Temporal Consistency of View Based Interorganizational Workflows

Johann Eder[1,2] and Amirreza Tahamtan[2]

[1] Alpen-Adria University of Klagenfurt, Dept. of Informatics Systems, A-9020 Klagenfurt, Austria
eder@isys.uni-klu.ac.at,
[2] University of Vienna, Dept. of Knowledge and Business Engineering, Rathausstrasse 19/9, A-1010 Vienna, Austria
amirreza.tahamtan@univie.ac.at

Abstract. Interorganizational workflows are a major step in automating B2B electronic commerce and to support collaborations of organizations on the technical level. Process views are an important conceptual modelling approach for interorganizational workflows as they allow interaction and communication while internal and private parts of the process can be hidden. However, it essential to guarantee that an interorganizational workflow is free of conflicts and the overall quality assurances of the whole workflow can be achieved. This paper proposes an approach for checking temporal consistency of interorganizational workflows crossing boundaries of organizations.

Key words: Interorganizational Workflow, Workflow View, Temporal constraints, Conformance, Consistency

1 Introduction

Automation of tasks and processes plays an essential role in order to provide cheaper services with a better quality. In recent years, workflow management systems (WfMS) have provided an effective and powerful tool for this aim. It is necessary for organizations to build short or long term cooperations with other organizations to reach the overall goal of a business process. Spread of the internet, as a means of communication , provides a powerful infrastructure for interorganizational workflows. Such workflows enable autonomous organizations, which may be geographically distant, to cooperate with each other. A challenging point when constructing such workflows is the balance between autonomy and cooperation. On one hand, organizations must reveal some information to the business partners necessary for communication, monitoring, tracking purposes, etc. and on the other hand organizations want to hide their internal process logic to protect their know-how and improve their business secrecy, or simply keep the possibility for quickly improving their their business processes or adapting their business processes to changing environments without having to go through a process of changing collaboration agreements. For external partners, as well, it is neither necessary nor desirable to have access to all parts of the provider's internal workflow as they do not want to be overloaded by unnecessary data and

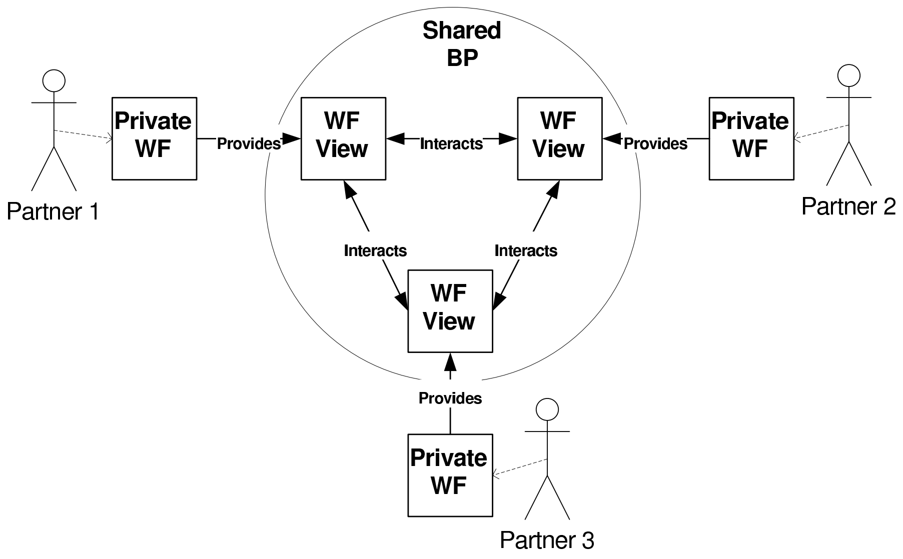

Fig. 1. An interorganizational workflow composed of views

messages and are interested in those parts of the workflow which address them. Additionally, it is a well established principle in software engineering to keep the coupling between parts as loose as possible.

