ON GROUPING OF ACTIVITIES INSTANCES IN WORKFLOW MANAGEMENT SYSTEMS

Dat C. Ma, Joe Y.-C. Lin and Maria E. Orlowska
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, Australia

Keywords: Grouping, Activity instances, Work items, Flexibility, Workflow.

Abstract: Current research in the flexibility of workflow management systems covers many aspects of this technology. The focus of this paper is primarily on the practical capabilities of workflow management systems in handling preferred work practice while dealing with many short duration activities. It is motivated by the requirement of merging or grouping work items by one performer to achieve work performance enhancements by avoiding unnecessary communication with the system but still executing the required activities. The paper proposes a new function to group activity instances for a given process, investigates the impact, benefits, and potential implementation of such of extended functionality.

1 INTRODUCTION

Workflow technology has been considered being the most common technology in supporting the automating of business processes. It has delivered effectively in the area of business and scientific process enforcement, which offered a clear separation of process logic from component applications and data involved in process execution. The technology has primarily offered productivity improvements mainly for repetitive business processes with substantial human involvement, and has provided strict business policy enforcement, effective scheduling, monitoring, and resource planning services.

However, one of the most critical limitations of current workflow technology is its rigour in executing predefined process structures. It is often discussed under the name of lack of flexibility (Aalst 1999), (Sadiq 1999), (Sadiq, Orlowska & Sadiq 2005), (Sadiq et al. 2005). In general, the scope of the term "flexibility" in workflow is quite diverse, which ranges from adaption to evolution of business process models and exception handling (Aalst 1999), (Sadiq, Orlowska & Sadiq 2005) to the requirement for specifying workflow patterns (Aalst et al. 2003) and the work practice conformance (Sadiq et al. 2005). Due to the well motivated challenges and potential benefits from effective handling flexibility in workflow technology, related research attracted lots of attention.

In this paper, we focus on another flexibility aspect of workflow technology; in supporting user driven grouping of work items (within one activity). Our aim is to address this flexibility aspect without impacting on the semantics of current workflow specifications, but to give users the freedom to group work items at the runtime. This work is motivated by the requirement to handle many short duration tasks, the way work as individuals executing them would prefer to do – without additional overhead from deployment of workflow management systems (WFMS). This issue was first raised by (Sadiq et al. 2005). In this paper, we show the distinction between two work items amalgamation principles: grouping several work items of the same activity into a new integrated work item without any intervention to the involved activity instances data versus merging several activity instances into one new instance subsuming data from all its components. We demonstrate how much can be achieved without serious modifications of the process specification language semantics and what impact on 'off shelf WFMS' such introduction must have.

In the following sections, we provide the basic related concepts in workflow technology. Section 3 proposes an approach to extend the functionality of

C. Ma D., Y. -C. Lin J. and E. Orlowska M. (2007).
ON GROUPING OF ACTIVITIES INSTANCES IN WORKFLOW MANAGEMENT SYSTEMS.
In *Proceedings of the Ninth International Conference on Enterprise Information Systems - ISAS*, pages 125-130
DOI: 10.5220/0002369801250130

WFMS by allowing grouping of activity instances. In section 4, we discuss a potential implementation of such extension in traditional WFMS environment. The conclusion and future work are presented in section 5.

2 BASIC TERMINOLOGY

In order to provide a background for further discussion, we define some basic terminologies.

Let a workflow process W be defined as W = <N, F> where N: finite set of nodes, F: flow relation $F \subseteq N \times N$. Further, we define two functions:

- $\forall n \in N$, NodeType: $n \rightarrow$ {Coordinator, Task}, such that $N = C \cup T$, $C \cap T = \phi$; where C: Set of Coordinators, T: Set of Tasks (or Activities).
- $\forall c \in C$, CoordType: $c \rightarrow$ {AND-Split, AND-Join, XOR-Split, XOR-Join, Begin, End}

A process model will have many activities. A *process instance* represents a particular case of the process. An *activity instance* is the representation of an activity within a process instance. Each activity instance is governed by a finite state machine (FSM) that is characterised by typical states (e.g. Scheduled, Active, Suspended, Completed, and Terminated).

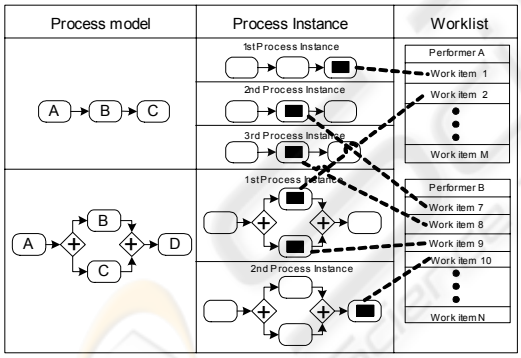

Figure 1: Relationships between key terminologies.

