2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops

Privacy Preservation Through Process Views

Amirreza Tahamtan
Dept. of Software Technology & Interactive Systems
Vienna University of Technology
Vienna, Austria
e-mail: tahamtan@ifs.tuwien.ac.at

Johann Eder
Dept. of Information Systems
Alpen-Adria University of Klagenfurt
Klagenfurt, Austria
e-mail: eder@isys.uni-klu.ac.at

***Abstract*—Privacy and security are major concerns for partners in B2B and B2C applications. Service providers want on the one hand isolate their private process in order to protect their business logic and improve their privacy and on the other hand they have to expose (at least some parts of) their private process to other partners in order to enable communication, interaction and data exchange. Process Views are powerful mechanisms which serve several purposes: improvement of privacy and security, process abstraction, separation of public and private parts of business processes, interface definition for interorganizational workflows, loosening the coupling between process components, etc. Following an extensive survey of approaches for process views, we present a correctness criterion for views and a novel technique for the construction of process views which can be used for improvement of privacy and security. The formal definition of the view construction operators allows to prove that all views constructed with these operators are correct. In addition, by application of workflow views, changes in a private process can be kept local such that the interaction with other partners are not affected.**

***Keywords*-Process Views, Aggregation, Abstraction, Correctness, Privacy**

I. Introduction

In the always more connected world in which people and companies interact and cooperate through internet a major concern and issue is preserving privacy. It is obvious that companies cannot fully shield their internal processes from other partners because being able to interact implies exposure of information to some extent. For example a transport company like DHL must allow its costumer to track their parcels. On the one hand, organizations must reveal some information to other partners which are necessary for communication, monitoring, tracking purposes, etc. and on the other hand they want to hide their internal process logic to protect their know-how and improve their business secrecy and privacy. By exposing only limited and necessary information it can be ensured that business details remain private. The service provider can decide which parts of its internal process are private and must be protected from others. For external partners, as well, it is neither necessary nor desirable to have access to all parts of a provider's internal workflow as they do not want to be overloaded by unnecessary data and uninteresting messages. External partners are interested in those parts of a workflow which address them. For instance costumers are not interested in how Amazon handles its business with its providers as long as they receive their ordered items.

Process views provide a mean for improving privacy in B2B as well as B2C application. 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 process or represent the original process in an abstracted or aggregated fashion. External users communicate and interact with the view and not with the private executable process. Views are in charge of redirecting the data and messages to the executable process as well as forwarding them to external users. By using views, organizations are able to show as little information as possible but sufficient for the communication and interaction to reach the goals of a business process. In this way the privacy of business partners can be considerably improved.

By application of views, as long as the view remains the same, organizations can change the underlying process without any concern and external partners also just need to conform to the view and not to its underlying process. As long as changes in a private process do not affect its view, these changes can be kept local and there is no need for modification of the view and therefore organizations can be sure that external partners can still communicate in a conformant and consistent manner. Process views have applications in many domains such as interorganizational workflows [16], [17], [6], cooperation [14], interconnection [4] and interoperability [5] of autonomous entities and many more.Views are powerful abstractions with a long and successful history in databases and modeling [18]. Views allow to define submodels for specific purposes, abstract from the global model, reduce the degree of coupling between software components, promoting logical independence of modules which are interfaced through views.

In this paper we introduce the notion of views to processes modeled as workflows (or web-service orchestrations, business processes, etc). The abstractions provided by views are also very helpful for process models [8]. For example, views help to balance autonomy and cooperation in cooperating business processes.

The contribution of this paper is introduction and formalization of two operators (abstraction and aggregation) and techniques for construction of workflow views by these two operators. By application of abstraction, parts of a process can be made invisible and the rest can be exposed as a view on the process to external partners. Aggregation is used for hiding the internal structure of a group of activities. In this way, communicating partners can interact with the process without knowing the details about the internal structure of the process. This work presents how these two operators should be applied in order to construct correct workflow views. As long as these criteria are satisfied, application of these operators results in correct workflow views. Consequently, process designers do not need to check the correctness between a workflow and its views. We define these operators for the full-blocked workflows supporting the usual patterns of sequence, AND-split, AND-join, XOR-split, XOR-join and structured loops. We have chosen full-blocked workflows as the modeling paradigm because by using full-blocked workflows structural errors such as deadlocks can be avoided [9]. Besides full-blocked workflows are supported by industry standards such as BPEL and OWL-S.

This paper is organized as follows: Section II provides an overview on related works on workflow views and compares the approach of this work with those of other authors. Section III provides a formal definition of correct workflow views and the workflow model that is used throughout this paper. Section IV represents the operators that are used for construction of workflow views and discusses their properties. After explaining how these operators can be used for construction of workflow views in section V we provide some concluding remarks.

II. Related Work

This section provides an overview on related work on workflow views and contrasts our approach to other concepts.