Workflow views provide a mean for this aim. Views define the accessible and visible parts of a process for external partners. A view can be a subset of the (activities) of the original workflow or represent the original workflow in an abstracted or aggregated fashion. External users communicate and interact with the view and not with the private executable workflow. Views are in charge of redirecting the data and information to the executable workflow and also forwarding them to the external user. In case of Business-to-Customer (B2C) the external user is normally a human user or an agent and in case of Business-to-Business (B2B) application the external user is another organization which in turn may interact and take part in the interorganizational workflow through its view. By using views, organizations are able to expose as little information as possible but sufficient for the communication and interaction to reach the goals of the business process. In this work we focus on Business-to-Business applications and assume that each organization takes part in the interorganizational workflow through its workflow view i.e. the shared business process is an integration of the partners' views. Partners communicate by their views and not with their private workflows. Fig. 1 depicts this scenario. Three partners take part in the shared business process, each with their own private workflow. Partners provide a view on their private workflow and the interaction with other partners in the shared business process proceeds through views of the partners. In such a case, it must be ensured that views are conformant with each other and can interact without conflict. Several issues such as structural conformance, data flow conformance, messaging conformance and temporal conformance must be checked in order to ensure the consistency and conformance of the overall interorganizational workflow. In this work we study the temporal conformance of workflow views in an interorganizational setting to ensure that interactions are done in a timely manner with respect to local and global constraints. A temporally conformant interorganizational workflow enables partners to execute without violating temporal constraints such as explicitly assigned deadlines and also reduces the cost of process because fewer temporal exception must be raised and handled accordingly. Such a mechanism is a helpful tool in hand of process designers and managers to foresee the possible upcoming temporal violations and react to them in a predictive manner. In this paper only the case with two partners is considered but in a straightforward manner the concepts can be extended and applied when more than two partners are involved.

2 Related Work

[1] applies views to enable interaction between autonomous workflows in an interorganizational setting. Partners in a business process own their private workflows and take part in a shared business process, here called coalition workflow, through views. A view is an abstraction of the corresponding private workflow and reflects the communication requirements of the coalition workflow. The views outsource the execution to the according private workflow. A tightly coupled approach between private workflows and its view(s) based on state dependencies and a loosely coupled between views in the coalition workflow based on control flow dependencies is applied. This work mainly discusses the views from an state perspective and is silent on correctness issues of views or how views may be built correctly. [2] propose an order preserving approach for constructing views. This work also applies an aggregation approach for constructing virtual activities out of base activities of the private workflow. The authors present three rules for construction of process views and an algorithm for automatic generation of possible process views with respect to these rules. [3] can be seen as a combination of [1, 2]. The construction of process views originate from the authors' previous work [2] enriched with the state mapping used in [1]. However the authors use slightly different terminology for naming the states. The state of the virtual activity is a function of the states of its member base activities which can be used for monitoring the state of the virtual activities by other partners. [4] proposes an approach based on software oriented architecture (SOA) for interconnection of workflows. A semantic registry for publishing and discovery of services is proposed, enabling other organizations to build a cross-organizational cooperation by finding and binding to other partners. Views contain only cooperative tasks. i.e. tasks that either send or receive data to/from external workflows. The interorganizational workflow consists of virtual activities which are in charge of

transferring data to/from executable activities. A virtual activity is a subset of cooperative activities. A trusted third party is responsible for monitoring the interaction policies. This work can be seen as an effort for enabling interorganizational workflows for a dynamic setting where partners and their interconnections are not predefined. However, the need for negotiation and a contract for setting up the interaction policy and identification of public virtual activities still remains and is only shifted after finding the partners from the registry. Besides, the assumption that a view consists of only cooperative tasks, may not be sufficient in all scenarios. [5] propose to use views for interoperability in cross-organizational workflows. The balance between security and trust is considered to be achieved through views. In other words, views restrict the access on the workflow and conceal the internal, private or unnecessary information. A view is defined as a structurally correct subset of workflow definition. The authors consider mainly access rights on objects associated with a view. Moreover, they do not describe how views can be constructed from a workflow and an integrated view in an interorganizational interoperation is missing. [5] presents a technique for identification of relevant tasks included in a view. With respect to a given role, some metrics for identification of most relevant tasks are defined . Workflow operations are used as criteria to evaluate role-task relevance. The metrics are extracted and calculated by analyzing the workflow logs. The proposed technique can be used for an optimized construction of views. However, a history of interaction between external user and the private process should be at hand. [7] introduces a methodology for decomposition of complex processes in scientific domain and building view based on flows. The interactions among flows are triggered by external messages. Each flow can have multiple views. The views of a flow are built based on the analysis of the required incoming messages, its immediate responses and the dependency between data and messages. The overall process is an integration of views of all partners. [8] introduces an object oriented approach for workflow views called the object deputy model. The definition and concept of views is taken from [5]. A deputy object has its own persistent identifier and is a subclass of its source object. A bilateral link between objects and one of its deputies allows for inheritance and update propagation. In this work views are used for two main purposes: restriction of access and composition. Authors do not mention how views can be constructed for mentioned applications. A workflow restriction view is defined the same as a workflow view, a structurally correct subset of a workflow definition. A workflow composition view is a virtual workflow composed of components from different workflows, which may span across organizational boundaries.