During the run time, a *workflow performer* (*participant*) performs the work by selecting a *work item* from a *worklist*. A work item is the representation of the work in the context of an activity instance. The worklist forms part of the user's interface between the workflow engine and the worklist handler. A *Worklist Handler* is a software component managing the interaction between the users and the worklist. Figure 1 shows the relationships between some key terminologies (WfMC 1999) using the Business Process Modelling Notation (or BPMN) (OMG 2004).

3 GROUPING WORK ITEMS

The notion of an activity instance in workflow system is a useful concept to separate individual fragments/phases of complex process execution. However, this notion can be too restrictive for certain scenarios, in particular when dealing with large loads of short-duration work items of the same type. We provide a motivating scenario that has been identified in real work practice showing an overhead (not contributing to the process itself) but imposed by the deployment of WFMS solution.

3.1 Motivating Example

One of the most widely used processes for automation by workflow technology is sale order processing. Figure 2 illustrates a simplified version of a typical order processing scenario.

Figure 2: Order Processing Workflow.

Consider the creation of purchase order managed by workflow as in Figure 2. Note that the Create Purchase Order requires a merchant to commence and complete the same activity for each individual purchase request. Given the current state of art in workflow technology, the performer interacts with the worklist handler for 1 work item at a time, which is proved to be a cumbersome task. A much preferred work practise would be to do this activity for a group of purchase order with a single point of interaction with the worklist handler. For example, grouping 3 purchase orders (i.e. 3 work items) into one group will reduce the number of interactions between the user and the worklist handler from 6 (i.e. 2 for each work item) into 2, which is not supported by current WFMS.

We now differentiate between *merging* and *grouping* of work items. The main difference between these two functions could be illustrated as in Figure 3. In Figure 3, PR and PO represent multiple work items appear on the worklist of activities Create Purchase Request and Create Purchase Order. Each work item has an associated content. For instance, the work item PR2 requests to order two products B and C with their respective values (i.e. 30 and 20) as its content.

As illustrated in Figure 3, when merging work items, a new work item is created as a result of merging (i.e. PO*) for PR.1 and PR.2. While in the case of grouping, a group of work items is created.

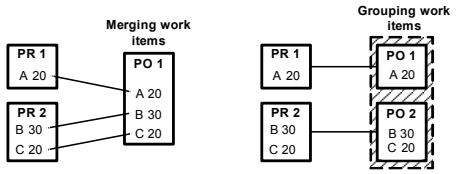

Figure 3: Merging and Grouping of work items.

If we denote t as a task and WI_t the set of all work items of task t across multiple instances, then we define the *Merge* and *Group* functions for work items as followed.

Merge: $WI_t^2 \rightarrow WI_t$, where WI_t^2 is the power set of WI_t. When the merge function is applied on a subset of WI_t, it will replace the subset by creating a new work item $wi_t \in WI_t$. The newly created work item wi_t has its content as the collated contents of all the work items it merged.

Group: $WI_t \rightarrow WI_t^2$. When the group function is applied on WI_t, it will necessarily produce a partition of WI_t. Each subset of WI_t is called a *group* iff it has more than one element (i.e. one work item). When grouping work items of a task t, no new work item is created and no item is removed, which are different from the case of merging work items.

In this work, we limit our research only to grouping of work items due to the fact that merging of work items may introduce abnormalities in the execution of workflow process models, which are illustrated by the examples in Figure 4 and Figure 5 using the BPMN specification.

Figure 4 illustrates a typical result of executing a part of a workflow, which consists of four activities with their corresponding work items (T1.1, T1.2, T2.1, T2.2, T3.1, T4.1, T4.2) and an XOR-Split coordinator (with decision condition "Total > 50").

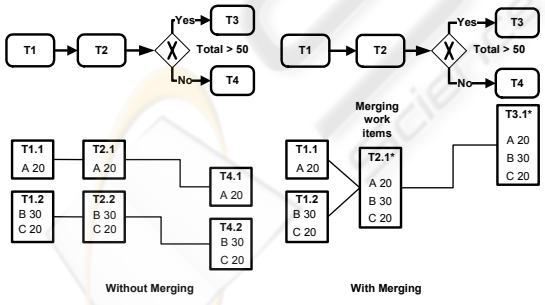

Figure 4: Merging with XOR-Split.

