Role-based Process View Derivation and Composition

XIAOHUI ZHAO, University of Canberra
CHENGFEI LIU, Swinburne University of Technology
SIRA YONGCHAREON, Unitec Institute of Technology
MAREK KOWALKIEWICZ, SAP
WASIM SADIQ, Infosys

Process view concept deploys a partial and temporal representation to adjust the visible view of a business process according to various perception constraints of users. Process view technology is of practical use for privacy protection and authorisation control in process-oriented business management. Owing to complex organisational structure, it is challenging for large companies to accurately specify the diverse perception of different users over business processes. Aiming to tackle this issue, this paper presents a role-based process view model to incorporate role dependencies into process view derivation. Compared to existing process view approaches, ours particularly supports run-time updates to the process view perceivable to a user with specific view merging operations, and thereby enables the dynamic tracing of process perception. A series of rules and theorems are established to guarantee the structural consistency and validity of process view transformation. A hypothetical case is conducted to illustrate the feasibility of our approach, and a prototype is developed for the proof-of-concept purpose.

H.4.1 [Office Automation]: Workflow Management

Additional Key Words and Phrases: Business process view, collaborative business process, process perception

ACM Reference Format:

1. INTRODUCTION

Historically, the workflow concept has evolved from the notion of process in manufacturing and the office (Georgakopoulos, Hornick, & Sheth, 1995). With the introduction of information technology, processes in the workflow place are largely automated by workflow/business process management systems. Such systems are designed to make work more efficient, integrate heterogeneous applications systems, and support inter-organisational processes in electronic commerce applications (Stohr & Zhao, 2001). Particularly, to help organisations survive and thrive in a changing market, the flexibility in process modelling and control has been identified as a key feature for the further application of business process management systems (Kumar & Zhao, 1999).

Process views have been proposed recently for fine-granularity control of process representation (C. Liu, Li, & Zhao, 2008; Weske, van der Aalst, & Verbeek, 2004). A process view depicts a partial representation of a business process, and thereby separates process representation from the executable processes. Further, process views allow one business process to have multiple views for different users, according to their relationships, observation intentions, etc. Such flexibility finds its advantages in areas of authority control, process visualisation, collaborative business process modelling etc. (Choi, Nazareth, & Jain, 2013; Ullah & Lai, 2013)

Typically a user's perception towards a business process is subject to the user's role/position in the company, yet this perception may evolve when the user exchange or transfer the process perception with others (Caetano, Zacarias, Silva, & Tribolet, 2005). As such, a process view for a user becomes a role-based temporal and partial representation for a business process, rather than a fixed or static one. Aiming to characterise the relations and interactions among roles, perceptions and process

views, this paper proposes a role-based process view model. This model looks into process perception evolution, and facilitates process view derivation according to changing perceptions. To ensure structural consistency and validity during process view derivations, we present a set of rules and theorems to guarantee the activity execution order preservation, synchronisation dependency, non-redundancy in structural elements, etc. Particularly, this work contributes to current process view research in the following aspects:

- Analyse process perception dependency and inter-relationship according to the role hierarchy, with an emphasis on perception evolution.
- Support both process view filtering and composition operations, and combinations of them.
- Maximally preserve process structural information during process view transformations, and guarantee structural consistency and validity.
- Develop a prototype for the proof-of-concept purpose.

The remainder of this paper is organised as follows: Section 2 discusses the motivation of role-based process view management with an example. Section 3 introduces a role-based process view model. Section 4 defines a set of rules on structural consistency, and discusses how these rules regulate the process view transformation. Section 5 illustrates the feasibility of our approach with a hypothetical case. Section 6 introduces a developed prototype for the proof-of-concept purpose. Section 7 reviews the related work, and discusses both the advantages and limitations of our approach. Finally, the concluding remarks are given in Section 8 with an indication on future work.

2. MOTIVATING EXAMPLE

This section illustrates how process views evolve as users' perceptions change. View v_0 in Figure 1 shows the full picture of an Accounts Receivable (AR) process, where nodes s and e denote the starting and ending points, respectively, and the other activities are delegated by t_1, t_2, etc. Since view v_0 shows all details of the process, it is also called *base process*.

Suppose there are three users involved in this business process, viz., clerk u_1 and u_2, and AR officer u_3. Owing to the concern of fraud connection, a duty segregation policy prohibits the same person to be in charge of validating customers and calculating invoices. Thus, we assume that u_1 is assigned to check customers and customer credits, and u_2 checks customer credits and sends invoices. AR officer u_3 is exclusively authorised to issue sales orders and initiate an AR process instance. As a management role, u_3 has the right to see and handle most activities except validating customers (because of duty segregation). To adapt to such diverse process perceptions, flexible process representation is highly sought after. For example, different process views v_1, v_2 and v_3 are expected to be created for these users, respectively. In addition, all these process views should keep execution order and process structure consistent with the base process. Therefore, these process views allow users to take part in this business process, and also protect confidential information from different users.

The capability of dynamically deriving and tracing the process view perceivable to a role/individual can also help check and analyse potential breach or violation against information security/restriction. For example, suppose a new clerk u_4 is recruited to be a backup of u_1 and u_2, u_4 is assigned with the perceptions of u_1 and u_2, and u_4 sees view v_4 obtained by merging v_1 and v_2. But if clerk u_4 and officer u_3 have recently married to be a couple, the management may need to analyse whether their

collective perception violates the company's information segregation policy. A combined process view v_5 can then be derived by merging views v_3 and v_4 to reflect their collective knowledge of the process.

The dashed arrow in view v_5 denotes a synchronisation dependency between t_1 and t_3, i.e., t_3 must start after t_1's completion. As the result of merging v_3 and v_4, v_5 keeps all the information derivable from them. In v_5, tasks t_1 and t_2 are placed in two branches in parallel, because the execution order between them is not specified in either v_3 or v_4. This phenomenon reflects the process view dynamics during the perception transitivity.

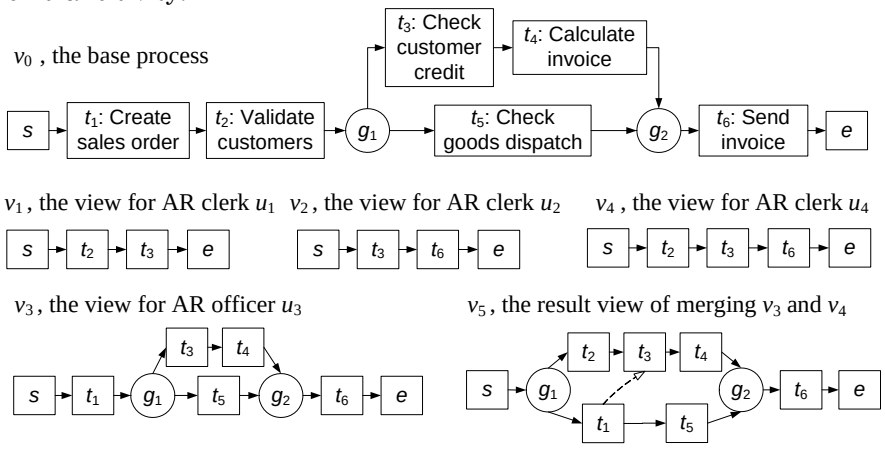


Fig. 1. Process view examples.

The above scenario illustrates that users have different perceptions over the same business process, and different perceptions result in different process views. A user's process view evolves when perception exchange or escalation occurs. Current works on process views mainly focus on process filtering and task aggregation, but few efforts have been put on view merging or the influences from user interactions.

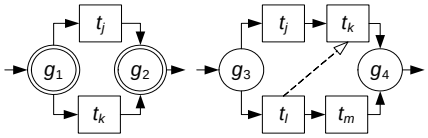
To address these issues, this paper proposes a role-based process view model, together with a set of rules and theorems to ensure the structural consistency and validity during process view derivation and composition. The reported work is based on a preliminary version of our work on process view derivation and composition (X. Zhao, Liu, Sadiq, Kowalkiewicz, & Yongchareon, 2011; Xiaohui Zhao, Liu, Sadiq, & Kowalkiewicz, 2008), with significant improvements and extensions on theoretical analysis and prototype implementation.

3. ROLE BASED PROCESS VIEW MODEL

Our role-based process view model consists of elementary process constructs, as well as concepts of process views, perceptions and the relations between roles.

- *Process Constructs*

Definition 1. (Gateway) Gateways are used to represent the structure of a control flow. Here we define four types of gateways, namely *Xor-Split*, *Xor-Join*, *And-Split*, *And-Join*. Figure 2 shows the samples of these gateways, where g_1 and g_2 denote *Xor-Split* and *Xor-Join* gateways, respectively, g_3 and g_4 denote *And-Split* and *And-Join* gateways, respectively.



(a) (b)

Fig. 2. Gateway examples.

Though a loop structure is functionally similar to a special *Xor-Split/Join* structure, it can trigger an already executed task to be started again, and therefore make trace (behaviour) analysis a lot more complex. The same happens to structural analysis, as it makes a graph cyclic. For this reason, we are not to explicitly discuss loop structures in this paper.

Definition 2. (Synchronisation Link) In an *And-Split/Join* structure, synchronisation links specify the synchronisation dependency between tasks in different branches.

The dashed arrow connecting t_l to t_k In Figure 2 (b) is a synchronisation link, which indicates that t_k can only start after t_l completes. The notion of synchronisation link has been first proposed in $ADEPT_{flex}$ (Reichert & Dadam, 1998), yet here we mainly follow the definition from Business Process Execution Language (BPEL) (Andrews et al., 2003), which restricts synchronisation links within *And-Split/Join* structures.

- *Process View and Perceptions*

Definition 3. (Process View) The structure of a process view v can be modelled as a directed graph formalised as tuple (T, G, L, SL), where the node set comprises T and G, and the edge set comprises L and SL, respectively:

– T=$\{s, e, t_1, t_2, \ldots, t_n\}$, $t_i \in T$ ($1 \le i \le n$) represents a task of v. s and e represent the starting point and the ending point of v, respectively.
– G=$\{g_1, g_2, \ldots, g_m\}$, $g_i \in G$ ($1 \le i \le m$) represents a gateway of v.
– L is a set of links. A link l=(m_1, m_2)$\in L$ indicates the execution dependency that node m_2 starts after m_1 finishes, where m_1, $m_2 \in N$, and N=$T \cup G$.
– SL is a set of synchronisation links. A link sl=(m_1, m_2)$\in SL$ indicates the execution dependency that node m_2 starts after m_1 finishes, where m_1, $m_2 \in N$.
– For each node $m \in N$, $ind(m)$ and $outd(m)$ define the number of links which take m as the target node and source node, respectively. Note, ind and $outd$ only count the number of plain links but not synchronisation links.
– $\forall n \in N \setminus \{s, e\}$, $ind(n)$=$outd(n)$=1. This property is guaranteed by the usage of gateways.