The time management concepts come from a related field, namely workflow management research. One of the eraliest works on time properties is [9]. Allen describes a temporal representation using the concept of temporal intervals and introduces a hierarchical representation of relationships between temporal intervals applying constraints propagation techniques. This work describes thirteen ways in which an ordered pair of interval can be related. [10] provides a methodology for calculating temporal plans of workflows at design time, instantiation time and run time. It considers several temporal constraints like lower bound, upper bound and fixed-date constraints and explains how these constraints can be incorporated. Moreover, a model for monitoring the satisfaction of temporal constraints during run time is provided. Eder et al. in [11] present a model for calculation of temporal plans and propose some algorithms for calculation and incorporation of time constraints. [12] provides a technique for modeling and checking time constraints whilst conditional and parallel branches are discriminated. In addition, an unfolding-method for detection of scheduling conflicts is provided. Marjanovic in [13] represents the notions of duration space and instantiation space and describes a technique for verification of temporal constraints in production worklfows. Our approach is complementary to that introduced in [13] in the way that a temporal plan for execution of all activities is calculated.Temporal aspects of web services, a very realted field to workflow research, have been studied in [14, 15, 16]. [14] uses temporal abstractions of business protocols for their compatibility and replaceability analysis based on a finite state machine formalism. [15, 16] exploit an extension of timed automata formalism called Web Service Time Transition System (WSTTS) for modeling time properties of web services. [17] provides an architecture for interorganizational workflows and [18] provides an algorithm for checking the structural conformance of the participating partners' workflows.

3 Workflow Views

We assume that each partner has a private workflow which is only visible to the owner of the workflow and external parties have no knowledge about the process structure and internal logic of the private workflow. In order to provide the necessary information for communication and interaction with other partners and/or service requester, the owner of the workflow provides views for each partner. In other word, a workflow can have many workflow views each for one partner or for a group of partners. In this way the workflow is able to interact with external parties whilst protecting private aspects of the process. In this paper we assume two ways for constructing views: abstraction and aggregation.

By application of abstraction operator (called τ-operator in process algebra [19]) it is possible to make parts of the process invisible to other external observer. This operator provides a mean for construction of views. By application of this operator, parts of the process that do not contribute to the interaction with another partner, are not interesting for external partner or are intended to be hidden because of privacy issues, can made unobservable and the rest of the process can be exposed as the view on the process. Left part of fig. 2 depicts an example. The internal workflow (left most part) consists of four activities. After receipt of the request, the request is processed, the history of the buyer profile is updated and the results are sent back to the requester. After making two activities invisible, the requester can interact with the view of the workflow containing two tasks *Receive Request* and *Send Results* which are sufficient for interaction between the requester and the provider.

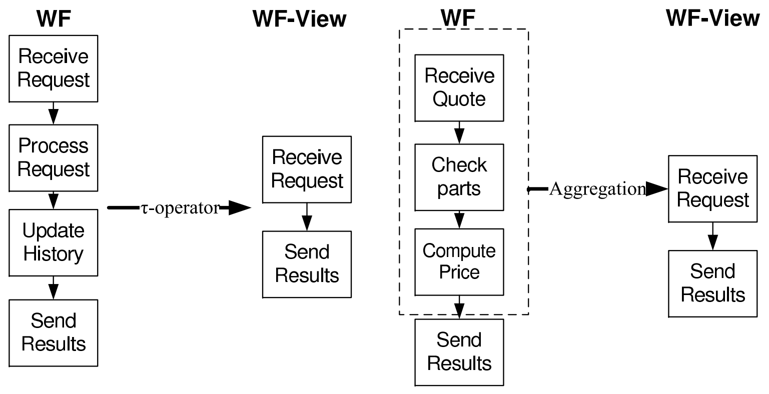

Fig. 2. Application of abstraction operator(left part), Aggregation (right part)

