International Journal of Grid and Distributed Computing
Vol. 9, No. 3 (2016), pp.99-108
http://dx.doi.org/10.14257/ijgdc.2016.9.3.12

A Distributed Workflow Mapping Method Oriented Homogeneous Activity*

Fei Yang[1] and Yinzhou Zhu[2]

[1]*Beijing Institute Petrochemical Technology, China*
[2]*Beihang University, China*
yangfei-wu@sohu.com

Abstract

Workflow model based-on homogeneous activity needs to be transformed into basic workflow model in order to run in distributed environment. A workflow model mapping algorithm oriented homogeneous activity is proposed which can generate independent basic workflow fragments for homogeneous activities of different configuration modes using different processing mechanisms, and then form the basic workflow model based on the assembly of these workflow fragments. The algorithm provides a complete set of virtual mapping mechanism to solve it effectively focusing on the problem of the topological relationship of redundant activities difficult to describe in the dynamic selection model of homogeneous activity.

Keywords: *Workflow, Homogeneous Activity, Serial Mode, Parallel Mode, Mapping Algorithm*

1. Introduction

Homogeneous activity pattern refers to a workflow model that can represent g group of redundant activities through a special activity node, which is called homogeneous activity. Homogeneous activity pattern is divided into static structure model and dynamic selection model. The static structure model mainly describes the topological relations of redundant nodes in homogeneous activity, and the dynamic selection mode is mainly to describe the way to select execution nodes dynamically in workflow instance running stage[1].

The topological relationships of redundant activities in static structure model are fixed, because the precursor number of the converge node can be determined in workflow definition phase. It is relatively simple to describe the mapping relationships of the model. However, the topological relationships of redundant activities in dynamic selection model are generated according to the actual running situation of the workflow instance dynamically and the precursor number of the converge node is often uncertain in workflow definition phase which makes it difficult to describe the starting conditions of the converge node. Especially in distributed workflow environments the distributed activity nodes adopt static deployment way, which makes it more difficult to change the starting conditions of the converge node according to the actual situation of workflow instance. Therefore, a mapping mechanism is needed to realize the simulation results of dynamic selection model on static deployment of distributed activity nodes.

* Supported by the Project of Excellent Talents Cultivation in Beijing(2013D005005000003)

ISSN: 2005-4262 IJGDC

2. Related Work

At present, workflow scheduling and control mechanism to support the topological structure of homogeneous activity has attracted much attention and interest of researchers and a series of studies have been carried out[2,3]. A process mapping method based on Petri net element was proposed which solved the problem of the existing method only mapping tasks of workflow but ignoring the topological structure of tasks through bilateral and unilateral mapping strategies[4]. A multi instance workflow task scheduling algorithm was proposed based on ant colony optimization algorithm in order to achieve the objectives of minimum activity instances' total dwelling time and minimum activity instances' total cost with constraints[5]. PTS++ algorithm was proposed to construct a complete set of trigger sequence in the Petri net to realize the mapping of activity triggers sequence to solve the problem of difficult to distinguish loop structure and parallel structure in workflow model[6]. An elastic management cluster approach was provided to design a distributed workflow management system that run according to a set of protocols which utilized a content-based publish/subscribe messaging substrate[7]. Workflow modeling and execution method of batch process were studied in order to improve the processing efficiency[8]. A method was proposed for dynamically grouping and allocating the appropriate number of resources according to the similarity among instances to solve the problem of how to reduced the execution time of process instances in the case of resource constrained[9].

Some scholars have also carried out some research work on the common relations among multiple workflow instances. A new method for time constrained dynamic verification of concurrent workflow was proposed by analyzing the temporal relations and resource constraints of activities in view of the potential resource characteristics of multiple parallel execution workflow[10]. According to the context of workflow activities, a unified formal description multi instance activity properties and control shell of activity were presented to control the multiple instance of activities[11]. A new solution of decomposition, synchronization and coordination of master and slave instance based on pet Petri net was studied to achieve the concurrent and shared processing of workflow instances[12].

The above researches focus on process semantics such as the execution order of control constraint relations among activities which is a static level constraint description. But there is no attention to process semantics about the connection or constraint relationship of dynamic vertical level of multiple instances.

3. Mapping of Homogeneous Activity

The mapping of homogeneous activity is designed to provide an effective method which can transform homogeneous activity model into basic workflow model running in a fully distributed workflow environment and make the dynamic and vertical constraint relation of multi instances to be handled effective.

3.1. Analysis to the Mapping of Homogeneous Activity

The homogeneous activity pattern is described in static structure mode, and static structure mode is divided into serial mode (including sequence down mode, sequence up mode and privilege mode) and parallel mode (including multiple merge mode and synchronous merge mode). Dynamic selection mode is divided into auto selection, manual selection (including single selection and multiple selection), association selection[13].

In the case of sequence down, sequence up and privilege of serial mode, the number of redundant activity node and the precursor of the converge node are

unique and can be determined in workflow definition phase no matter what kind of dynamic selection mode is used.