[14] proposes workflow views for communication and cooperation of autonomous workflows in an interorganizational setting. A partner owns his private workflow which is only visible to him. Partners participate in a shared business process through workflow views. A workflow view is an abstraction of the corresponding private workflow and reflects the communication requirements of the shared workflow. When running the shared workflow, the workflow views outsource the execution to the according private workflows. The cross-organizational workflow is composed of the private workflows of the participating partners. Further, an architecture for cross-organizational workflows is proposed. Our work proposes two techniques for constructing views, which can be applied in an interorganizational setting. Besides, correctness criteria for views are considered.

[11] proposes an order preserving approach for constructing views. A workflow view is composed of virtual activities. A virtual activity is an aggregation of base activities of a workflow (cf. complex activity). In this approach, a workflow view is not a subset of the activities of a workflow (e.g. by application of the abstraction operator) rather each virtual activity contains one or more base activities. In other words, a view contains all activities of the base process but in a new grouping. The proposed approach may not be fully sufficient, as all activities of the base process must be contained in a view. Our work is complementary to [11] by introduction of an additional operator that enables presentation of only selected parts of a workflow in its view.

[12] can be seen as a combination of [14], [11]. The construction of process views originates from the authors' previous work [11] enriched with the state mapping used in [14]. However the authors use slightly different terminology for naming the states.

[4] proposes an approach based on 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. Participants take part in the interorganizational workflow with their views. A view contains only cooperative tasks. i.e. tasks that either send data to external workflows or receive data from external workflows. The interorganizational workflow consists of virtual activities. Virtual activities are in charge of transferring data to/from executable activities. A virtual activity is a subset of cooperative activities. The cooperation between partners is handled and modeled by cooperation policies whose execution is monitored by a third party. our approach is more general than [4]. Not only communication activities but also other activities which may be necessary for cooperation can be included in a view.

[5] proposes the application of workflow 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 a workflow and conceal the internal, private or unnecessary information. Access restriction is considered in our proposed approach. Because such decisions, e.g. which partner can access an activity, depend heavily on the context, it is left to the user to decide which activities are included in a view for a partner.

[13] based on the framework provided in [5] presents a technique for identification of relevant tasks included in a view for a specific role. Authors use some metrics for tasks (regarding a given role) in a workflow to find the most relevant tasks for the according role. Such metrics are mined from workflow logs which implies that a private workflow must be exposed to specific roles before such data can be mined for construction of views. Our approach assumes that workflow owners use views to protect their business know-how encoded in the private workflow and they do not want to expose their private workflow to external users.

[10] introduces a methodology for decomposition of complex processes in scientific domain and building views based on flows with the aim of an increased flexibility and scalability by separation of flows. Moreover, this separation yields in a better modifiability of the process for different situations and applications. Note that only predefined exceptions can be handled by exception flows and unknown exception are not considered in this work. The interactions among flows are triggered by external messages. Each flow can have multiple views. The authors propose to build a view of a flow based on the analysis of the required messages. [10] uses views as a tool for decomposition of a complex workflow into some sub-flows. Our work applies views mainly for the interaction,

cooperation and access restriction.

[15] introduces an object oriented approach for workflow views called the object deputy model. The definition and concept of workflow views is again taken from [5]. [15] sees views as a subset of a flow which can inherit properties from the original flow. Views are used for two main purposes: restriction of access and composition. It is not discussed how workflow views can be constructed for mentioned applications. The focus of our work is on construction and correctness of views.

Eshuis and Grefen in [7] provide an approach for constructing optimized process views. Their approach is somehow similar to the approach presented in our work. However, the main goal of [7] is cooperation and we are focused on privacy. They apply a two-phase approach based on inheritance for construction of process views. In the second phase it is necessary that process views be exposed to the external partner. In our approach in order to improve the privacy of the service provider, there is no need that the internal process or any of it intermediate views be exposed to external partners. It is solely up to the service provider to decide which parts of the process he wants to expose to other partners and no input from other partners are needed. [7] provides some rules for constructing views but do not provide any proof that application of these rules results in correct views. We formally prove that our approach produces correct process views.

Bobrik et al. in [3], [2] propose a technique for constructing views aimed at process visualization. The limitation of their approach is that they consider no loops and provide no proof for correctness verification of resulting process views. Our approach is complementary in the sense that we consider loops and provide a proof for correctness of the views.

Table I at the end of this work compares the related works on workflow views with our approach.

Our approach presents a new operator (abstraction) for construction of views and ensures that application of view construction operators results in correct views.

III. Correctness of Views

In order to provide only the necessary information for communication and interaction with other partners and hiding the private parts of a process, a workflow owner provides views. A workflow can have many views, one for each partner, for each role of a 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.

Definition 1: **(Workflow Model)**

A workflow is modeled as a directed acyclic graph $G = (N, E)$, where N denotes the set of nodes and E the set of edges. Nodes correspond to activities and control nodes and edges to the dependencies between activities and control nodes. All workflows are full-blocked, i.e. each split-node or start of loop has a counterpart join-node or end of loop and each outgoing path of a split node or start of a loop ends eventually in a join node or end of a loop and vice versa.