In a business process, tasks carry all the business information instead of control constructs, such as links, synchronisation links and gateways. Therefore, we define that a user's process perception is subject to the set of tasks that the user is allowed to see.

Definition 4. A user u's perception q_v cover process view v contains the tasks that u is allowed to see, i.e., $q_v = \{t \mid t \in v.T$ and t is visible to $u\}$. Predicate $can\_see(r, v)$ is used to represent the fact that role r (delegating a group of users) can see view v.

The following two functions are defined to represent the process view filtering and merging operations.

– $filter(v, q_v)$ returns the process view generated from view v according to perception q_v.

– $merge(v_1, v_2)$ returns the process view that combines views v_1 and v_2.

The details on how to handle tasks, gateways, and links of process views during process view transformations will be discussed in Section 4.

When roles exchange process information, their perception will be transferred and merged accordingly. To represent such perception changes, the following relations are defined:

Definition 5. Perception Inheritance ($\rightarrow$). Let x and y be roles such that $x \rightarrow y$, i.e., x has an inheritance only relation over y. For a process view v, the following expressions hold:

$\forall v, (x \rightarrow y) \wedge can\_see(y, v) \Rightarrow can\_see(x, v)$ or

$\forall v, (x \rightarrow y) \wedge can\_see(x, v') \wedge can\_see(y, v) \Rightarrow can\_see(x, merge(v', v))$.

Definition 6. Perception Authorisation ($\overset{q_v}{\succ}$). Let x and y be roles, and q_v be a perception defined on view v such that $x \overset{q_v}{\succ} y$, i.e., x authorises perception q_v to y. Then the following expressions hold:

$\forall v, (x \overset{q_v}{\succ} y) \wedge can\_see(x, v) \Rightarrow can\_see(y, filter(v, q_v))$ or

$\forall v, (x \overset{q_v}{\succ} y) \wedge can\_see(x, v) \wedge can\_see(y, v_1) \Rightarrow can\_see(y, merge(v_1, filter(v, q_v)))$.

Definition 7. Inheritance-authorisation (*IA*) ($\overset{q_v}{\triangleright}$). Let x and y be roles, and q_v be a perception defined on view v such that $x \overset{q_v}{\triangleright} y$, i.e., $x \overset{q_v}{\succ} y$ and x has an inheritance relation over y. Then the following expression holds.

$(x \overset{q_v}{\triangleright} y) \Rightarrow (x \overset{q_v}{\succ} y) \wedge (x \rightarrow y)$

Based on above definitions and properties, some inference rules can be derived for relation transitivity. Let x, y and z be roles, v_1 and v_2 be two process views, q_{v1} and q_{v2} be the perceptions defined on v_1 and v_2, respectively, and $q_{v2} \subseteq q_{v1}$. The following rules can be derived.

(1) $(x \rightarrow y) \wedge (y \rightarrow z) \Rightarrow x \rightarrow z$;

(2) $(x \overset{q_{v1}}{\succ} y) \wedge (y \overset{q_{v2}}{\succ} z) \Rightarrow x \overset{q_{v2}}{\succ} z$;

(3) $(x \overset{q_{v1}}{\triangleright} y) \wedge (y \overset{q_{v2}}{\triangleright} z) \Rightarrow x \overset{q_{v2}}{\triangleright} z$;

(4) $(x \rightarrow y) \wedge (z \overset{q_{v1}}{\succ} y) \Rightarrow z \overset{q_{v1}}{\succ} x$;

(5) $(x \rightarrow y) \wedge (z \overset{q_{v1}}{\triangleright} y) \Rightarrow z \overset{q_{v1}}{\succ} x$;

Rules (1), (2) and (3) represent the basic transitivity in the monolithic relation context, rules (4) and (5) represent the deduction of hybrid relations.

Figure 3 illustrates relationship among aforementioned concepts with a meta model, where numerical parameters are used to show corresponding cardinality. A process is constructed as a combination of links, synchronisation links, gateways and tasks. A role owns a perception over a process, and perceptions can be inherited and authorised between roles. A perception is defined as a set of visible tasks to the role, and according to each perception a process view can be created, which is a partial view of the base process.

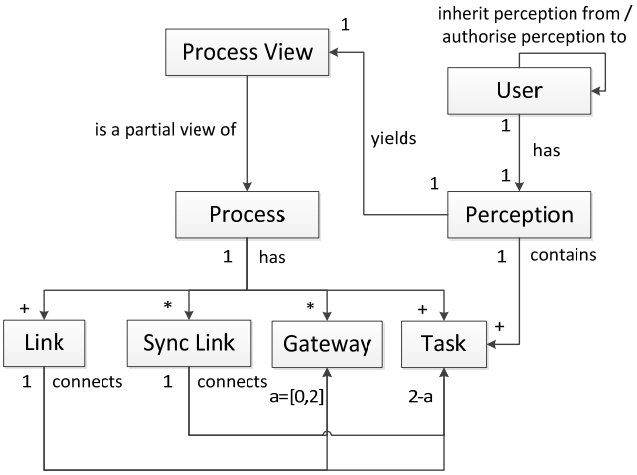


Fig. 3. the meta model of the role-based process view model.

4. PROCESS VIEW TRANSFORMATION

During process view transformation, the structural information of the base process should be kept at the maximal extent. To guarantee the structure preservation, consistency and validity, we defined a set of rules as follows.

4.1 Consistency and Validity Rules

- *Preliminary*

For a process view v, we define the following notions and functions to formally represent its structural characteristics:

– A *dummy branch* denotes a branch in a *Split/Join* structure such that the branch contains nothing but only one link.
– A common split gateway predecessor (CSP), x, of a set of tasks, T, denotes a split gateway such that x is the predecessor of each task in T. Function $CSP(t_1, t_2)$ returns the set of common split gateway predecessors of t_1 and t_2, or returns null if the two tasks have no common split gateway predecessors.
– A *path* denotes a sequence of nodes such that from each of its nodes there is a link to the next node in the sequence. Here, the node set for v is $N=T\cup G$.
– A task t is said to be involved in a Split/Join structure scoped by a pair of gateways g_1 and g_2, if $\exists$path $p=(g_1, l_1, \ldots, t, \ldots, l_m, g_2)$.
– $before(t_1, t_2)$ denotes that task t_1 will be executed earlier than task t_2. This means that there exists a path starting from t_1 to t_2 in the corresponding directed graph. Apparently, *before* is a transitive binary relation.
– $branch(g, t_1, t_2)$ is a boolean function, which returns true if t_1 and t_2 lie in the same branch led from split gateway g, and returns false otherwise.
– $preN(n)$ and $postN(n)$ return the immediate preceding and succeeding task (or gateway) of n, respectively, where n is a task or gateway.

- *Structural Consistency Rules*

Given two process views v_1 and v_2 derived from view v, v_1 and v_2 are required to comply with the following rules:

Rule 1. (Order preservation) For the tasks belonging to v_1 and v_2, the execution sequences of these tasks should be consistent, i.e.,

If $\exists t_1, t_2 \in v_1.N \cap v_2.N$ such that *before*(t_1, t_2) exists in v_1, then *before*(t_1, t_2) also exists in v_2.

Rule 2. (Branch preservation) For the tasks belonging to v_1 and v_2, the branch subjection relationship of these tasks should be consistent, i.e.,
$\forall t_1, t_2 \in v_1.N \cap v_2.N$ (where $v_1.N = v_1.T \cup v_1.G$ and $v_2.N = v_2.T \cup v_2.G$, as defined in Definition 3, and $g \in CSP(t_1, t_2)$ in v_1, $g \in CSP(t_1, t_2)$ in v_2, if *branch*(g, t_1, t_2) in v_1, then *branch*(g, t_1, t_2) in v_2, or if ¬*branch*(g, t_1, t_2) in v_1, then ¬*branch*(g, t_1, t_2) in v_2.

Rule 3. (Synchronisation dependency preservation) When task t is deleted during a filtering operation, and t is involved with a synchronisation link l, e.g., synchronisation link (t_2, t_4) as shown in Figure 4 (a), then l should be
– adjusted to lead from *preN*(t) if t is the source task, as synchronisation link (t_2, t_4) changes to (t_1, t_4) in Figure 4 (b);
– adjusted to lead to *postN*(t) if t is the target task of l, as synchronisation link (t_2, t_4) changes to (t_2, t_5) in Figure 4 (c).

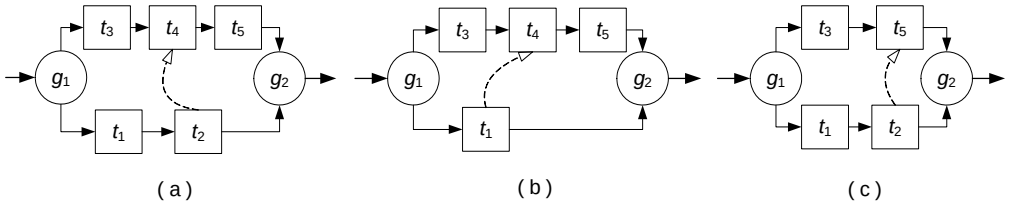


Fig. 4. Gateway examples.

- *Structural Validity Rules*

Given a process view v, the following rules are defined to verify structural correctness:

Rule 4. (No empty *Split/Join* structures) If a Split/Join structure contains only dummy branches, the *Split/Join* structure should be deleted.

Rule 5. (No dummy or single branch in *And-Split/Join* structures) If a dummy branch emerges in an *And-Split/Join* structure after a filtering operation, the dummy branch should be deleted. If the *And-Split/Join* structure contains only one non-dummy branch, the structure will be downgraded into a sequential structure.

