the involved parties is validated, missing deadlines while delivering required services may lead to a global failed execution. Based on the existing CoopFlow approach and using the Time Petri Net theory, we proposed a method for expressing and publishing sensible time deadlines, by each partner. State reachability analysis is used for checking temporal correctness of the resulting IOW. A further work will concentrate on formulating a systematic method for assuring the satisfaction and consistency of all the published time constraints, within the context of the global business process, while maintaining the core advantage of CoopFlow; i.e. that each partner can keep the critical parts of its business process private. We have to prove that deadline local verification processes executed by partners can lead to a deadline conformance in the resulting interconnected workflow. Furthermore, we advocate that deadlines should be negotiated. We will concentrate on specifying a communication protocol between eventual candidates, which can lead to constraint negotiation in order to achieve the cooperation.

REFERENCES

Amirreza, T. N. and Eder, J. (2008). Temporal consistency of view based interorganizational workflows. In *2nd International United Information Systems Conference*, pages 96–107, Klagenfurt, Austria.

Benatallah, B., Ponge, J., and Toumani, F. (2005). On temporal abstractions of web services protocols. In *CAiSE Short Paper Proceedings*, Porto, Portugal.

Berthomieu, B. and Diaz, M. (1991). Modeling and verification of time dependent systems using time petri nets. *IEEE Transactions on Software Engineering*, 17(3):259–273.

Berthomieu, B. and Menasche, M. (1983). An enumerative approach for analyzing time petri nets. In *Proceedings IFIP*, pages 41–46, Paris. Elsevier Science Publishers.

Berthomieu, B. and Vernadat, F. (2006). Time petri nets analysis with tina. In *QEST: Third International Conference on the Quantitative Evaluation of Systems (QEST 2006)*, pages 123–124, Riverside, California, USA.

Chebbi, I., Dustdar, S., and Tata, S. (2006). The view-based approach to dynamic inter-organizational workflow cooperation. *Data Knowl. Eng.*, 56(2):139–173.

Chebbi, I. and Tata, S. (2005). *CoopFlow*: A framework for inter-organizational workflow cooperation. In *On the Move to Meaningful Internet Systems Conferences*, pages 112–129, Agia Napa, Cyprus.

Chebbi, I. and Tata, S. (2007). Workflow abstraction for privacy preservation. In *Web Information Systems Engineering Workshops*, pages 166–177, Nancy, France.

Diaz, G., Navarro, E., Cambronero, M.-E., Valero, V., and Cuartero, F. (2007). Testing time goal-driven requirements with model checking techniques. In *ECBS ’07: Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, pages 503–514, Washington, DC, USA.

Eder, J., Gruber, W., and Panagos, E. (2000). Temporal modeling of workflows with conditional execution paths. In *DEXA ’00: Proceedings of the 11th International Conference on Database and Expert Systems Applications*, pages 243–253, London, UK. Springer-Verlag.

Eder, J. and Panagos, E. (2000). Managing time in workflow systems. In *in Workflow Handbook 2001, Layna Fischer (Ed.), Future Strategies Inc., 2001*, pages 109–132.

Godary, K. (2008). Lpt : Little parametric tool, outil pour la validation d'une borne temprelle paramétrée. *Sixième Conférence Internationale Francophone d'Automatique (CIFA'08)*.

Kazhamiakin, R., Pandya, P., and Pistore, M. (2006). Representation, verification, and computation of timed properties in web. In *Proceedings of the IEEE International Conference on Web Services*, pages 497–504, Washington, DC, USA.

Klai, K., Tata, S., and Desel, J. (2009). Symbolic abstraction and deadlock-freeness verification of inter-enterprise processes. In *Business Process Management, 7th International Conference*, pages 294–309, Ulm, Germany.

Merlin, P. M. (1974). *A study of the Recoverability of Computing Systems*. Technical report ♯58 (phd thesis), Computer Science Department, University of California at Irvine.

Pezze, M. and Young, M. (1999). Time petri : A primer introduction. In *Tutorial presented at the Multi-Workshop on Formal Methods in Performance Evaluation and Applications*, pages 41–46, Zaragoza, Spain.

Tata, S., Klai, K., and M'Bareck, N. O. A. (2008). Coopflow: A bottom-up approach to workflow cooperation for short-term virtual enterprises. *IEEE T. Services Computing*, 1(4):214–228.

Toussaint, J., Simonot-Lion, F., and Thomesse, J.-P. (1997). Time constraints verification methods based on time petri nets. In *Proceedings of the 6th IEEE Workshop on Future Trends of Distributed Computing Systems*, page 262, Washington, DC, USA.