A control node is either an AND-split (AS), AND-join (AJ), XOR-split (XS), XOR-join (XJ), start of loop (LS) or end of loop (LE). The workflow model is capable of representing all typical control flow structures. The activities contained between LS and LE are repeated as long as the exit condition is not satisfied. If the exit condition is satisfied, the loop terminates and the execution of the flow continues with the immediate successor activity of LE. Figure 1 illustrates an example of our workflow model.

In order to construct correct views on workflows it is necessary that activities in a view have the same ordering as in the underlying workflow. In other words a view on a flow can not change the ordering of the activities of the underlying workflow. Such a view is called an order-preserving view.

Definition 2: **(Correctness of Views)**

A workflow $G\prime = (N\prime, E\prime)$ is a correct view of a workflow $G = (N, E)$ if and only if the following properties hold:

(a) $G\prime$ is a valid full-blocked workflow definition

(b) The nodes contained in $G\prime$ are a subset of the nodes of G or represent a subset of the nodes of G

(c) $\forall$ nodes $a, b \in N \bigcap N\prime : [a > b]_{G\prime} \Leftrightarrow [a > b]_G$, where $[a > b]_G$ denotes that there is a path from node a to node b in the graph G and $N\prime \subset N$

A view $G\prime$ of a workflow G with the property (c) is called an order preserving view of G. An order preserving view does not change the order of the nodes of the underlying workflow.

IV. View Construction Operators

Here two ways for construction of views are considered:

1) Abstraction
2) Aggregation

A. Abstraction Operator

By application of the abstraction operator (compare τ-operator in process algebra [1]) it is possible to make parts of a process invisible to external observers. The abstraction operator is like a renaming operator that renames the label of a node and makes it invisible from outside. The abstracted nodes then become internal nodes and invisible to an external observer. The abstraction operator provides a mean for construction of views. By application of this operator parts of a process that do not contribute to the interaction with another partner, are not interesting for external partners or are intended to be hidden because of privacy issues can be made unobservable and the rest of the process can be exposed as a view on the process.

Definition 3: **(Abstraction operator)**

Let $G = (N, E)$ be a workflow. Application of the abstraction operator on a node j of a workflow G results in a workflow $G\prime = (N\prime, E\prime)$, denoted as $ABS(G, j) = G\prime$

case 1: j is an activity

Let j be an activity,
$ABS(G, j) = G\prime = (N\prime, E\prime)$ with
$N\prime = N - \{j\}$,
$E\prime = \{(n_s, n_t)|[(n_s, n_t) \in E \bigwedge n_s \neq j \bigwedge n_t \neq j]\}$
$\bigcup\{(n_s, n_t)|[(n_s, j) \in E \bigwedge (j, n_t) \in E]\}$

case 2: j **is a control node**

Let j be a control node and $j\prime$ its counterpart control node, $|\{(n_s, n_t) \mid (n_s, n_t) \in E \bigwedge (n_s = j) \bigwedge (n_t \neq j\prime)\}| \leq 1$
$\Rightarrow ABS(G, j) = G\prime = (N\prime, E\prime)$ with
$N\prime = N - \{j, j\prime\}$,
$E\prime = \{(n_s, n_t) | [(n_s, n_t) \in E \bigwedge n_s \neq j \bigwedge n_t \neq j \bigwedge n_s \neq j\prime \bigwedge n_t \neq j\prime]\}$
$\bigcup\{(n_s, n_t) | n_t \neq j\prime \bigwedge [(n_s, j) \in E \bigwedge (j, n_t) \in E]\}$
$\bigcup\{(n_s, n_t) | n_s \neq j \bigwedge [(n_s, j\prime) \in E \bigwedge (j\prime, n_t) \in E]\}$

Abstraction of an activity j removes the activity j from the set of nodes N as well as all edges whose source node or target node is the activity j from the set of edges E. An edge from the predecessor of the activity j to its successor is added to the set of edges E.

The precondition for abstraction of a control node is that at most one of its outgoing paths (or incoming paths of its counterpart) contains activities. All other paths between a control node and its counterpart must directly connect them. This is reflected by checking the cardinality of edges that does not directly connect j and $j\prime$. Activities of other paths must be previously abstracted as explained in case 1 of the definition 3. Abstraction of a control node results in simultaneous removal of the control node and its counterpart as well as all edges that directly connect them. Again here edges are added from the predecessor of the control node to its successor activity and from the predecessor of counterpart control node to the successor of counterpart control node. The properties for abstraction of control nodes ensure that after abstraction the resulting flow is again a valid full-blocked workflow. Figures 1 and 2 illustrate the application of abstraction operator on activities and control nodes respectively.

Definition 4: **(Silent activity)**

An abstracted activity is called a silent activity and is unobservable from outside by an external observer.

Let $G = (N, E)$ be the graph depicted on the left part of the figure 1.