Rule 6. (Dummy branch in *Xor-Split/Join* structures) For an *Xor-Split/Join* structure, if the tasks on a branch are all deleted, the branch (with only one link now, and is called dummy branch) should remain to indicate the existence of an alternative execution path than the other branches. If multiple dummy branches exist in that structure, these dummy branches should be combined into one.

Rule 7. (No redundant links between tasks) When merging multiple views into one view, the execution orders that are derivable from others should be removed, i.e.,
If $\exists$path $p=(n_1, l_1, \ldots, l_{m-1}, n_m)$ in v and $l \in v.L$ such that $l=(n_1, n_m)$, and $\{n_1, n_m\} \subseteq v.T$, then remove l from $v.L$.

Rule 8. (Symmetry of gateways) The gateways must be used in pairs canonically. This means $\forall g_1 \in v.G$, *type*(g_1)=*And-Split* (*Xor-Split*), $\exists g_2 \in v.G$, *type*(g_2)=*And-Join* (*Xor-Join*), and g_1 and g_2 construct a closed *Split/Join* structure, i.e., all branches start from g_1 and end at g_2. This rule indicates that the approach

assumes the business processes and process views are all well-formed (block-structured).

Rule 9. (Validity of synchronisation links) A synchronisation link (n_1, n_2)$\in v.SL$ is invalid if
– n_1 and n_2 are not involved in a common *And-Split/Join* structure, or
– n_1 is a split gateway or n_2 is a join gateway, or
– n_1 is a task involved in an *Xor-Split/Join* structure, yet n_2 is not involved in the same *Xor-Split/Join* structure.

Invalid synchronisation links should be removed.

- *Rule on Information Loss*

Rule 10. (Information loss) Suppose base process p contains link l=(t_1, t_2) or synchronisation link sl=(t_1, t_2), process view v is obtained by merging two other views that are based on perceptions q_1 and q_2 of roles r_1 and r_2, respectively. If $\exists$ t_1 and t_2 such that $t_1 \in q_1$, $t_2 \in q_2$, $t_1 \notin q_2$ and $t_2 \notin q_1$, then $l \notin v.L$ or $sl \notin v.SL$.

This rule indicates a case of information loss due to the dependency between the visibility of a process element and the perceptions of involved roles. Because neither q_1 nor q_2 contains both of tasks t_1 and t_2, the execution order between t_1 and t_2 is not known by either r_1 or r_2. Thus, the combined view v cannot derive out this execution order information, i.e., (t_1, t_2), since it is already lost in the pre-merging process views.

Most traditional process view approaches solely rely on process view filtering operations (Eshuis & Grefen, 2008; Issam, Schahram, & Samir, 2006; D.-R. Liu & Shen, 2003; van der Aalst & Weske, 2001). To handle this case, they often combine q_1 and q_2 first, and then use the combined perception to filter the base process. Yet, the result from such filtering will retain link or synchronisation link (t_1, t_2), as the combined perception would contain both t_1 and t_2, and therefore the filtering operation would not remove link (t_1, t_2). This actually reveals a limitation of reusing filtering operations to realise view merging, as filtering operations do not consider the potential information loss from the pre-merging process views.

4.2 Theorems on Process View Merging

Compared to And-Split/Join structures, Xor-Split/Join structures have special characteristics in preserving structural information. This subsection particularly investigates these characteristics with the following findings, which serve as a cornerstone for realising the process view merging operation.

Lemma 1. *When filtering view v into view v', if task t is involved in an Xor-Split/Join structure in v, and t also exists in v', then the Xor-Split/Join structure exists in v', too.*

Proof. As stated in Rule 6, an Xor-Split/Join structure will not be deleted unless it contains no tasks. Therefore, the existence of t denotes the existence of its belonged Xor-Split/Join structure. □

Lemma 2. *Given views v_1 and v_2 both derived from view v, if task t exists in two views, and t is involved in an Xor-Split/Join structure of view v_1, then t is also involved in an Xor-Split/Join structure of view v_2.*

Proof. As indicated by the proof for Lemma 1, the existence of t represents the existence of an Xor-Split/Join structure in v. As t exists in v_1 and v_2, the Xor-Split/Join structure must exist in v_1 and v_2. Therefore, t should be involved in this Xor-Split/Join structure contained in v_2 at least. □

Theorem 1. *Given views v_1 and v_2 both derived from view v, if task t exists in v_1 and v_2, then all the nested XOr-Split/Join structures in which t is involved in v_1 correspond to the ones in which t is involved in v_2.*

Proof. As indicated by the proofs for Lemma 1 and Lemma 2, all the Xor-Split/Join structures containing t are kept in v_1 and v_2. Suppose $stru_1$ and $stru_2$ are two Xor-Split/Join structures containing t in v, and $stru1$ is nested by $stru_2$. Because of Lemma 1, $stru_1$ and $stru_2$ also exist in v_1 and v_2. Due to the branch preservation rule, $stru_1$ is guaranteed to exist in a branch of $stru_2$ in v_1 and v_2. Thus, the nested relation between $stru_1$ and $stru_2$ is preserved in v_1 and v_2. Similarly, we can prove that the nested relation of all the other involved Xor-Split/Join structures is preserved in v_1 and v_2. Therefore, the Xor-Split/Join structures containing t in views v_1 and v_2 correspond to each other. □

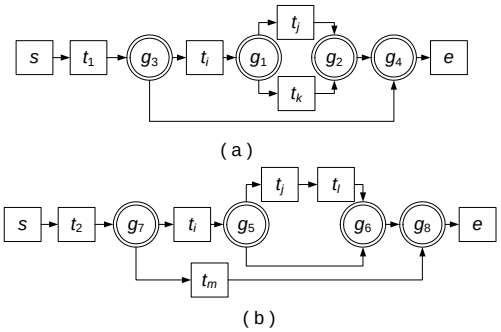


Fig. 5. Matching Xor-Split/Join structures

For example, Figure 5 shows two views derived from the same process, and both views have common task t_j. According to Theorem 1, we can consider that the closest Xor-Split/Join structured, i.e., the structure scoped by g_1 and g_2 in Figure 5 (a) and the structure scoped by g_5 and g_6 in Figure 5 (b), correspond to each other. Consequently, we can infer that task t_k should belong to the dummy branch shown in Figure 5 (b). Further, for the second closest Xor-Split/Join structures, the structure scoped by g_3 and g_4 in Figure 5 (a) corresponds to the structure scoped by g_7 and g_8 in Figure 5 (b). This means that t_m in Figure 5 (b) belongs to the dummy branch shown in Figure 5 (a).

When two process views are merged together, different tasks/gateways with the same preceding/succeeding task/gateway need to be restructured into a new Split/Join structure with newly added gateways. For example, when combining the two views in Figure 5, tasks t_1 and t_2 will be re-arranged into a Split/Join structure between s and g_3 (or g_7, since these two gateways correspond to each other) in the result view whereby a pair of new gateways will be added to represent this new Split/Join structure. Theorem 2 guarantees that all such new gateways are And-Split/Join gateways.

Theorem 2. *In case of merging two process views v_1 and v_2, if n_1, $n_2 \in v_1.N \cap v_2.N$, $n_3 \in v_1.N \setminus v_2.N$ and n_3 is on a path from n_1 to n_2 in v_1, and exist $n_4 \in v_2.N \setminus v_1.N$ and n_4 is on a path from n_1 to n_2 in v_2, then a pair of And-Split and Join gateways, g_x and g_y, will be added between n_1 and n_2 to connect n_3 and n_4 in a parallel structure in the result view.*

Proof. This Theorem can be proven from the perspective of execution order preservation. In v_1, the path containing n_3 from n_1 to n_2 denotes that n_3 will be executed after (or immediately after) n_1 and before (or immediately before) n_2. In v_2,

the path containing n_4 from n_1 to n_2 denotes that n_4 will be executed after (or immediately after) n_1 and before (or immediately before) n_2. According to Rule 1, the merged view should preserve all these execution order information. Thus the result view should reflect that both n_3 and n_4 will be executed after (or immediately after) n_1 and before (or immediately before) n_2. Therefore, we can conclude that the newly added gateways, g_x and g_y, are And-Split/Join gateways. □

4.3 Analysis on View Operations

As two basic view transformation operations, view filtering and view merging are discussed in detail in this subsection. The enabling algorithms are presented in Appendix.

- *View Filtering*

View filtering denotes the operation of filtering off a set of tasks from a given view. This operation comprises the following steps:

(1) Remove specified tasks

The tasks excluded in the perception are removed from the source process view.

(2) Adjust links and synchronisation links

The removal of tasks may break the connectivity of the view graph. Therefore, some links and synchronisation links need to be adjusted to connect the isolated nodes, while keeping the order preservation according to Rules 1-3.

(3) Check Split/Join structures

The Split/Join structures may also be broken during the task removal, and therefore they need to be adjusted according to Rules 2, 4, 5, and 6.

- *View Merging*

A view merging operation combines two process views, and organises the result view in a correct structure. This operation comprises the following steps:

(1) Match *Xor-Split/Join* structures

As stated in Rule 6, an *Xor-Split/Join* structure will not be deleted unless it contains no tasks. Therefore, if there is a common task in an *Xor-Split/Join* structure contained in two different process views, these two *Xor-Split/Join* structures should correspond to each other. Thus, the first step of the merging operation is to match the *Xor-Split/Join* structures of the input process views.

(2) Combine views and remove redundant links

During the combination, common tasks are merged together first, and all links are inherited. This action simply preserves all previous execution order information, yet it may also generate redundant execution order information. Take the merging of views in Figure 6 (a) into the one in Figure 6 (b) for example, the link from s to t_j and the one from t_j to e are redundant, as the order information is already covered by other links. According to Rule 7, such redundant links should be removed, and thereby a cleaned view can be obtained as shown in Figure 6 (c).

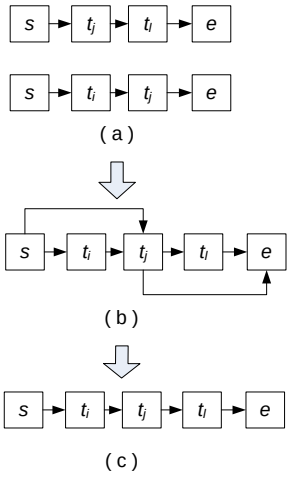


Fig. 6. Combining views and removing redundant links.