In Figure 4, in the case that there is no merging, after executions of task T2, task T4 is selected to execute as the result of evaluating the "Total" value of the contents of T2.1 and T2.2 (i.e. 20 and 50 respectively). However, a different result happened when merging is performed at T2. In this case, instead of T4, activity T3 is selected to execute as a result of evaluating the "Total" value (i.e. 20+30+20=70) of the merged work item T2.1.

Figure 5 illustrates a result of executing a part of a workflow, which consists of five activities with their corresponding work items and an AND-Join coordinator in two situations, i.e. when there is no merging and when there is merging at task T2.

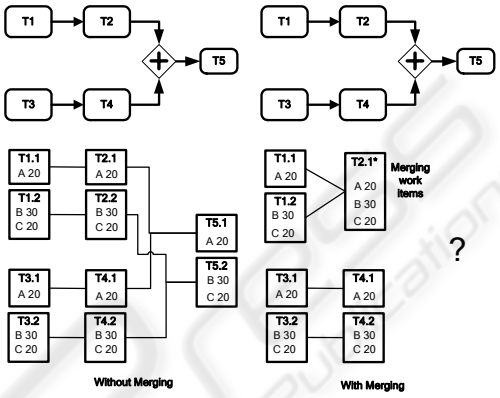

Figure 5: Merging with AND-Join.

In Figure 5, work item T5.1 is created as a result of completion both T2.1 and T4.1. Similarly, the completion of both T2.2 and T4.2 triggers the execution of T5.2.

However, in the case of merging work items (the right hand side of Figure 5), after the completion of T2.1* and T4.1, it is not desired to execute T5, since there is a non-correspondence between contents of T2.1* and T4.1 (similarly for T2.1* and T4.2). For clarity, if T2, T4, T5 refer to the tasks of Receive Purchase Order, Receive Payment, Send Goods respectively, then T5 should not be executed when there is a mismatch between the contents of a purchase order and the payment.

The main reason that contributes to the above anomalies is the violation of the coupled relationship between activity instance and process instance. Originally, each activity will have at most one execution (i.e. one activity instance) within any given process instance. This relationship is a critical foundation for workflow specifications. For instance, workflow constructs are used to route the execution of activities only within the scope of a process instance.

However, in the case of merging, this relationship does not hold. When merging, a new activity instance is created in replace of specific activity instances that originally belong to multiple process instances, thus the newly created activity instance now belongs to multiple process instances. This requires the modification in workflow

specifications to reflect the change in semantics and to make process work properly in a cross process instance environment. Such modification leads to major extension to current WFMS.

3.2 Grouping Activity Instances

According to section 2, a work item and its corresponding activity instance are two perspectives of an activity execution. We propose to use the concept of ***Instance group***, which is necessary to represent a group of selected activity instances at a point in time during workflow execution and its behaviour is modelled as a FSM (called ***Instance Group FSM*** or ***iGFSM***).

For simplicity, we also propose that the ungrouping of an instance group takes place automatically when the instance group reaches Completed state, Terminated state, or when the group is empty (i.e. after removing all work items).

The relationships between instance groups, activity instances, and process instances are as followed.

- Each activity instance can only belong to an instance group and each instance group can consist multiple instances of the same activity.
- Each process instance can have multiple instance groups and each instance group can belong to multiple process instances

When grouping work items (i.e. corresponding to grouping activity instances), certain design considerations or constraints on grouping function must be taken into account. We identified the following grouping constraints on group function.

- **Pre-grouping constraints**. Pre-grouping constraints specify whether activity instances must be in the same state or not, and the allowed states for activity instances to be able to group.
- **State synchronising constraints**. State synchronising constraints specify whether activity instances and its belonging instance group must have the same state or not at any point of execution time.
- **Membership constraints**. Membership constraints specify whether members of an instance group can be changed during instance group execution.

Different levels of flexibility on group function are determined by grouping constraints. The full flexibility of group function allows activity instances to be grouped at any time, work items can be executed as individuals even after grouping, inserting and removing work items from a group are allowed. At the other end, the flexibility of the function is lost when activity instances are required to have the same state before grouping; states of activity instances and instance group must be the same at any time of execution thus not allowing work items to be executed individually within a group; items can not inserted/removed into/from an existing group.

Grouping constraints can be specified at the activity level or instance group level, which determines the behaviour of group function for all the instance groups of an activity or for specific instance groups. For instance, users can freely specify that for the task Create Purchase Order all the purchase orders within a group must be completed at the same time, while for the task Create Invoice, each invoice must be completed individually.