The set of nodes of G in figure 1 is N=$\{start, a, b, c, d, e, AS, AJ, end\}$ and the set of its edges is
E={(start,a), (a,b), (b,AS), (AS,c), (AS,d), (c,AJ), (d,AJ), (AJ,e), (e,end)}.

After application of the abstraction operator on activity $b \in G$, the graph transforms to a new graph $G\prime$ with $N\prime$={start, a, c, d, e, AS, AJ, end} and
$E\prime$={(start,a), (a,AS), (AS,c), (AS,d), (c,AJ), (d,AJ), (AJ,e), (e,end)}. The new graph $G\prime$ is depicted on the right part of the figure 1.

The set of nodes of G on the left part of figure 2 is N={start, a, b, d, e, AS, AJ, end} and the set of its edges is
E={(start,a), (a,b), (b,AS), (AS,d), (AS,AJ), (d,AJ), (AJ,e), (e,end)}.

After application of the abstraction operator on control nodes, the graph transforms to a new graph $G\prime$ with

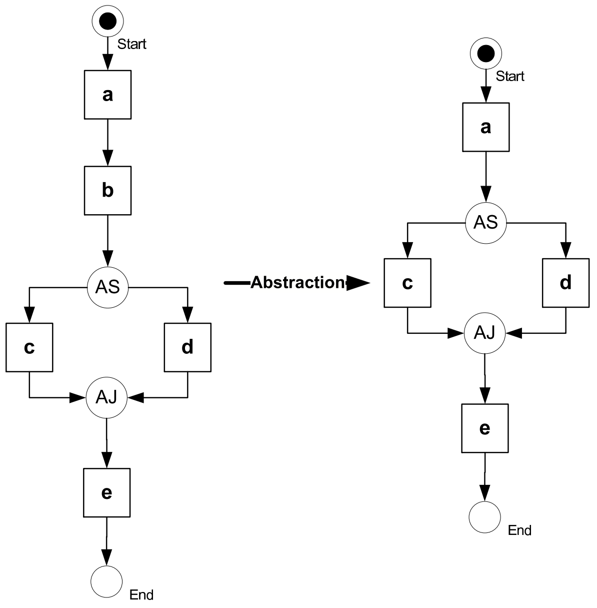

Fig. 1. Application of abstraction on the activity b

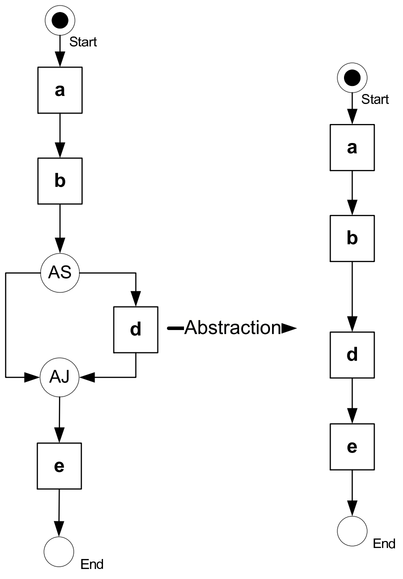

Fig. 2. Application on control nodes

$N\prime$=$\{start, a, b, d, e, end\}$ and
$E\prime$={(start,a), (a,b), (b,d), (d,e), (e,end)}. The new graph $G\prime$ is depicted on the right part of the figure 2.

Proposition 1: **(Properties of the abstraction operator)**

(a) Abstraction operator is commutative, i.e.
Let $G = (N, E), \forall$ nodes $a, b \in N$:
$ABS(ABS(G, a), b) = ABS(ABS(G, b), a)$

(b) Abstraction operator is associative, i.e.
Let $G = (N, E), \forall$ nodes $a, b, c \in N$:
$ABS(ABS(ABS(G, a), b), c) =$

$ABS(ABS(ABS(G,c),b),a)$

(c) Abstraction is an order preserving operation and does not change the order of remaining nodes, i.e. Let $G=(N,E)$ and $G\prime=(N\prime,E\prime)$ be the resulting graph after application of the abstractor operator on a node of G,
$\forall$ nodes $a,b\in N\prime:[a>b]_{G\prime}\Leftrightarrow[a>b]_G$, where $[a>b]_G$ denotes that there is a path from a to b in G
The resulting graph $G\prime$ is a valid full-blocked workflow definition.

The properties (a) and (b) follow directly from the definition of the abstraction operator (definition 3). The abstraction operator removes a node from the set of nodes and consequently the ordering of the nodes in the resulting graph remains the same as in the underlying workflow. In other words, abstraction is an order-preserving operator and the resulting workflow view is an order-preserving view. If abstraction is applied on an activity and because it is assumed that the original flow is full-blocked, the resulting flow is again full-blocked. Abstraction of control nodes results also in a full-blocked workflow. Because the abstraction of a control node must be followed by abstraction of its counterpart control node and removal of empty edges and it is assumed that the original flow is full-blocked, the resulting graph after abstraction is again full-blocked.