(3) Add *And-Split/Join* gateways

Common nodes exist in any pair of process views, at least the starting and ending nodes, i.e., *s* and *e*. For example, the views in Figure 7 (a) have tasks *s*, t_m and *e* in common. When combining such corresponding tasks, these common nodes make result in some Split/Join structures, as shown in Figure 7 (b). To comply with the process view structure definition, *And-Split/Join* gateways should be added properly to the result view, as shown in Figure 7 (c).

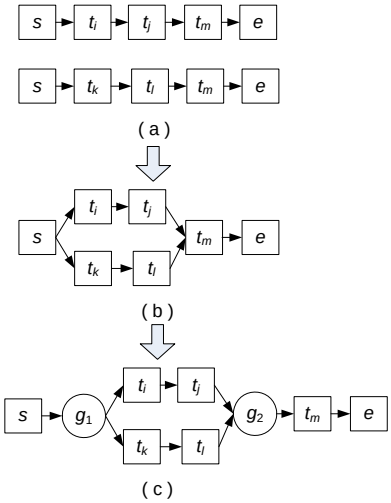


Fig. 7. Adding And-Split/Join gateways.

(4) Check *And-Split/Join* structures

In last step *And-Split/Join* gateways are added wherever a task connects to two or more nodes, but this cannot guarantee the added gateways are well in pairs. For example, the result view in Figure 8 (a1) may change to Figure 8 (a2) after adding gateways g_1 and g_2. The path from g_1 to g_2 via t_m actually reflects the synchronisation dependency between tasks t_i, t_l and t_m. Therefore, this path should be reconnected with two synchronisation links as shown in Figure 8 (a3), meanwhile g_1 and g_2 are

removed as their structure downgrades to a sequential one according to Rule 5. Task t_m is now left without any incoming or outgoing links but only synchronisation links, which violates the structural correctness. Thus, extra links are added to make link t_m be in a branch, as shown in Figure 8 (a4). The added links do not change the execution order, because synchronisation links own a higher priority.

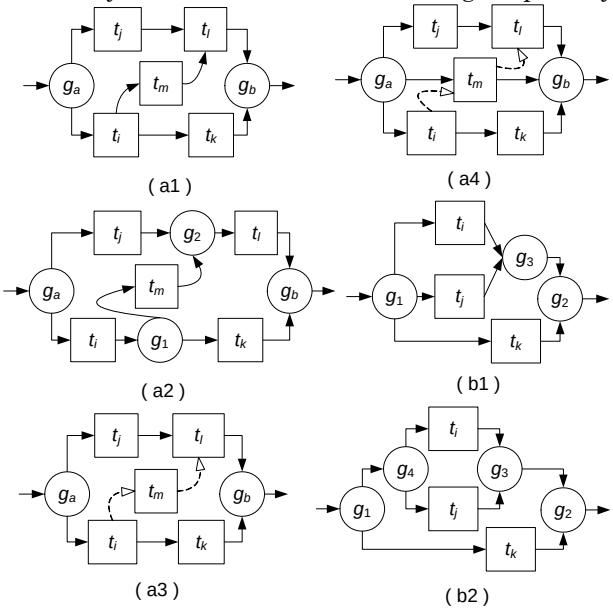


Fig. 8. Check *And-Split/Join* structures.

To guarantee the gateways are well in pairs, the obtained view may be complemented with extra *And-Split/Join* gateways, as mentioned in Rule 8. For example, the view shown in Figure 8 (b1) will add *And-Split* gateway g_4 to evolve to the view in Figure 8 (b2).

5. HYPOTHETICAL CASE

In this section, we use a hypothetical case to illustrate how our approach applies to a business scenario. Figure 9 shows a simplified sales process, which starts from receiving orders, then handles shipping (either outsource it or do it by itself) and produces in parallel, and finishes by dispatching goods. For representation simplification, we re-depict this process as v_0 in Figure 11, where t_1, t_2, …, t_8 delegate the concrete tasks.

Five roles are involved in this business process. As shown in Figure 10, initially CEO, workshop manager (WM) and sales manager (SM) inherit process perceptions from workshop manager, workshop staff (WS) and sales assistant (SA), respectively. Later, CEO and workshop staff may authorise perceptions to the workshop manager and the sales manager, respectively. Symbols "I" and "A" along arrows indicate perception inheritance and perception authorisation relations, respectively.

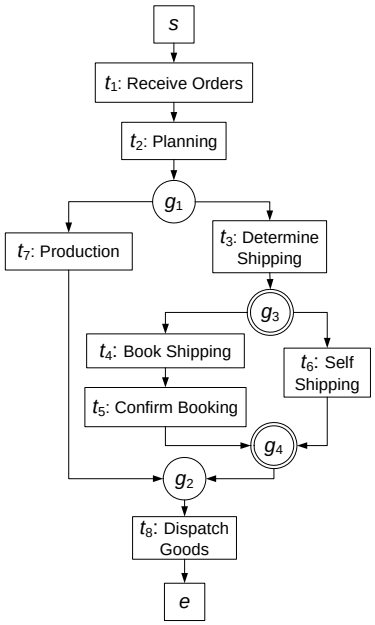


Fig. 9. The business process in the hypothetic case.

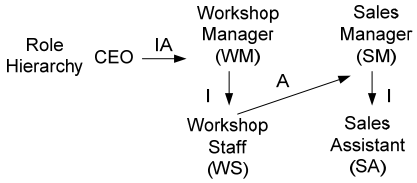


Fig. 10. The role hierarchy in the hypothetic case.

At the initial time, CEO can see the whole sales process, and thus he sees process view v_0 in Figure 11. Workshop staff holds the perception of t_2, t_7 and t_8, and the sales assistant holds the perception of t_1, t_3 and t_8. Accordingly, these two roles see process views v_1= *filter*(v_0, {t_2, t_7, t_8}) and v_2= *filter*(v_0, { t_1, t_3, t_8}), respectively. Similarly, workshop manager and sales manager see process views v_3 and v_4 in Figure 11, respectively.

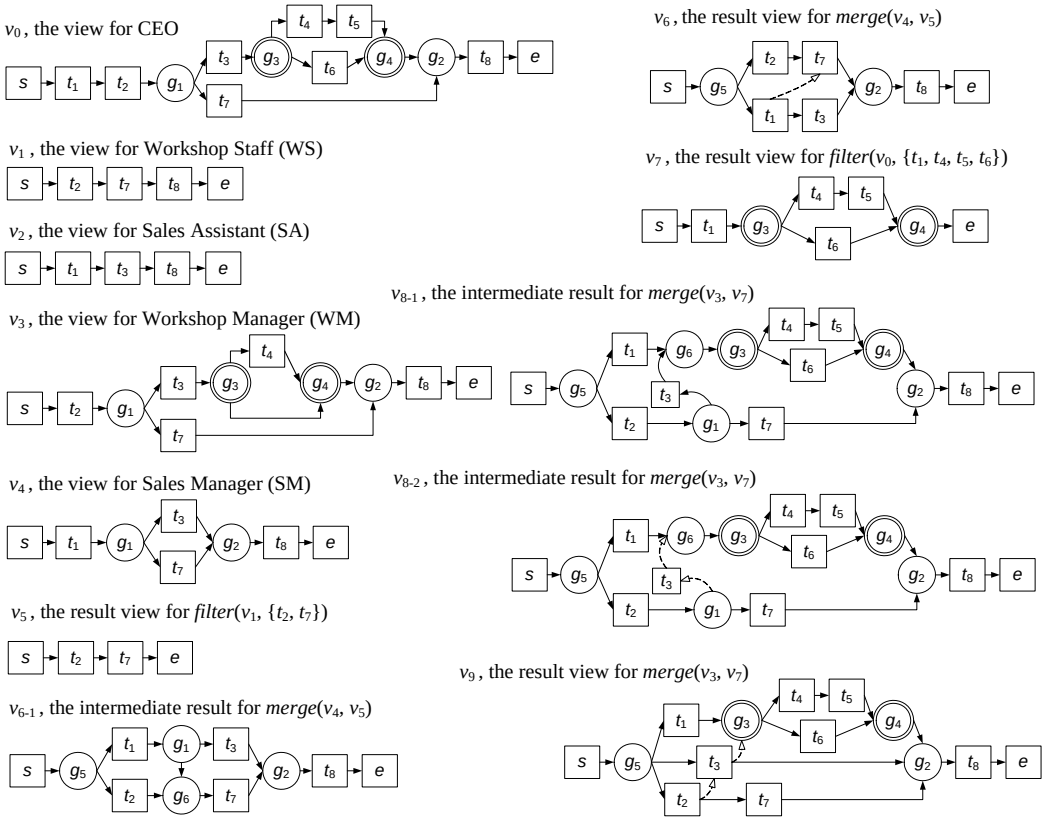


Fig. 11. Involved process views.

To notify sales manager about the production progress, workshop staff may authorise the perception of t_2 and t_7 to sales manager. With such authorisation, sale manager can perceive view v_6=*merge*(v_4, *filter*(v_1, {t_2, t_7})). Views v_5 and $v_{6\text{-}1}$ illustrate the intermediate results of this transformation. The authorisation of perception t_2 and t_7 results in view v_5=*filter*(v_1, {t_2, t_7}). View $v_{6\text{-}1}$ shows the intermediate result after combining v_4 and v_5, removing redundant links and adding new *And-Split/Join* gateways, i.e., g_5 and g_6. In $v_{6\text{-}1}$, the path from g_1 and g_6 connects two unpaired gateways, and therefore the link between g_1 and g_6 should be converted into a synchronisation link, as stated in step 3 of view merging in Section 4.3.

To let workshop manager know more about the logistics flow, CEO may authorise the perception of t_1, t_4, t_5, and t_6 to workshop manager. Thus, workshop manager can now see view v_9=*merge*(v_3, *filter*(v_0, {t_1, t_4, t_5, t_6})). As involved intermediate views, v_7=*filter*(v_0, {t_1, t_4, t_5, t_6}) represents the authorised view to workshop manager, and $v_{8\text{-}1}$ shows the result after combining v_3 and v_7, removing redundant links and adding new *And-Split/Join* gateways, i.e., g_5 and g_6. The *Xor-Split/Join* structures in v_3 and v_7 correspond to each other, because they own a common task t_4, as stated by Theorem 1. In $v_{8\text{-}1}$, the path from g_1 and g_6 connects two unpaired gateways, and therefore the link between g_1 and t_3, and the link between t_3 and g_6 should be converted into synchronisation links. Consequently, this view changes into $v_{8\text{-}2}$, where t_3 is only connected with two synchronisation links. Further, as stated in step 4 of view merging operation discussed in Section 4.3, t_3 will be adjusted into a new branch between g_5 and g_2. View v_9 shows the final result view of merging v_3 and v_7.