3.2.1 Behaviour of an Instance Group

The behaviour of an instance group is modelled as an iGFSM. State of an instance group at a given time is identified by the states of the activity instances constitute the group.

We define the state of an instance group as followed. If:

- I_t denotes an instance group of task t
- $i_t \in I_t$ denotes an activity instance of a task belonging to the instance group I_t
- $State(i_t) \in$ {Scheduled, Active, Suspended, Completed, Terminated} is a function that returns the state of an activity instance i_t

Then state of I_t: $State(I_t) = \prod_{i_t \in I_t} State(i_t)$ (i.e. the Cartesian product of states of all the activity instances i_t belonging to the instance group I_t).

Table 1: An example for the definitions of states of instance group I_t .

$State(I_t)$	Conditions
Scheduled	$\forall\ i_t \in I_t,\ State(i_t) = Scheduled$
Active	$\exists\ i_t \in I_t,\ State(i_t) = Active$
Suspended	$\neg (\exists\ i_t \in I_t,\ State(i_t) = Active)$ $\wedge\ (State(I_t) \neq Terminated)$ $\wedge\ (State(I_t) \neq Completed)$
Terminated	$\forall\ i_t \in I_t,\ State(i_t) = Terminated$
Completed	$\neg (\forall\ i_t \in I_t,\ State(i_t) = Terminated)$ $\wedge$ $(\forall\ i_t \in I_t,$ $(State(I_t) = Terminated) \vee$ $(State(I_t) = Completed))$

Conventionally, we can define iGFSM of an instance group following the FSM of an activity instance (section 2). Depending on which grouping constraints that designers would like to enforce on the group function, different ways to define the states of iGFSM can be proposed. Table 1 represents

definitions on iGFSM states, which give the group function the full flexibility.

In general, when grouping activity instances, WFMS does not need to create an iGFSM for the corresponding instance group, since its states can be derived by the states of activity instances constituting the group. However, in practice certain benefits can be gained from extending WFMS to generate iGFSM. For instance, in validation the transitions between states (e.g. Active to Completed, Active to Suspended, but not Scheduled to Completed) of an iGFSM, the implementation of iGFSM reduces the repeating interactions within the WFMS to check states of individual activity instances, especially when the number of activity instances in an instance group is large.

3.2.2 Impact of Instance Group

We investigate the impact of instance group on the semantics of workflow specifications and the execution of a workflow.

Workflow specifications. The concept of instance group does not make any change to or affect the semantics of activity instance within a process. Thus the strict relationship between activity instance and process instance is still held, which does not require any modification to the workflow specifications.

Workflow execution. When a performer selects work items in a worklist, it leads to the creation of an instance group that groups the corresponding activity instances together. The instance group can allow each individual activity instance to be executed independently and when it completes, the control thread is released to trigger the next activity. However, instance group can also allow activity instances within the same group to be executed as a whole, when the instance group is completed, all the control threads of activity instances in the group are forced to be released at once to trigger multiple executions of the next activity (or activities).

4 DEPLOYMENT OF INSTANCE GROUPING

The implementation of instance grouping requires some modification on the WFMS. Since we aim to enhance the WFMS without complete software re-engineering, we are proposing the modification to be carried out on the client side of the software instead of the server side. i.e. no change to the workflow engine. The implementation of the WFMS can be described in two parts: modification on the Worklist Handler user interface and controlling the FSMs of the grouped work items.

Worklist Handler user interface. Traditionally, when a work item is scheduled to the performer, he/she may activate the task by pressing the "commence" button. Once the performer finishes working on the task, he/she may press the "complete" button on the worklist to complete the task and begin the next task. Although a performer is allowed to open multiple work items concurrently, but only 1 work item may be activated or committed at once. Therefore, the performer is working in a very inefficient working pattern by repeating the process such as "Press the Commence button" → "Work on the task" → "Press the Complete button" over and over again.

Our proposed instance grouping approach is introduced to save the hassle of multiple interactions between the user and system. The first modification to the worklist handler user interface is very simple but yet very useful, simply by adding a "Group" button and an "Ungroup button" to the worklist user interface. The purpose of the "Group" button is to allow work items to change their states at the same time, whereas the "Ungroup" button allows the performer cancelling the grouping anytime.

The interface of the worklist handler should be altered to distinguish the grouped work items from those not in a group. In addition to, other useful features for performers can be provided, such as automatic grouping work items based on certain conditions or sorting work items in different orders based on different properties. Figure 6 shows a sample screenshot of the proposed worklist.