B. Aggregation Operator

Another way of constructing workflow views is application of aggregation. By aggregation some activities are grouped into a so-called abstract or aggregated activity. Aggregation can be used for hiding the internal structure of a group of activities. The aggregated activity is consequently contained in the workflow view and is in charge of sending and receiving data to and from activities in the underlying workflow.

Definition 5: **(Aggregated activity)**

Let $G=(N,E)$ be a workflow. An aggregated activity A_{G_A} represents a graph $G_A=(N_A,E_A)$ with the following properties:

- G_A is a connected subgraph of G
- G_A has a unique first node and a unique last node, i.e. G_A has only one incoming edge and only one outgoing edge
- If a split-node (join-node) or start of loop is in N_A, its counterpart join-node (split-node) or end of loop is also in N_A

$N_A\subseteq N$ denotes the set of activities and control nodes in the aggregated activity and E_A the dependencies between activities and control nodes, where $E_A=\{(n_s,n_t)\in E|n_s,n_t\in N_A\}$.

Note that G_A is the graph representing internal structure of the aggregated activity and A_{G_A} identifies the aggregated activity.

Definition 6: **(Aggregation)**

Aggregation is an operator that groups a subset of nodes of a workflow into an aggregated activity. Let $G=(N,E)$ be a workflow and $G\prime=(N\prime,E\prime)$ the resulting graph after application of aggregation on some activities of G. Application of the aggregation operator on a workflow G results in a new workflow $G\prime$, denoted by $AG(G,N_A)=G\prime$.

$AG(G,N_A)=G\prime(N\prime,E\prime)$ with
$N\prime=N-N_A\bigcup\{A_{G_A}\}$,
$E\prime=\{(n_s,n_t)\in E|n_s\notin N_A\bigwedge n_t\notin N_A\}\bigcup\{(n_s,A_{G_A}),(A_{G_A},n_t)|\exists(n_s,n_f),(n_l,n_t)\in E,n_f,n_l\in N_A\}$, where n_f denotes the first node of G_A and n_l its last node respectively.

All nodes $n\in N_A$ are removed from N and the aggregated activity A_{G_A} is added. All edges of the nodes contained in G_A are as well removed from the set of edges E and two edges are added. One edge from the predecessor of the first node of the aggregated activity to the aggregated activity and another from aggregated activity to the successor of the last node of the aggregated activity.

The above properties for N_A are required to ensure that the workflow after application of aggregation is still a valid full-blocked workflow definition. The following figures demonstrate that by lack of any of these properties in definition 5 the resulting graph is not a valid full-blocked workflow definition anymore.

Figure 3 clarifies why it is required that an aggregated activity be a connected subgraph of the underlying workflow. The aggregated activity in figure 3 is not a connected subgraph and it can be seen that after aggregation the resulting workflow is not a valid workflow definition anymore.

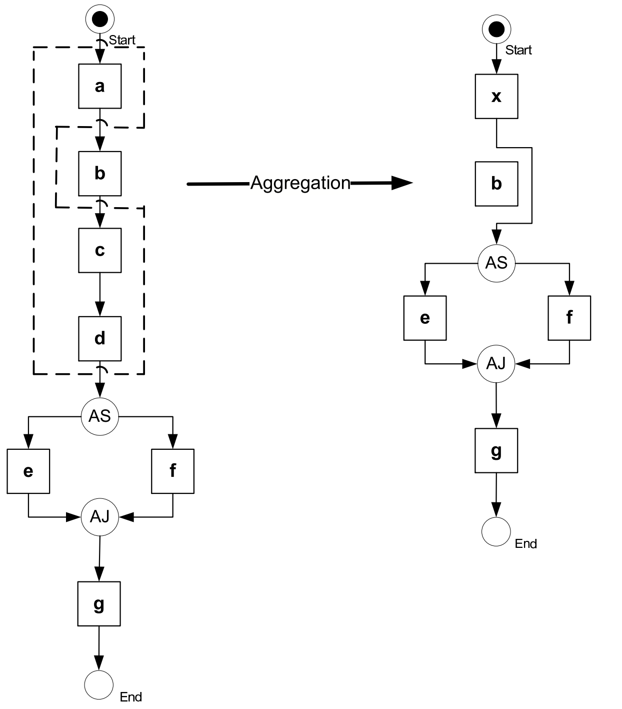

Fig. 3. Aggregated activity is not a connected subgraph

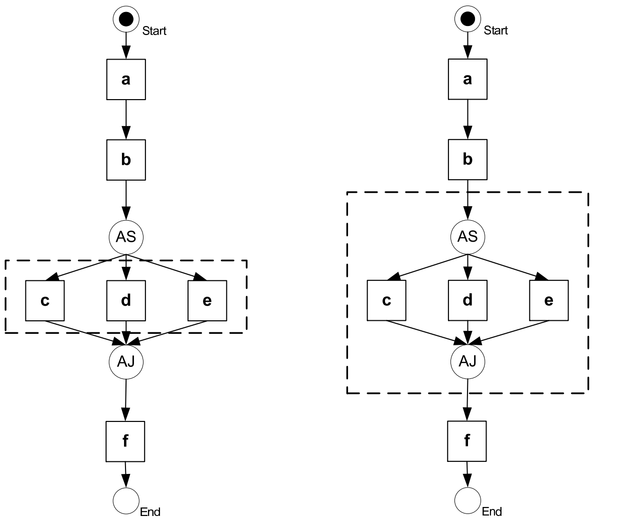

Fig. 4. Left: A wrongly aggregated activity, Right: A correctly aggregated activity