The execution order information between t_1 and t_3 is not derivable from either v_3 or v_7, and therefore the merged view can only place t_1 and t_3 in parallel branches.

Table 1 lists all the perception relations between the roles involved in this hypothetical case, where (*i*-*iv*) can be directly obtained from the role hierarchy and (*v*) can be derived out using the inference rules defined in Section 3.

Table 1. Perception Relations between Roles

(*i*)	$CEO \overset{q=\{t_1,t_4,t_5,t_6\}}{\rhd} WM$;
(*ii*)	$WM \rightarrow WS$;
(*iii*)	$WS \overset{q=\{t_2,t_7\}}{\succ} SM$;
(*iv*)	$SM \rightarrow SA$;
(*v*)	$CEO \overset{q=\{t_1,t_4,t_5,t_6\}}{\succ} WM$.

6. PROTOTYPE IMPLEMENTATION

To prove the concept, we have implemented a prototype called "Artifact-M for BPEL", which is available at http://sites.google.com/site/maxsirayongchareon/artifact-m/bpel-view. Artifact-M for BPEL is an extension of Artifact-M which was originally developed for artefact-centric process modelling (Yongchareon, Liu, Yu, & Zhao, 2015). The prototype fully supports process view construction operations including hiding, aggregating, filtering and merging over business processes written in Business Process Execution Language (BPEL) (Andrews et al., 2003). The software provides automatic validation of BPEL process structure to ensure the construction of BPEL views is safe and sound. View consistency checking is also supported based on the set of consistency rules mentioned in Section 4.1, to guarantee sound view derivation.

Figure 12 illustrates the architecture and the working process of Artifact-M. First, view transformation operations are first defined in Process View Definition Language (PVDL) (Yongchareon, Zhao, Liu, & Kowalkiewicz, 2008) (an XPath-like language specially designed by us). In PVDL, an XPath-like expression can contain multiple view operations over the tasks on a path of a business process or a process view. A PVDL file will be later converted into a Process View Transaction Definition (PVTD) (Yongchareon et al., 2008) file automatically. PVTD breaks down the XPath-like expressions in PVDL into primitive view operations, which can be performed by the Artifact-M Engine. The transformed view is outputted as a BPEL file, and is graphically viewed using external BPEL viewers, such as SAP Maestro and Flash BPEL Viewer.

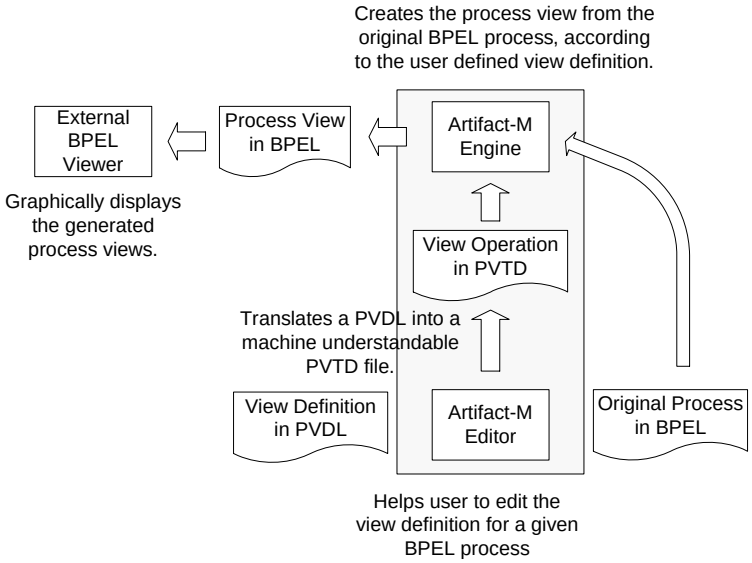


Fig. 12. The architecture of Artifact-M for BPEL.

The user interface of Artifact-M for BPEL is shown in Figure 13.

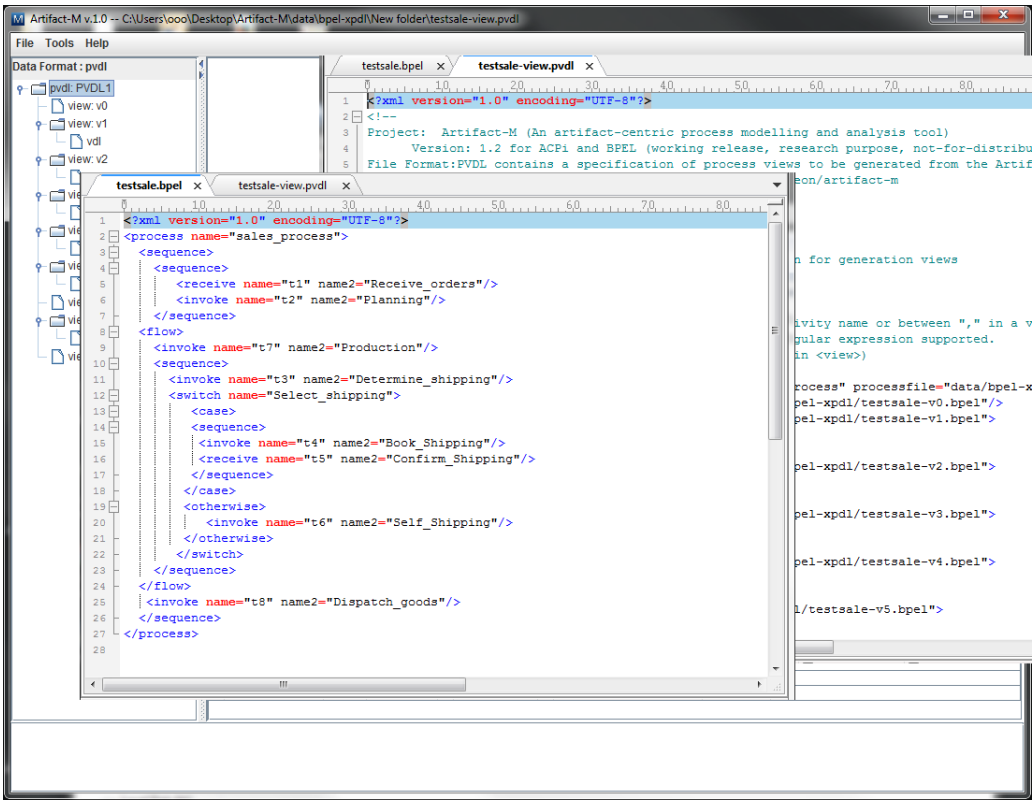


Fig. 13. The main screen of Artifact-M, the BPEL code of the running example, and the PVDL file for view definitions.

7. RELATED WORK AND DISCUSSION

The "visibility line" of business has been first discussed in 80's from the pure business perspective (Shostack, 1984). With the prevalence of process-oriented management, the incorporation of process views into business process management becomes an inevitable trend. As the de facto standard process modelling language in Web service world, BPEL can describe both executable and abstract processes, where the latter serve similarly as process views. Martens (2005) has discussed the consistency between BPEL executable processes and abstract ones. For general processes, Sadiq and Orlowska (2000) have applied graph reduction techniques in validating the correctness of a business process structure. Some structural validity rules in this paper are inspired by their work, while our work extended a lot on structural validation on composite processes.

In the area of inter-organisational collaboration, process views also play an important role in privacy protection. van der Aalst and Weske (2001) proposed a "top-down" workflow modelling scheme in their public-to-private approach. In this scheme, organisations first agreed on a public workflow, and later each organisation refined the part it was involved in, and thereby generated its private workflow. Schulz and Orlowska (2004) focused on the cross-organisational interactions, and proposed to deploy coalition workflows to compose private workflows and workflow views together to enable interoperability. Chiu et al. (2004) adapted the view concept from database systems, and employed a virtual workflow view to hide internal information. The virtual workflow view only presents the information necessary for process enactment, enforcement and monitoring, instead of all details. In regard to process interoperability within virtual enterprises, Perrin and Godart (2004) used synchronisation points between process services to coordinate collaboration, and thereby allowed partners to personalise their internal processes without affecting the cooperation. Issam et al. (2006) extracted an abstract workflow view to describe the choreography of a collaboration scenario and compose individual workflows into a collaborative business process, and in that way partial visibility of workflows and resources are enabled. Our previous works (Xiaohui Zhao & Liu, 2010, 2013; Xiaohui Zhao, Liu, Yang, & Sadiq, 2009, 2011) also established a relative workflow model for collaborative business process modelling. A relative workflow comprises the local workflow processes of the host organisation and the filtered workflow process views from its partner organisations. In this way, it can provide a relative collaboration context for each participating organisation. Compared with these works, this paper motivated process views from the perspective of role-based perception control, and analysed the view derivation and composition according to the role hierarchy and interactions between roles.

Kopka and Wellen (2002) have touched the topic of role-based process views in the domain of multimedia system development process. As a preliminary work towards this topic, their work proposed the idea of creating different logical views of the same business process for different involved roles, without further exploring the support to automatic view generation or formal process perception description. Work by Shen and Liu (2003) further explored the relevance between roles and the influence to process views. Permission rules were used in their approach to describe the relationships among roles, tasks and operations (view or execute). Yet, the interaction among roles and the corresponding evolution of a role's perception over a business process is not touched.