Another approach is using virtual activities for constructing views. In this approach some executable activities of the private workflow can be grouped into a so-called virtual activity. Virtual activities are in charge of forwarding and receiving data to and from executable activities of the private workflow. The right part of fig. 2 illustrates an exemplary application of aggregation. In this simple example, at first the quote is received by the workflow. It is checked which parts are needed for the production, the price is computed and finally the result is sent back to the external partner. To construct a view, the first three activities are grouped into the virtual activity "Receive Request". The service requester interacts with the view and the view sends the results to the executable activities and also to the external partner. Note that there is not a unique way of constructing views. Executable activities can be grouped in many different ways into virtual activities. It may depend on the interacting roles, constraints of the workflow owner and the context of the interorganizational workflow.

4 Temporal plans of workflows and their views

Assume a situation when a view of a workflow is defined and this view takes part in an interorganizational workflow. It is necessary to calculate the temporal execution plan of the views. In order to avoid temporal conflicts and exceptions it must be clear for the other partners in which temporal interval they can send messages and in which temporal interval they can expect messages. We assume that each workflow has a deadline and based on the calculation of the temporal plan of the private workflow, we calculate the temporal plan of its views. In other words we calculate a valid interval in which an activity, executable or virtual can execute.

4.1 Model description

Time is considered as discrete values expressed in some basic units like Minutes (M), Seconds (S) etc. The basic concepts used for calculation of the temporal plans come originally from the field of project management and operations research [20] like CPM and PERT. There are two kinds of temporal constraints:

– **Implicit constraints** are derived implicitly from the structure of the process e.g. an activity can start execution if and only if all its predecessors have finished execution.
– **Explicit Constraints** e.g. assigned deadlines, can be set explicitly by the process designer or enforced by law, regulations or business rules

As the basic modeling language we used timed activity graphs or timed graphs [10, 11]. They are familiar workflow graphs where nodes correspond to activities and edges the dependencies between activities, enriched with temporal information. Fig. 3 shows an example of a node.

Activity Name	Activity Duration
Best Case Earlies Possible Start	Best Case Latest Allowed End
Worst Case Earlies Possible Start	Worst Case Latest Allowed End

Fig. 3. A node in the timed graph

All activities have durations. *a.d* denotes the duration of an activity *a*. At the first use of the model an estimation of the activity durations, e.g. expert opinion, may be used. Later, workflow logs can be mined for actual activity durations. In this work deterministic values for activity durations are used. We are aware that activity durations may vary in real life applications. However, we use fixed values for clarification of the concepts and avoid the math involved in probability distributions. We calculate an interval in which an activity may execute. This interval is delimited by *earliest possible start* (eps-value) and *latest allowed end* (lae-value). $a.eps$ denotes the *eps*-value of an activity a and is the earliest point in time in which an activity a can start execution. $a.lae$ represents the latest point in time in which an activity a can finish execution in order to hold the assigned deadline. Both *eps* and *lae* values are calculated for *best case* and *worst case.* If there is an XOR-split in the workflow, there are multiple paths to be chosen and based on some evaluated conditions at run-time one of the available paths is executed. The best case is given, if the shortest path is executed and we have the worst case when the longest path is executed. It is possible that XOR-split has other branches whose length lies between best and worst cases. In this paper only best and worst cases are considered. If all branches of an XOR-split have the same length, both cases have the same *eps* and $lae - values$. *eps*-values are calculated in a forward pass by adding the *eps*-value of the predecessor to its duration. For example $b.eps = a.eps + a.d$ if an activity a is a predecessor of

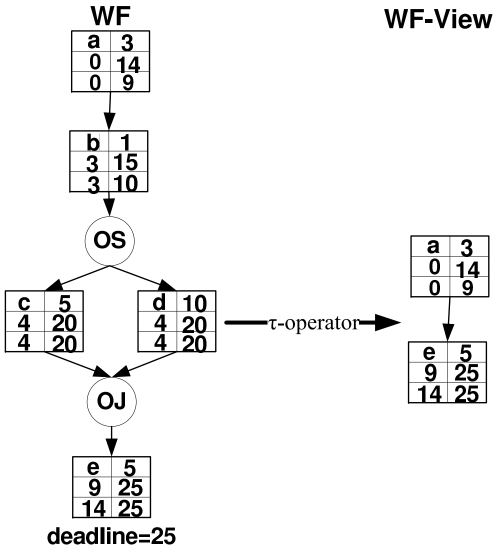

Fig. 4. Temporal plan of a WF and its view by application of τ-operator

an activity b. If an activity a has multiple predecessors, e.g. if an activity a is an immediate successor of an AND-join, the maximum of eps-value of predecessors of a is taken into account. The $eps - value$ of the first activity or the set of first activities are set to 0.