The left part of the figure 4 illustrates a wrongly aggregated activity. As it can be seen the aggregated activity has three incoming edges and three outgoing edges. In such a case it is not clear which activity is the first and the last activity of the aggregated activity. The right part of the figure shows a correctly aggregated activity with only one incoming and outgoing edge.

Figure 5 illustrates the need to include the counterpart of control nodes in aggregation. If it is not the case, the resulting graph is not a valid workflow definition as it can be seen in figure 5. Note that in this figure the exit condition of the loop is not depicted.

Figure 6 demonstrates an application of aggregation. The workflow G on the left part of the figure 6 has $N = \{$a, b, c, d, e, f, g, AS, AJ$\}$ and $E = \{$(start,a), (a,b), (b,c), (c,d), (d,AS), (AS,e), (AS,f), (e,AJ), (f,AJ), (AJ,g), (g,end)$\}$. The resulting graph $G\prime$ has $N\prime = \{$a, x, e, f, g, AS, AJ$\}$ and $E\prime = \{$(start,a), (a,x), (x,AS), (AS,e), (AS,f), (e,AJ), (f,AJ), (AJ,g), (g,end)$\}$.

The activities contained in an aggregated activity are not necessarily executable activities. Aggregated activities can again be aggregated into a new aggregated activity. In other words aggregation can be applied in a recursive fashion. Aggregation is transitive in the sense that if an activity j is contained in an aggregated activity x and the aggregated activity x is itself contained in another aggregated activity y, then the activity j is contained in the aggregated activity y, $j \in x \bigwedge x \in y \Rightarrow j \in y$.

Proposition 2: **(Properties of aggregation)**

(a) Let $G = (N, E)$ be a workflow and $G\prime = (N\prime, E\prime)$ the resulting workflow after application of aggregation on some activities of G and N_A the set of nodes of the aggregated activity A_{G_A}:
$\forall a, b \in N$:
$[a > b]_G \bigwedge a, b \notin N_A \Rightarrow [a > b]_{G\prime}$
$[a > b]_G \bigwedge a \notin N_A \bigwedge b \in N_A \Rightarrow [a > A_{G_A}]_{G\prime}$
$[a > b]_G \bigwedge a \in N_A \bigwedge b \notin N_A \Rightarrow [A_{G_A} > b]_{G\prime}$

(b) $\forall a, b \in N\prime$:
$[a > b]_{G\prime} \bigwedge a, b \neq G_A \Rightarrow [a > b]_G$
$[a > A_{G_A}]_{G\prime} \Rightarrow \forall i \in N_A : [a > i]_G$

(c) $G\prime$ is a valid full-blocked workflow definition

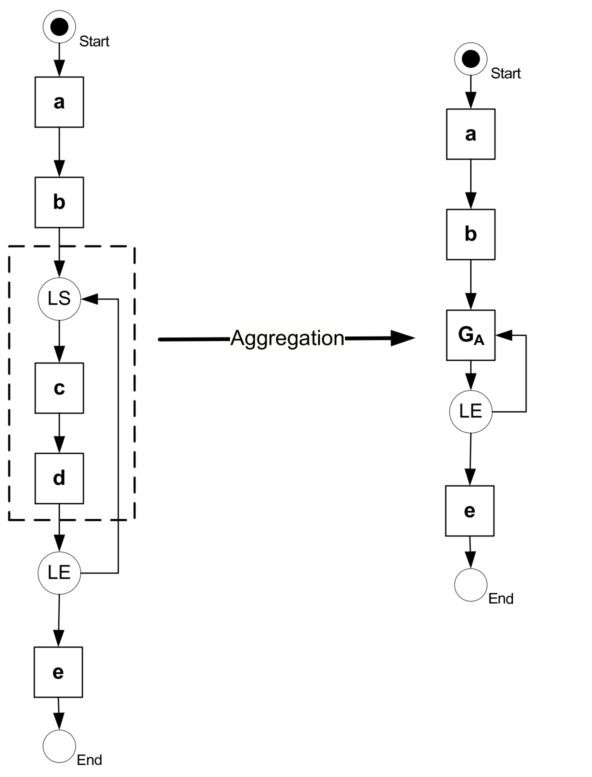

Fig. 5. The counterpart of control nodes must be included in aggregation

The above properties of the aggregation operator follow directly from the definition of the aggregation operator (definition 6).