Process structural consistency also attracted some research efforts. D.-R. Liu and Shen (2003); D. R. Liu and Shen (2004) proposed an order-preserving approach to derive a structurally consistent process view from a base business process. In their approach, the generation of "virtual activities" (compound tasks) need to follow their proposed membership rule, atomicity rule, and order preservation rule. Eshuis and Grefen (2008) formalised the operations of task aggregation and process customisation, and also proposed a series of construction rules for validating the structural consistency. Most of these researches concentrated on process view filtering only, while our approach covered both process view filtering and composition operations with a richer set of rules and theorems. Besides, Petri net (van der Aalst, 2003) and process algebra (Busi, 2006) are two popular mathematic tools for structural and semantic analysis of processes. From our experience with these tools (X. Zhao et al., 2011; Xiaohui Zhao et al., 2011), Petri net has speciality in rigorously presenting the concurrent structures of processes, and thereby is suitable for validating structural soundness. Yet the size of Petri net increases exponentially when the process tends to be complex. Process algebra is particularly useful in proving semantic equivalence between processes with different structures. However, process algebra struggles in intuitively representing the structural transformation of a process, because it does not have a standard or easy-to-read graphical format. Due to these reasons, we stick to conventional flow chart (adapted to BPMN format) for process representation in this paper, plus BPMN is becoming overwhelmingly popular in process modelling in industry.

Some other work like (Bobrik, Reichert, & Bauer, 2007) adopted process views for process visualisation, and may relax some structural constraints to adapt to actual user requirements. Küster, Gerth, Forster, and Engels (2008) have investigated the techniques for consolidating and merging processes from the perspective of process change and version management. With this perspective, their work focuses on how to merge the changes made by different process users to the same business process, rather than dynamically generating/updating process perceptions according to interactions among process users. In software engineering domain, semantic view of program execution holds a similar philosophy with process views, as it was proposed to reflect the projections of execution traces at different abstraction levels. As the founders of semantic views, Hoffman, Eugster, and Jagannathan (2009) have implemented semantics views by selectively aggregating collections of events with shared semantic traits found in a program execution trace mainly for software debugging purpose. In regard to information abstraction, this work and our work share similar philosophy. Yet, semantic views concentrate on the semantic equivalence between execution descriptions specific to certain programing languages, instead of business logics in business processes. In comparison, our work focuses more on process perceptions of different roles, structural consistency between base process and transformed views, and view transformation according to perception evolutions.

Our framework systematically analysed the derivation and composition of process views with a role-based process view model, and provided a set of process view operations which for the first time supports process view merging. As a pioneer wok in this area, our framework established the foundation for process view transformation, including validation rules, consideration on information loss during view merging, enabling algorithms for automatic view generation.

As a typical artefact from the perspective of design science research (Gregor & Hevner, 2013), our framework can be evaluated in terms of its validity, utility, quality and efficacy as follows.

1. Validity – The framework has been examined by experienced industry experts to ensure it meets the goal, i.e., better facilitate process view management with supports to process view mergers. At conceptual level, a hypothetical case is used to demonstrate the functionality of the framework. At technical level, a prototype has been implemented to prove the feasibility of the framework.
2. Utility – The domain experts from our industry partners have identified more applications of the framework outside the original motivation. A typical example is in the scenario of business process co-creation, different roles, such as process architect and business analyst, work together to create a business process yet they view the same process with different focuses, e.g., data dependency and value chain embedded in the business process, respectively. This co-creation can be well supported by our process view framework. This new application strongly evidences the value of our work. The proposed process view support is to be integrated into SAP's next generation ERP system.
3. Quality – As the first attempt to analysing and supporting process view merging, our work explicitly discusses the information loss phenomenon for the first time, and distinguishes the difference between process view filtering and merging operations in terms of their expressiveness of information dependency resulting in information loss. Further, the framework can trace process perception evolution with the help of perception authorisation and inheritance operations. The proposed rules and view operations fully guarantee the structural correctness of transformed views and the proper information reservation and loss during the transformation. The framework is rigorously formalised and grounded on a theoretical foundation to ensure the accuracy and soundness of the process view and perception model.
4. Efficacy – Our process view transforming mechanism natively supports information loss occurred in process mergers without re-extracting another view from the base business process. In comparison, view filtering based approaches awkwardly need to analyse the dependences between the visibility of a process element and the process perceptions of involved roles, and then regenerate the view from the base process to correctly discard certain process information, in order to comply with the information loss rule.

The proposed framework was established on the basis of a series of restrictions, which resulted in some inherent limitations. Here, we discussed about these limitations as follows.

A. The view merging operation may need fine tuning to better its efficiency. A potential improvement could be done by lowering the process perception definition from task level (please refer to Definition 4 in Section 3) down to the level of any visible process elements (including tasks, links, gateways, etc.), to enable process view filtering techniques for process view merger. Yet, this will considerably increase the complexity of defining a view perception.
B. The process view model only works with well-formed (i.e., block-structured) processes or views. Yet, the block structure is already a restriction in BPEL, and therefore all BPEL processes comply with this restriction. In addition, the restriction on well-formedness is likely to be sidestepped by converting free style modelled (non-well-formed) business processes into well-formed business

processes, and then using the latter for process view manipulation. In this area, some work has already been done in attempt to convert Business Process Modelling Notations (BPMN) diagrams into block-structured BPEL processes (Doux, Jouault, & Bézivin, 2009; Ouyang, Dumas, Aalst, Hofstede, & Mendling, 2009), which seems to be a good solution to this issue.

8. CONCLUSIONS

This paper proposed a role-based process view model and analysed the process view derivation and composition. This work emphasised the process perception dependency and evolution of different roles, and the influence to process views. A set of rules and theorems were defined to regulate the process view transformations to guarantee the structural validity and consistency. As a bridge, this work bridged conceptual perception relations and technical process view transformations, and thereby furthers the application of business process management.

Our future work includes applying the process view model in supporting the cooperation between different process users, such as business analysts and process architects, who have different interests with the process presentation.

9. APPENDIX

In this appendix, the enabling algorithms for process view generation are presented.

- Introduction to involved functions

preN(n) and *postN*(n) return the set of immediate preceding and succeeding tasks and gateways of node n, respectively, where n is a task or gateway;

type(n) returns the type of n, where n is a task or gateway, the possible values for *type* include *Normal*, *Start*, *End*, *And-Split*, *And-Join*, *Or-Split*, and *Or-Join*;

pair(g) returns the corresponding gateway in pair with gateway g, i.e., the corresponding split gateway if g is a join gateway, or the corresponding join gateway if g is a split gateway;

Tasks(v, g_1, g_2) returns all the tasks contained in the *Split/Join* structure specified by gateways g_1 and g_2 in view v;

COSP(v, t) returns the closest preceding *Or-Split* gateway of task t in view v;

CASP(v, t) returns the closest preceding *And-Split* gateway of task t in view v;

CAJS(v, t) returns the closest succeeding *And-Join* gateway of task t in view v.

- Algorithms

Algorithm 1. *filter*(v, T) filters off tasks in set T from view v and adjusts the remaining links and gateways according to the view filtering operation discussed in Section 4.3.

Input v – the input process view;
T – the set of tasks for filtering;
Output v – the result process view.

1 $v.T$=$v.T \setminus T$;
2 **while** ($\exists l$=(t_1, t_2)$\in v.L$, $t_1 \in T$ or $t_2 \in T$)
3 $v.L$=$v.L \setminus \{l\}$;
4 **if** $t_1 \in T$ **then**
5 $v.L$=$v.L \cup \{(n_a , t_2) \mid n_a \in preN(t_1)\}$;
6 **if** (∃sync link sl=(x, t_1)$\in v.SL$ **then** $v.SL$=$v.SL \cup \{(x, t_2)\} \setminus \{sl\}$;
7 **else**

```
 8      v.L=v.L∪{(t1, nb) | nb∈postN(t2)};
 9      if (∃sync link sl=(t2, x)∈v.SL then v.SL=v.SL∪{(t1, x)}\{sl};
10    end if
11  end while
12  for each g∈v.G, type(g)=And-Split
13    if ∃l=(g, pair(g))∈v.L then v.L=v.L\{l};
14  for each g∈v.G, type(g) )∈{And-Split, Or-Split} AND outd(g)≤1
15    if (outd(g)=0) OR (outd(g)=1 AND ∃l=(g, pair(g))∈v.L) then
16      v.L=v.L∪{( na, nb) | na∈preN(g), nb∈postN(pair(g)};
17        while(∃sync link sl=(x, g)∈v.SL)
18            v.SL=v.SL∪{(x, postN(pair(g)))}\{sl};
19        end while
20        while (∃sync link sl=(pair(g), x)∈v.SL)
21            v.SL=v.SL∪{( na, x)| na∈preN(g)}\{sl};
22        end while
23    else
24        v.L=v.L∪({(na, nb)|na∈preN(g), nb∈postN(g)}∪{(na, nb)|na∈(preN(pair(g)),
    nb∈postN(pair(g)});
25        while(∃sync link sl=(x, g)∈v.SL)
26            v.SL=v.SL∪{(x, nb) |nb∈postN(g)}\{sl};
27         end while
28        while (∃sync link sl=(pair(g), x)∈v.SL)
29            v.SL=v.SL∪{(na, x)| na∈preN(pair(g))}\{sl};
30        end while
31    end if
32      v.L=v.L\({(na, g)| na∈preN(g)}∪{(pair(g), nb)| nb∈postN(pair(g)});
33      v.G=v.G\{g, pair(g)};
34  end for
35  v.SL=v.SL\v.L;
36  return v;
```

Lines 2-11 reconnect the links involved with the removed tasks, according to Rules 1 and 2, while Lines 6 and 9 adjust the synchronisation links according to Rule 3. Lines 12-13 delete the single dummy branches from And-Split/Join structures, according to Rule 5. Here, the dummy branches in an Or-Split/Join structure are already combined into one dummy branch, due to the definition of set operation. Lines 14-33 check if any Split/Join structures degrade into sequential structures after the removal of dummy branches, according to Rules 4 and 5, while Lines 17-22 and Lines 25-30 adjust synchronisation links according to Rule 3.