In contrast to the eps-values, lae-values are calculated in a backward pass by subtracting the lae-value of the successor from its duration, e.g. $a.lae = b.lae - b.d$ if an activity b is a successor of an activity a. If an activity a has multiple successors, e.g. if activity a is an immediate successor of an AND-split, the minimum of eps-value of predecessors of a is taken into account.The lae-values of the last activity or the set of last activities are set to the assigned deadline. If no deadline is assigned, length of the longest path may be used as deadline. In this case the critical path has no buffer time available.

In addition to eps and lae-values we can define *earliest possible end* (epe-value) and *latest allowed start* (las-value) for the activities. However, given activity durations they may be calculated interchangeably applying the following formulas: $a.epe = a.eps + a.d$ and $a.lae = a.las + a.d$. When parallel structures are present in the model, the longest path (worst case calculation) between the split and join nodes is always considered. Hence, best case has the same temporal values as the worst case. The reason is that the parallel join must wait in any case for the longest branch of the parallel structure to commit. For a more detailed discussion refer to [12]. Given the deadline of 25, left Part of Fig. 4 shows the calculated temporal plan of the depicted workflow.

4.2 Calculation of timed graphs of views

As noted, we assume two ways for constructing views: application of τ-operator and aggregation. If τ-operator is applied on the executable private workflow process, there is no need to recalculate the timed execution plan of the view. In this case, after calculation of the temporal plan of the private process, the calculated values can be directly taken over for the activities contained in the view. See Fig. 4 for an example. If aggregation is applied, the temporal values of the virtual activities in views can be calculated in a straightforward manner. The

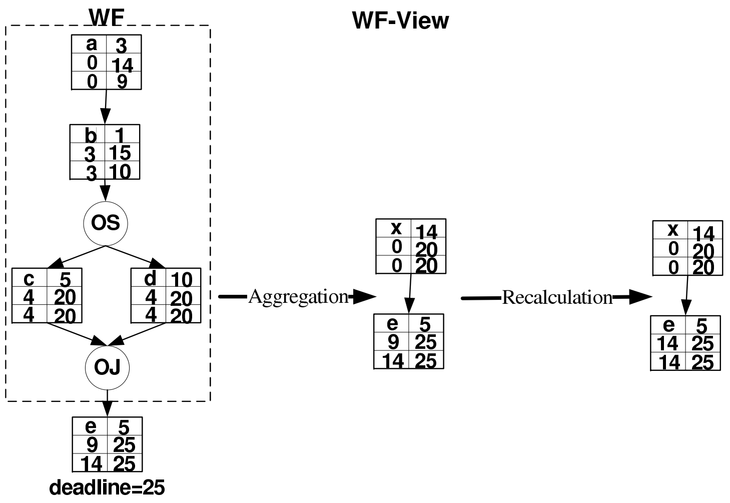

Fig. 5. Temporal plan of a view by application of aggregation

duration of the virtual activity is sum of the durations of the executable activities contained in it. Note that the parallel or join structure can be considered as one activity whilst the length of its longest path is taken as its duration. This yields to a worst case estimation of the duration of the virtual activity. But for other partners it is more important to know in which time interval they can interact with the workflow with a valid temporal window. Hence, worst case duration for the virtual activities is a reasonable estimation. This yields always to a temporally valid interaction with the view and therefore guarantees the correct temporal behavior of the flow. The eps-value of the virtual activity is the eps-value of its first executable activity or the minimum value of the set of its first executable activities. The eps-value of the virtual activity is the eps-value of its last executable activity or the maximum value of the set of its last executable activities. The temporal values of other activities that are not contained in another virtual activity can be taken over directly from temporal plan of the private workflow. Figure 5 shows an example for this case. As can be seen in Fig. 5, the virtual activity x has a duration of 14 time units, where as the activity e in the view has an eps-value of 9 time units. It implies that the virtual activity e in the view of Fig. 5 can start execution before the virtual activity x (its direct predecessor) has finished execution, which is a violation of the structural constraints. The reason is that the virtual activity x reflects its worst case values whilst eps-value of activity e is its best case value. To resolve this issue, it is necessary to recalculate the *eps*-values of the view. Note that the *lae*-values remain unaffected. This can be done in the same way as explained before (the forward pass calculation). The end results are depicted in the right most part of the Fig. 5.