It is clear that if two nodes are not member of an aggregated activity, the path between these two nodes remains the same and hence the ordering between these nodes remains also the same. ($[a > b]_G \bigwedge a, b \notin N_A \Rightarrow [a > b]_{G\prime}$) is true because aggregation only affects members of an aggregated activity and all other nodes remains unaffected. If a node a precedes a node of an aggregated activity, aggregation removes all edges of the aggregated activity and adds one incoming edge to the aggregated activity, i.e. the node a precedes the aggregated activity and ($[a > b]_G \bigwedge a \notin N_A \bigwedge b \in N_A \Rightarrow [a > A_{G_A}]_{G\prime}$) is true.

If a node b follows a node of an aggregated activity, aggregation removes all edges of the aggregated activity and adds one outgoing edge from the aggregated activity, i.e. the node b follows the aggregated activity and ($[a > b]_G \bigwedge a \in N_A \bigwedge b \notin N_A \Rightarrow [A_{G_A} > b]_{G\prime}$) is true. With the same argumentation the reverse direction can also be proved. The above properties show that aggregation is an order-preserving

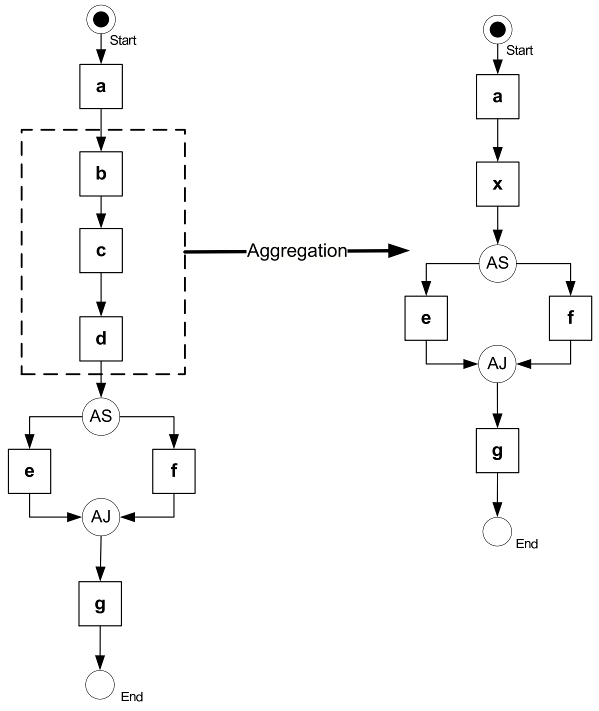

Fig. 6. Application of aggregation

operator.

Also directly from definition of the aggregation operator follows that the resulting graph after aggregation is a valid full-blocked workflow. Definition 6 requires that the counterpart of each join-node or split-node must also be contained in an aggregated activity. Because the underlying workflow is a full-blocked workflow, the resulting graph remains also a valid full-blocked workflow definition.

Definition 7: **(Event Correspondence)**

An aggregated activity begins when its first activity begins and terminates when its last activity has terminated. The start of an aggregated activity corresponds to the start of its first activity and its end corresponds to the end of its last activity

V. Construction of Views

Application of the operators presented in the previous section on a workflow guarantees a correct construction of view. The presented operators in section IV provide powerful tools for many purposes. Through abstraction a subset of the activities of the workflow or the internal activities within loops, parallel or conditional structures can be exposed in a view and aggregation provides tools for hiding the internal structures of a sub-workflow.

Proposition 3: **(Property of views)**

Let $N = (G, E)$ be a workflow. Construction of views by application of abstraction operator or aggregation as defined in definitions 3 and 6 on N results in a graph $N\prime = (G\prime, E\prime)$ which is again a workflow.

The proof follows directly from the definition and properties of abstraction operator and aggregation.

Theorem 1: **(Aggregation and abstraction construct correct views)**

Let $N = (G, E)$ be a workflow. Application of abstraction operator or aggregation as defined in definitions 3 and 6 on N results in a graph $N\prime = (G\prime, E\prime)$ which is a correct view on N.

In the properties of the abstraction operator (proposition 1) and properties of the aggregation (proposition 2) it has been shown that the abstraction operator and aggregation are order-preserving operators and the resulting graph after application of the abstraction operator and aggregation is a valid full-blocked workflow definition. From the definition of the abstraction operator (definition 3) follows that the set of nodes in the resulting graph is a subset of the nodes of the underlying workflow and from the definition of aggregation (definition 6) follows that only one node representing a subset of nodes of the underlying workflow is added to the set of nodes of the resulting graph and the other nodes are a subset of the nodes of the underlying workflow. Hence, workflow views constructed by application of abstraction operator and aggregation satisfy all the correctness criteria in definition 2 and such views are correct workflow views.

After application of abstraction or aggregation operators on a workflow, the resulting workflow is again a valid full-blocked workflow definition on which abstraction or aggregation operators can again be applied. Operators can be applied consecutively on a workflow to construct a view.