Algorithm 2. *matchOr-Split/Join*(v_1, v_2) matches the *Or-Split/Join* structures of views v_1 and v_2, and returns the combined view. This algorithm corresponds to the first step for the view merging operation discussed in Section 4.3.

```
Input    v1 – an input process view;
         v2 – another input process view;
Output   v – the result process view.
 1  T°=v1.T∩v2.T; G°=∅;
 2  for each t∈T°
 3    if t is involved in an Or-Split/Join structure then
 4      g1=COSP(v1, t); g2=COSP(v2, t);
```

```
5     while (g1∉G° AND g1≠null)
6       G°= G°∪{g1}; g1′=pair(g1) in v1; g2′=pair(g2) in v2;
7           v2.G=v2.G∪{g1}\{g2}; v2.G=v2.G∪{g1′}\{g2′};
8       P=Tasks(v1, g1, g1′); Q=Tasks(v2, g2, g2′);
9       T°= T°\(P∩Q);
10      for each tx∈postT(g2) in v2
11        v2.L=v2.L∪{(g1, tx)}\{(g2, tx)};
12      for each tx∈preT(g2′) in v2
13        v2.L=v2.L∪{(tx, g1′)}\{(tx, g2′)};
14      let l=(tx, g2)∈v2.L; v2.L=v2.L∪{(tx, g1)}\{l};
15      let l=(g2′, ty)∈v2.L; v2.L=v2.L∪{(g1′, ty)}\{l};
16      let sl=(tx, g2)∈v2.SL; v2.SL=v2.SL∪{(tx, g1)}\{sl};
17      let sl=(g2′, ty)∈v2.SL; v2.SL=v2.SL∪{(g1′, ty)}\{sl};
18      if (∃l1=(g1, g1′)∈v1.L) AND ¬(∃l2=(g2, g2′)∈v2.L) then v1.L=v1.L\{l1};
19      if ¬ (∃l1=(g1, g1′)∈v1.L) AND (∃l2=(g2, g2′)∈v2.L) then v2.L=v2.L\{l2};
20      g1=COSP(v1, g1); g2=COSP(v2, g2);
21    end while
22  end if
23 end for
24 v.L=v1.L∪v2.L; v.T=v1.T∪v2.T; v.SL=v1.SL∪v2.SL; v.G=v1.G∪v2.G;
25 return v;
```

This algorithm iteratively checks the common tasks belonging to v_1 and v_2, and matches the involved *Or-Split/Join* structures with Lines 3-23. Lines 7-17 replace the involved *Or-Split/Join* gateways in v_2 with the corresponding ones in v_1. Lines 18-19 handle the dummy branches of *Or-Split/Join* structures according to Rules 4 and 6. Line 24 combines the constitute sets of views v_1 and v_2 into the ones of result view v. The returned view is the result after matching *Or-Split/Join* structures.

Algorithm 3. *cleanRedundantLinks*(v) removes redundant links in view v, according to Rule 7. This algorithm corresponds to the procedure of removing redundant links of the second step for the view merging operation discussed in Section 4.3.

```
Input   v – the input process view;
Output  v – the result process view.
1  N°={s}; L′=v.L;
2  for each n∈N°
3     N°= N°\{n};
4    for each l=(n1, n2) ∈ L′
5      if n=n1 then
6        L′= L′\{l};
7      if n2∉N° then N°= N°∪{n2} else v.L= v.L\{l};
8      end if
9  end for
10 return v;
```

Algorithm 4. *addAnd-Split/JoinGateways*(v) adds *And-Split/Join* gateways to view v to connect the tasks or gateways which have excessive links, due to the view combination. Proper synchronisation links may be generated to sort the execution order of tasks between unpaired *And-Split/Join* gateways. This algorithm corresponds to the third step for the view merging operation discussed in Section 4.3.

Input v – the input process view;
Output v – the result process view.

1 **while** (($\exists n \in v.T \cup v.G$ such that ($ind(n)>1$ OR $outd(n)>1$) AND $type(n) \in$ {start, normal, end}) OR ($ind(n)>1$ AND $type(n) \in$ {*And-Split*, *OR-Split*}) OR ($outd(n)>1$ AND $type(n) \in$ {*And-Join*, *OR-Join*})
2 **if** $ind(n)>1$ **then**
3 **create** *And-Join* gateway g; $v.G=v.G \cup \{g\}$;
4 **for each** $n_x \in preN(n)$ in v
5 $v.L=v.L \cup \{(n_x, g)\} \setminus \{(n_x, n)\}$;
6 $v.L=v.L \cup \{(g, n)\}$;
7 **end if**
8 **if** $outd(n)>1$ **then**
9 **create** *And-Split* gateway g; $v.G=v.G \cup \{g\}$;
10 **for each** $n_x \in postN(n)$ in v
11 $v.L=v.L \cup \{(g, n_x)\} \setminus \{(n, n_x)\}$;
12 $v.L=v.L \cup \{(n, g)\}$;
13 **end if**
14 **end while**
15 **while** ($\exists g_1, g_2 \in v.G$ such that $type(g_1)=$*And-Split*, $type(g_2)=$*And-Join* AND g_1, g_2 do not construct a closed *Split/Join* structure AND $\exists$path $p=(g_1, l_1, \ldots, ln, g_2)$)
16 **let** $l_1=(g_1, n_x) \in v.L$; **let** $l_n=(n_y, g_2) \in v.L$; $v.L=v.L \setminus \{l_1, l_n\}$;
17 **if** $type(pre(g_1)) \notin$ {*And-Split*, *Or-Split*} **then** $v.SL=v.SL \cup \{(n_a, n_x) \mid n_a \in preN(g_1)\}$;
18 **if** $type(post(g_2)) \notin$ {*And-Join*, *Or-Join*} **then** $v.SL=v.SL \cup \{(n_y, n_b) \mid n_b \in postN(g_2)\}$;
19 **end while**
20 **return** v;

Lines 1-14 check each illegal *Split/Join* structure, and insert proper *And-Split/Join* gateways. Lines 15-19 check for the paths exist between two unpaired *And-Split/Join* gateways, and break the paths by converting proper links into synchronisation links. Lines 17-18 check the type the adjacent node before converting a link into a synchronisation link, according to Rule 9.

Algorithm 5. *checkAndSplit/JoinStruc*(v) examines the tasks and gateways inside *And-Split/Join* structures in view v in terms of incoming/outgoing degrees. $preT(n)$ and $postT(n)$ will return the sets of immediate preceding and succeeding tasks of node n, respectively, where can be a task or a gateway. This algorithm corresponds to the fifth step for the view merging operation discussed in Section 4.3.

Input v – the input process view;
Output v – the result process view.

1 **while** (($\exists g \in v.G$ such that $type(g)=$*And-Split* AND $outd(g)=1$) OR ($\exists g \in v.G$, $type(g)=$*And-Join* AND $ind(g)=1$))
2 $v.G=v.G \setminus \{g\}$;
3 $v.L=v.L \cup \{(n_a, n_b) \mid n_a \in preN(g), n_b \in postN(g)\} \setminus \{(n_a, g), (g, n_b) \mid n_a \in preN(g), n_b \in postN(g)\}$;
4 **if** $\exists sl=(g, n_x) \in v.SL$ **then** $v.SL=v.SL \cup \{(n_a\ n_x) \mid n_a \in preN(g)\} \setminus \{sl\}$;
5 **if** $\exists sl=(n_y, g) \in v.SL$ **then** $v.SL=v.SL \cup \{(n_y, n_b) \mid n_b \in postN(g)\} \setminus \{sl\}$;

6 **end while**
7 **while** ($\exists g \in v.G$ such that $type(g)$=*And-Split* AND $\exists$tasks t_x, $t_y \in postT(g)$ such that $CAJS(t_x) \neq CAJS(t_y)$)
8 **let** TX={t | $t \in postT(g)$, such that $CAJS(t)=g_1$ AND ($\forall tx \in postT(g)$, $CAJS(tx)=g_1$ OR $before(g_1, CAJS(t_x))$)};
9 **create** *And-Split* gateway g_a;
10 $v.G=v.G \cup \{g_a\}$; $v.L=v.L \cup \{(g, g_a)\}$;
11 **for each** $t \in TX$
12 $v.L=v.L \cup \{(g_a, t)\} \setminus \{(g, t)\}$;
13 **end while**
14 **while** ($\exists g \in v.G$ such that $type(g)$=*And-Join* AND $\exists$tasks t_x, $t_y \in preT(g)$ such that $CASP(t_x) \neq CASP(t_y)$)
15 **let** TY={t | $t \in preT(g)$ such that $CASP(t)=g_1$ AND ($\forall tx \in preT(g)$, $CASP(tx)=g_1$ OR $before(CASP(tx), g_1)$)};
16 **create** *And-Join* gateway g_b;
17 $v.G=v.G \cup \{g_b\}$; $v.L=v.L \cup \{(g_b, g)\}$;
18 **for each** $t \in TY$
19 $v.L=v.L \cup \{(t, gb)\} \setminus \{(t, g)\}$;
20 **end while**
21 **while** ($\exists t \in v.T \setminus \{s, e\}$ such that $ind(t)=0$ OR $outd(t)=0$)
22 **if** $ind(t)=0$ **then let** $sl=(tx, t) \in v.SL$; $v.L=v.L \cup \{(CASP(tx), t)\}$;
23 **else let** $sl=(t, tx) \in v.SL$; $v.L=v.L \cup \{(t, CAJS(tx))\}$;
24 **end if**
25 **end while**
26 **return** v;

Lines 1-6 check the *And-Split/Join* gateways with only one incoming or outgoing link, and adjust related links and synchronisation links according to Rules 1 and 3. Lines 7-13 and lines 14-20 complement *And-Split* gateways and *And-Join* gateways, respectively, according to Rule 8. Lines 21-25 check the tasks without outgoing or incoming links.

With the aforementioned algorithms, operation $merge(v_1, v_2)$ can be easily realised by invoking *matchOr-Split/Join*(v_1, v_2), *cleanRedundantLinks*(v), *addAnd-Split/JoinGateways*(v) and *checkAndSplit/JoinStruc*(v) in order.

REFERENCES

Andrews, Tony, Curbera, Francisco, Dholakia, Hitesh, Goland, Yaron, Klein, Johannes, Leymann, Frank, . . . Weerawarana, Sanjiva. (2003). Business Process Execution Language for Web Services (BPEL4WS) 1.1.

Bobrik, Ralph, Reichert, Manfred, & Bauer, Thomas. (2007). *View-Based Process Visualization*. Paper presented at the 5th International Conference on Business Process Management, Brisbane, Australia.

Busi, Nadia (Producer). (2006). Process Algebras, Bisimulation (and Logics). Retrieved from http://www.cs.purdue.edu/homes/jv/events/TiC06/B-SLIDES/nb.pdf

Caetano, Artur, Zacarias, Marielba, Silva, António Rito, & Tribolet, José M. (2005). *A Role-Based Framework for Business Process Modeling*. Paper presented at the 38th Hawaii International Conference on System Sciences, Big Island, HI, USA.