5 Temporal consistency of views

In an interorganization workflow several partners take part in the public shared process. We assume that each partner participates in the interorganizational workflow with its view. Fig. 6 illustrates such a scenario. This workflow consists of there partners, a buyer, a seller and a supplier. The dashed lines between virtual activities demonstrate a message exchange and the number above them shows the ordering of the message sequence. This example contains interactions between a buyer and a seller (right part) and buyer and a supplier (left part). For sake of brevity the private workflows of the participating partners are omitted and only the views are illustrated.

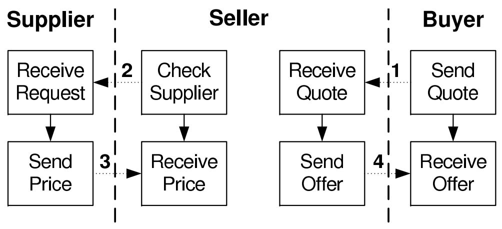

Fig. 6. An interorganizational workflow

The buyer sends a request for quote to the seller (message 1), the seller checks if the needed parts for the production of the item is deliverable by its supplier (message 2), if the parts required for manufacturing are deliverable by the supplier, the supplier sends the price to the seller (message 3). After receipt of the price from the supplier, the seller calculates the whole price and sends it back to the buyer (message 4). In order to check if two activities of different views are temporally consistent, it must be checked if the execution intervals of both corresponding activities overlap. Corresponding activities are in this sense those activities that communicate with each other. In other words, activities that are sender or receiver of the same message e.g. activities "Send Quote" of buyer and "receive Quote" of seller or activities "Check Supplier" of seller and "Receive Request" of supplier. Put it another way, in order to check the temporal consistency of two corresponding activities it must be checked if there is any temporal interval in which both activities can execute. Let $x.[eps, lae]$ denote the valid execution interval of an activity x contained in a view. This interval can be calculated based on the temporal execution plan of the private executable workflow as explained earlier. Let $wc.x.[eps, lae]$ and $bc.x.[eps, lae]$ denote the execution interval of an activity x for worst case and best case respectively. Let x be an activity and y its corresponding activity in another view. Five cases can be identified:

- $x.[eps, lae] \bigcap y.[eps, lae] = \varnothing$, these activities can not interact.
- $wc.x.[eps, lae] \bigcap wc.y.[eps, lae] \neq \varnothing$, in any case both activities can interact
- $bc.x.[eps, lae] \bigcap wc.y.[eps, lae] \neq \varnothing$, activity y can interact in any case but activity x only at its best case
- $wc.x.[eps, lae] \bigcap bc.y.[eps, lae] \neq \varnothing$, activity x can interact in any case but activity y only at its best case
- $bc.x.[eps, lae] \bigcap bc.y.[eps, lae] \neq \varnothing$, in this case, both activities can interact only at their best cases

If there is no overlap of the execution interval of corresponding activities, those activities are temporally not consistent. Note that there are some intermediate or mixed forms between best case and worst case that we do not consider. The proposed approach for checking temporal consistency between two activities can be extended in a straightforward manner to other forms of communications e.g. if an activity has multiple receivers. In such a case it must be checked if the intersection of more than two temporal intervals is not empty and in which case they can communicate, i.e. worst case or best case. In order to decide if two views are temporally consistent, all pairs of corresponding activities must be temporally consistent. Finally an interorganizational workflow is temporally consistent if all of its participating views are temporally consistent. Other issues for full consistency like messaging conformance, data flow conformance and structural conformance are out of scope of this paper.