Definition 8: **(View Constructor)**

Let $G = (N, E)$ be a workflow and $G\prime = (N\prime, E\prime)$ a view on G. The view constructor operator, $VC(G) = G\prime$, is a sequence of abstraction or aggregation operators, denoted by $G \overset{\alpha}{\rightarrow} .. \overset{\alpha}{\rightarrow} .. \overset{\alpha}{\rightarrow} G\prime$, where α is either the abstraction operator or the aggregation operator.

After each application of α the workflow transforms to a new workflow. Note that the sequence is finite and can be empty. Obviously there is no unique way of constructing views rather the same view can be constructed by different sequence of operators. The view constructor is transitive in the sense that if $G\prime$ is a view on G and $G\prime\prime$ a view on $G\prime$ then $G\prime\prime$ is a view on G. However, the commutativity is not valid because different sequence of operators, produce different views.

VI. Conclusions

Workflow views have many applications and are handy tools in the hand of process managers and designers and provide organizations with obvious advantages such as improvement of privacy and protection of business details. On the one hand they allow for interaction with other partners and on the other hand they protect the business know-how and internal parts of the private process. We have introduced two operators and techniques for construction of workflow views by these operators. As long as these operators are applied correctly, a correct construction of views can be guaranteed and process designers do not need to care for correctness verification.

Acknowledgements:

This work is supported by the European Project WS-Diamond in FP6.STREP and the Austrian projects GATiB and Secure 2.0.

References

[1] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Comput. Sci.*, 37:77–121, 1985.

[2] R. Bobrik and T. Bauer. Towards configurable process visualizations with proviado. In *Proc. of the WETICE IEEE-Workshop on Agile Cooperative Process-Aware Informations Systems*, 2007.

[3] R. Bobrik, M. Reichert, and T. Bauer. View-based process visualization. In *Proc. of the 5th International Conference on Business Process Management*, 2007.

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

[5] D.K.W. Chiu, S.C. Cheung, K. Karlapalem, Q. Li, and S. Till. Workflow view driven cross-organizational interoperability in a web-service environment. In *Proc. of the Web Services, E-Business, and the Semantic Web, CAiSE 2002 International Workshop*, 2002.

[6] J. Eder and A. Tahamtan. Temporal consistency of view based interorganizational workflows. In *Proc. of the 2nd International United Information Systems Conference*, 2008.

[7] R. Eshuis and P. Grefen. Constructing customized process views. *Data & Knowledge Engineering*, 64(2):419–438, 2008.

[8] H. Frank and J. Eder. Towards an automatic integration of statecharts. In *ER'99: 18th International Conference on Conceptual Modeling*. Springer, 1999.

[9] B. Kiepuszewski, A.H.M. ter Hofstede, and C. Bussler. On structured workflow modelling. In *Proc. of CAiSE*, 2000.

[10] Q. Li, Z. Shan, P.C.K. Hung, D.K.W. Chiu, and S.C. Cheung. Flows and views for scalable scientific integration. In *Proc. of InfoScale 06, ACM International Conference Proceeding Series, Vol. 152*, 2006.

[11] D.R. Liu and M. Shen. Workflow modeling for virtual processes: An order-preserving process-view approach. *Information Systems*, 28(6):505–532, 2003.

[12] D.R. Liu and M. Shen. Business-to-business workflow interoperation based on process-views. *Decision Support Systems*, 38(3):399–419, 2004.

[13] D.R. Liu and M. Shen. Discovering role-relevant process-views for disseminating process knowledge. *Expert Systems with Applications*, 26(3):301–310, 2004.

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

[15] Z. Shan, Z. Long, Y. Luo, and Z. Peng. Object-oriented realization of workflow views for web services an object deputy model based approach. In *Proc. of the 5th International Conference on Advances in Web-Age Information Management*, 2004.

[16] A. Tahamtan. *Modeling and Verification of Web Service Composition Based Interorganizational Workflows*. PhD thesis, University of Vienna, 2009.

[17] A. Tahamtan. *Web Service Composition Based Interorganizational-Workflows: Modeling and Verification*. Suedwestdeutscher Verlag fuer Hochschulschriften, 2009.

[18] D. Tsichritzis and A. Klug. *The ANSI/X3/SPARC DBMS framework: Report of the study group on database management systems*. AFIPS press, 1978.

TABLE I
SUMMARY OF RELATED WORK ON WORKFLOW VIEWS

Paper	Construction of Views		Correctness	Applied for
	Abstraction	Aggregation		
[4]	No	Yes	No	Cooperation
[5]	No	No	No	Interoperability
[10]	No	No	No	Flexibility
[11]	No	Yes	Yes	N.A.
[12]	No	No	No	Monitoring
[13]	No	Yes	Yes	N.A.
[14]	No	No	No	Cooperation
[15]	No	No	No	Access restriction
[7]	Yes	Yes	No	Cooperation
[3]	Yes	Yes	No	Visualization
Our approach	Yes	Yes	Yes	Cooperation & Privacy