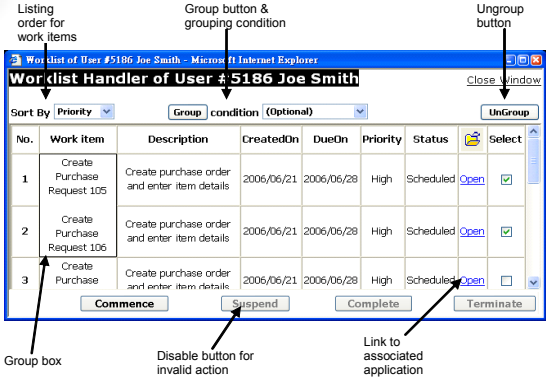

Figure 6: A sample screenshot of worklist.

FSMs of the grouped work items. WFMS needs to accommodate the modified user interface. In particular, the state of the group of work items is now managed by the worklist handler.

Once work items are in a group, the state changes of individual work item and instance group (iGFSM) must be validated by checking all grouping

constraints (section 3.2). There are four types of grouping control where validation of the group constraint may take place.

- Initialization of a group
- Insertion to a existing group
- Removal from a group
- State changes as a group

The validation of grouping constraint first takes place at the initialization stage when a group of work items are proposed to be grouped by the user. Generally, some grouping may result the inconsistence of the process. For example, work items at the Suspended state can not be grouped with work items of another state. The validation of such inadequate behaviour should be automatically checked by the worklist handler to maintain process quality.

The grouping can be enforced by the membership constraint with some specific condition. For example, if size of the group is limited to between five and ten work items, then the insertion or removal of work items into or from a group needs to be checked.

Finally, some grouping constraint can be used to synchronize the state changes of all work items of a group. The worklist handler virtually synchronises the FSMs for each work item into a virtual FSM (i.e. iGFSM). The work list handler keeps track of all work item status and the virtual FSM only changes state when all grouping constraints are satisfied.

As we demonstrated above, the implementation of the concept of instance grouping only requires the extension at the worklist handler therefore minimum modification efforts are required to incorporate this new feature.

5 CONCLUSIONS

In this paper, we propose a practically driven extension to the functionality of typical workflow management systems by offering, if required, the grouping work items facility. We identified the distinction between merging and grouping of work items, and showed that traditional workflow management systems can not deliver using the merging function without major modifications into the semantics of workflow specifications language and the workflow engine itself. We propose a more restrictive version of activity instance merging limited to the grouping of instance to provide this overhead relaxation without any modifications to the semantics of the specification language. However, it is still not for free entirely; there is a need for modification of worklist handler to support the proposed workflow flexibility extension. The paper concludes with an overview of potential implementation of introduced new workflow systems' functionality. The consideration of extending workflow specification to incorporate merging of items will form our future research.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the comments and suggestions provided by Dr. Shazia Sadiq.

REFERENCES

Aalst, W. M. P. 1999, 'Generic Workflow Models: How to Handle Dynamic Change and Capture Management Information?' in *Proceedings of the Fourth International Conference on Cooperative Information Systems (CoopIS'99)*, IEEE Computer Society Press, Los Alamitos, CA, pp. 115-126.

Aalst, W. M. P., Hofstede, A. H. M. , Kiepuszewski, B & Barros, A. P. 2003, 'Workflow patterns', *Distributed and Parallel Databases,* vol. 14, no. 3, pp. 5-51.

OMG 2004, *Business Process Modelling Notation (BPMN) version 1.0*, Object Management Group (OMG), San Mateo, CA. Retrieved: June, from http://www.bpmn.org/.

Sadiq, S. 1999, 'Workflows in Dynamic Environments – Can they be managed ?' in *Proceedings of The Second International Symposium on Cooperative Database Systems for Advanced Applications (CODAS99)*, Woollongong, Australia.

Sadiq, S., Orlowska, M. & Sadiq, W. 2005, 'Specification and validation of process constraints for flexible workflows', *Information Systems,* vol. 33, no. 5, pp. 349-378.

Sadiq, S., Orlowska, M., Sadiq, W. & Schulz, K. 2005, 'When workflows will not deliver - The case of contradicting work practice', in *the 8th International Conference on Business Information Systems (BIS 2005)*, Poznan, Poland.

WfMC 1999, *Workflow Management Coalition Terminology & Glossary*, The Workflow Management Coalition (WfMC). Retrieved: June, from http://www.wfmc.org/standards/docs/TC-1011_term_glossary_v3.pdf.