6 Conclusion

Consistency of interorganizational workflows are important issues to be taken into account when designing, negotiating and executing such a coalition among different organizations. In this paper we have presented a technique for temporal consistency checking of an interorganizational workflow whilst each partner takes part with its view. This technique helps process designers to check the consistency before executing and hence reducing the cost of process as fewer exceptions must be raised and handled because of temporal violations.

References

1. Schulz, K.A., Orlowska, M.E.: Facilitating cross-organisational workflows with a workflow view approach. Data and Knowledge Engineering, 51(1), pp. 109–147. Elsevier (2004)

2. Liu, D.R., Shen, M.: Workflow modeling for virtual processes: An order-preserving process-view approach. Information Systems, 28(6), pp. 505–532. Elsevier (2003)
3. Liu, D.R., Shen, M.: Business-to-business workflow interoperation based on process-views. Decision Support Systems, 38(3), pp.399–419. Elsevier (2004)
4. Chebbi, I., Dustdar, S., Tata, S.: The view-based approach to dynamic inter-organizational workflow cooperation. Data and Knowledge Engineering, 56(2), pp. 139–173. Elsevier (2006)
5. Chiu, D.K.W., Cheung, S.C., Karlapalem, K., Li, Q., Till, S.: Workflow View Driven Cross-Organizational Interoperability in a Web-Service Environment. In: Proc. of Web Services, E-Business, and the Semantic Web, CAiSE 2002 International Workshop, pp. 41–56. Toronro, Ontario, Canada (2002)
6. Liu, D.R., Shen, M.: Discovering role-relevant process-views for disseminating process knowledge. Expert Systems with Applications, 26(3), pp. 301–310. Elsevier (2004)
7. Li, Q., Shan, Z., Hung, P.C.K., Chiu, D.K.W., Cheung, S.C.: Flows and Views for Scalable Scientific Integration. In: Procs. of InfoScale '06, ACM International Conference Proceeding Series, Vol. 152. Hong Kong (2006)
8. Shan, Z., Long, Z., Luo, Y., Peng, Z.: Object-Oriented Realization of Workflow Views for Web Services- An Object Deputy Model Based Approach. LNCS, vol. 3129, pp. 468–477. Springer Verlag (2004)
9. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM, 26(11), pp. 832–843. ACM, New York, NY, USA (1983)
10. Eder, J., Panagos, E.: Managing Time in Workflow Systems. In: Lawrence, P. (eds.) WfMC WorkFlow Handbook 2001. J. Wiley & Sons (2001)
11. Eder, J., Panagos, E., Rabinovich,M.: Time Constraints in Workow Systems. In: Proc. of the 11th International Conference on Advanced Information Systems Engineering (CAiSE), pp. 286–300. Heidelberg, Germany (1999)
12. Eder, J., Gruber, W., Panagos, E.: Temporal modeling of workflows with conditional execution paths. In: 11th In ternational Conference on Database and Expert Systems Applications (DEXA 2000). London, Greenwich (2000)
13. Marjanovic, O.: Dynamic Verification of Temporal Constraints in Production Workflows. In: Proc. of the Australasian Database Conference (2000)
14. Benatallah, B., Casati, F., Ponge, J., Toumani, F.: On Temporal Abstractions of Web Service Protocols. In: Proc. of CAiSE Forum, (2005)
15. Kazhamiakin, R., Pandya, P., Pistore, M.: Timed Modelling and Analysis in Web Service Compositions. In Proc. of ARES'06. (2006)
16. Kazhamiakin, R., Pandya, P., Pistore, M.: Representation, Verification, and Computation of Timed Properties in Web Service Compositions. In: Proc. of ICWS'06. (2006)
17. Eder, J., Lehmann, M., Tahamtan, A.: Choreographies as Federations of Choreographies and Orchestrations. In: Proc. of International Workshop on Conceptual Modeling of Service-Oriented Software Systems. Tuscon, AZ, USA (2006)
18. Eder, J., Lehmann, M., Tahamtan, A.:Conformance Test of Federated Choreographies. In: Proc. of the 3rd International Conference on Interoperability for Enterprise Software and Applications (I-ESA'07). Madeira Islands, Portugal (2007)
19. Bergstra, J.A., Klop, J.W.: Algebra of communicating processes with abstraction. Comput. Sci., 37, pp. 77-121 (1985)
20. Philipose, S.: Operations Research A Practical Approach. Tata McGrawHill, New Delhi, New York (1986)