Chiu, Dickson K.W., Cheung, S.C., Till, Sven, Karlapalem, Kamalakar, Li, Qing, & Kafeza, Eleanna. (2004). Workflow View Driven Cross-Organizational Interoperability in a Web Service Environment. *Information Technology and Management, 5*(3-4), 221-250.

Choi, Jae, Nazareth, Derek, & Jain, Hemant. (2013). The Impact of SOA Implementation on IT-Business Alignment: A System Dynamics Approach. *ACM Transactions on Management Information Systems, 4*(1).

Doux, Guillaume, Jouault, Frédéric, & Bézivin, Jean. (2009). *Transforming BPMN process models to BPEL process definitions with ATL*. Paper presented at the 5th International Workshop on Graph-Based Tools, Zurich, Switzerland.

Eshuis, Rik, & Grefen, Paul. (2008). Constructing Customized Process Views. *Data & Knowledge Engineering, 64*, 419-438.

Georgakopoulos, Diimitrios, Hornick, Mark, & Sheth, Amit. (1995). An overview of workflow management: From process modeling to workflow automation infrastructure. *Distributed and parallel Databases, 3*(2), 119-153.

Gregor, Shirley, & Hevner, Alan R. (2013). Positioning and Presenting Design Science Research for Maximum Impact. *MIS Quarterly, 37*(2), 337-355.

Hoffman, Kevin J., Eugster, Patrick, & Jagannathan, Suresh. (2009). *Semantics-aware Trace Analysis*. Paper presented at the ACM SIGPLAN conference on Programming Language Design and Implementation, Dublin, Ireland.

Issam, Chebbi, Schahram, Dustdar, & Samir, Tata. (2006). The View-Based Approach to Dynamic Inter-Organizational Workflow Cooperation. *Data & Knowledge Engineering, 56*(2), 139-173.

Kopka, Corina, & Wellen, Ursula. (2002). *Role-Based Views to Approach Suitable Software Process Models for the Development of Multimedia Systems*. Paper presented at the 4th International Symposium on Multimedia Software Engineering, Newport Beach, CA, USA.

Kumar, Akhil, & Zhao, J. Leon. (1999). Dynamic Routing and Operational Controls in Workflow Management Systems. *Management Science, 45*(2), 253-272.

Küster, Jochen M., Gerth, Christian, Forster, Alexander, & Engels, Gregor. (2008). *A Tool for Process Merging in Business-Driven Development* Paper presented at the 20th International Conference on Advanced Information Systems Engineering Forum, Montpellier, France.

Liu, Chengfei, Li, Qing, & Zhao, Xiaohui. (2008). Challenges and Opportunities in Collaborative Business Process Management. *Information System Frontiers, 11*(3), 201-209.

Liu, Duen-Ren, & Shen, Minxin. (2003). Workflow Modeling for Virtual Processes: an Order-Preserving Process-View Approach. *Information Systems, 28*(6), 505-532.

Liu, Duen Ren, & Shen, Minxin. (2004). Business-to-Business Workflow Interoperation based on Process-Views. *Decision Support Systems, 38*(3), 399-419.

Martens, Axel. (2005). *Consistency between Executable and Abstract Processes.* Paper presented at the 7th IEEE International Conference on e-Technology, e-Commerce, and e-Services, Hong Kong, China.

Ouyang, Chun, Dumas, Marlon, Aalst, Wil M. P. van der, Hofstede, Arthur H. M. ter, & Mendling, Han. (2009). From Business Process Models to Process-oriented Software Systems: The BPMN to BPEL Way. *ACM Transactions on Software Engineering and Methodology, 19*(1).

Perrin, O., & Godart, C. (2004). A Model to Support Collaborative Work in Virtual Enterprises. *Data & Knowledge Engineering, 50*, 63-86.

Reichert, Manfred, & Dadam, Peter. (1998). ADEPTflex: Supporting Dynamic Changes of Workflow without Loosing Control. *Journal of Intelligent Information Systems, 10*, 93-129.

Sadiq, Wasim, & Orlowska, Maria E. (2000). Analyzing Process Models Using Graph Reduction Techniques. *Information Systems, 25*(2), 117-134.

Schulz, K.A., & Orlowska, M.E. . (2004). Facilitating Cross-organisational Workflows with a Workflow View Approach. *Data & Knowledge Engineering, 51*(1), 109-147.

Shen, Minxin, & Liu, Duen-Ren. (2003). *Discovering Role-Relevant Process-Views for Recommending Workflow Information.* Paper presented at the 14th International Conference on Database and Expert Systems Applications, Prague, Czech Republic.

Shostack, Lynn. (1984). Designing Services that Deliver *Harvard Business Review, 62*(1), 133-139.

Stohr, Edwards, & Zhao, J. Leon. (2001). Workflow Automation: Overview and Research Issues. *Information Systems Frontiers, 3*(3), 281-296.

Ullah, Azmat, & Lai, Richard. (2013). A Systematic Review of Business and Information Technology Alignment. *ACM Transactions on Management Information Systems, 4*(1).

van der Aalst, W.M.P. (2003). Challenges in Business Process Management: Verification of business processes using Petri nets. *The Bulletin of the European Association for Theoretical Computer Science, 80*, 174-198.

van der Aalst, W.M.P., & Weske, Mathies. (2001). *The P2P Approach to Interorganizational Workflows.* Paper presented at the 13th International Conference on Advanced Information Systems Engineering.

Weske, M., van der Aalst, W. M. P., & Verbeek, H. M. W. (2004). Advances in Business Process Management. *Data & Knowledge Engineering, 50*(1), 1-8.

Yongchareon, Sira, Liu, Chengfei, Yu, Jian, & Zhao, Xiaohui. (2015). A View Framework for Modeling and Change Validation of Artifact-Centric Inter-Organizational Business Processes. *Information Systems, 47*(1), 51-81.

Yongchareon, Sira, Zhao, Xiaohui, Liu, Chengfei, & Kowalkiewicz, Marek. (2008). FlexView Manual - Supports for Process View Operations (Technical Report).

Zhao, X., Liu, C., Sadiq, W., Kowalkiewicz, M., & Yongchareon, S. (2011). Implementing Process Views in the Web Service Environment. *World Wide Web, 14*(1), 27-52.

Zhao, Xiaohui, & Liu, Chengfei. (2010). Steering Dynamic Collaborations between Business Processes. *IEEE Transactions on Systems, Man and Cybernetics, 40*(4), 743-757.

Zhao, Xiaohui, & Liu, Chengfei. (2013). Version Management for Business Process Schema Evolution. *Information Systems, 38*(8), 1046-1069.

Zhao, Xiaohui, Liu, Chengfei, Sadiq, Wasim, & Kowalkiewicz, Marek. (2008). *Process View Derivation and Composition in a Dynamic Collaboration Environment.* Paper presented at the 16th International Conference on Cooperative Information Systems, Monterrey, Mexico.

Zhao, Xiaohui, Liu, Chengfei, Yang, Yun, & Sadiq, Wasim. (2009). Aligning Collaborative Business Processes: An Organisation-oriented Perspective. *IEEE Transactions on Systems, Man and Cybernetics, 39*(6), 1152-1164.

Zhao, Xiaohui, Liu, Chengfei, Yang, Yun, & Sadiq, Wasim. (2011). CorPN: Managing Instance Correspondence in Collaborative Business Processes. *Distributed and Parallel Database, 29*(4), 309-332.

Received June 2013; revised November 2014; accepted March 2015

10. BIOGRAPHIES

Xiaohui Zhao is an assistant professor in Information Systems at University of Canberra, Australia, since 2012. He received his Ph.D. in 2007 from Swinburne University of Technology, Australia. Since then till 2011, he has been working as a postdoctoral research fellow in Swinburne University of Technology and Eindhoven University of Technology, the Netherlands. He held a lecturer position in the department of Computing at Unitec Institute of Technology, New Zealand from 2011 to 2012, and received a TechNZ grant from the Ministry of Science and Innovation, New Zealand. His research interests include business process modelling and analysis, service composition and outsourcing, etc. He has published over 50 papers in prestigious journals and conferences so far.

Chengfei Liu received the BS, MS and PhD degrees in Computer Science from Nanjing University, China in 1983, 1985 and 1988, respectively. Currently he is a Professor in the Faculty of Science, Engineering and Technology, Swinburne University of Technology, Australia. His current research interests include keywords search on structured data, query processing and refinement for advanced database applications, query processing on uncertain data and big data, and data-centric workflows. He is a member of IEEE, and a member of ACM.

Sira Yongchareon is a Lecture at Department of Computing, Unitec Institute of Technology, New Zealand. He has received his PhD and M.IT from Swinburne University, Melbourne, Australia in 2012 and 2008 respectively. Prior to that, he has worked in software industry for more than seven years. His research specialization is in the area of business process management and information systems with a particular focus on developing an artifact-centric approach to modeling business processes. He has published several ERA-A/A*-ranked conference papers and journals in the field of Information Systems and Business Process Management. He has served as a program committee of several international conferences and as an invited journal reviewer, such as WWW, IEEE TSMC, IEEE CEC, IJCIS, BPM, WISE, DASFAA.

Marek Kowalkiewicz received his PhD from Poznan University of Economics, Poland. He is the senior director of products and innovation, SAP, Palo Alto, California, establishing and driving the conditions for developers at SAP to be happy, creative and pround. Before this, he has been working for SAP in different branches over the world, and has solid experience in the areas of enterprise applications, business process management, service-oriented architecture, etc. He has published a series of papers in reputed international journals and conferences.

Wasim Sadiq received the Ph.D. degree in computer science from the University of Queensland, Brisbane, Australia, in the area of conceptual modeling and verification of workflows. He is with the Infosys as the general manager in research. He was the vice president of SAP Research Centre, Brisbane, Australia. He has over 22 years of research and development experience in the areas of enterprise applications, business process management, workflow technology, service-oriented architectures, database management systems, distributed systems, and e-learning. He has published and presented several research papers in leading international conferences and journals and has filed more than 20 patent applications. He has led several

research projects collaborating with academic and industry partners in Australia, Europe, and the U.S.