The converge node can start a task and pass it to its successor in multiple selection of parallel mode no matter what dynamic selection mode is used because the relationship between the converge node and its precursor is one to one which can be defined and described in workflow definition phase. While in synchronous merge of parallel mode, the number of precursor of the converge node is uncertain in workflow definition phase so it is difficult to describe. The most difficult to describe is manual multiple selection synchronous merge mode in the various modes of dynamic selection, as shown in Figure 1.

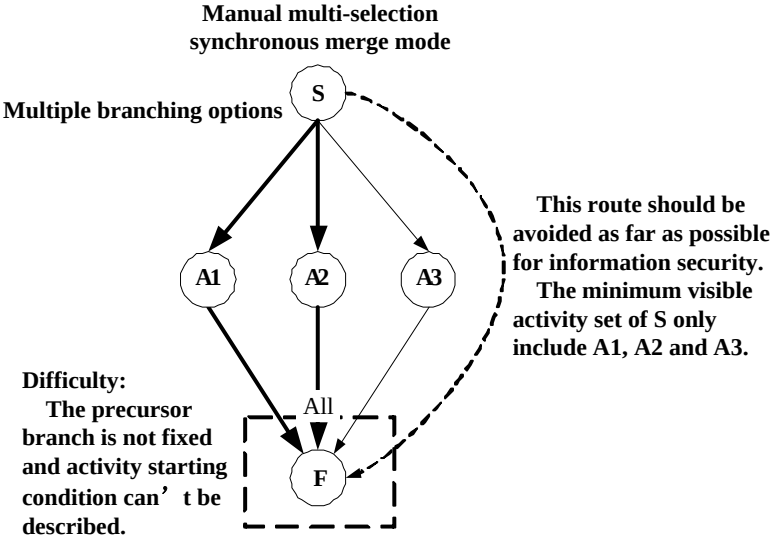

Figure. 1 Mapping Description of Manual Multi-Selection Synchronous Merge Mode

3.2. Mapping of Manual Multiple Selection Synchronous Merge Mode

Dynamic Structure mode of homogeneous activity usually relies on its precursor because it determines which redundant activities can be executed. Dynamic Structure mode includes auto selection mode, manual selection mode, association selection mode and association avoidance mode.

An interactive way needs to be provided users to set the successor of homogeneous activity after the precursor task completed and before transferred to its successor for manual multi-selection synchronous merge mode. The topological relationships of homogeneous activity can be determined only after the successor node is set through the above way according which the mapping process can be carried out.

The topological relationships of activity nodes include not only the simple precursor and successor relationships but also the mapping of data. The business processor of the precursor must not only have the authority to describe the local process but also understand its business logical if the work is left to the precursor of homogeneous activity to be finished in workflow running phase which is contrary to the original intention of workflow model. A complete set of virtual mapping mechanism is proposed to describe all redundant activities in workflow definition phase then filter the precursor relationships of homogenous activity according to actual situation on the basis of which to process task in workflow running phase.

Business logic of homogeneous activity and all possible precursor and successor relationships are described in workflow definition phase and the relationships are filtered after the topological relationships are determined in workflow running

phase. The mapping process of manual multi-selection synchronous merge mode includes the following four steps.

(1) Configure all branches of homogeneous activity as unconditional execution. Configure the precondition of task execution and join the judgment of users selected by the precursor node. Configure the starting condition of the convergence node as all precursor instances of homogeneous activity arrived.
(2) Provide the interaction mode for user to set successor nodes dynamically on the precursor node of homogeneous activity.
(3) Determine whether the current user is the user that the precursor node selected when workflow instance execution on homogeneous activity. If it is true, the task will be executed in accordance with the activity configuration, otherwise skipped.
(4) Converged data of homogeneous activity will be merged on the convergence node.

4. Workflow Mapping Algorithm for Homogeneous Activity

The mapping goal is to map the workflow model oriented homogeneous activity to basic workflow model and enable it deploy to the activity node of distributed workflow.

A workflow model mapping algorithm based on homogeneous activity *MapSWFtoWF* is proposed according to the goal which based on the workflow model oriented homogeneous activity[1,13] and can handle the case of homogeneous activity for serial mode, parallel mode and sub-flow mode. If homogeneous activity is configured as serial mode, a serial workflow fragment is generated by the serial mode mapping algorithm. If homogeneous activity is configured as parallel mode, a parallel workflow fragment consisting of redundant activities is generated by the parallel mode mapping algorithm. If homogeneous activity is configured by a sub-flow mode then the sub workflow is mapped. Finally the generated workflow fragment will replace the homogeneous activity and form a basic workflow model.

Algorithm 1 Mapping algorithm of workflow model oriented homogeneous activity *MapSWFtoWF*

Input: the current workflow management *thisUnit*,
homogeneous activity workflow model *swf*, organization *ORG*

Output: basic workflow model *wf*

//Initialize basic workflow model *wf* using homogeneous activity workflow model *swf*

$wf \leftarrow Copy(swf)$;

// Calculate the mapping of every homogeneous activity in *wf*

For (all $san \in wf.SAN$) do

// Obtain the privileged executives collection of homogenous activity

$suserList \leftarrow GetUserList(san.SRole, thisUnit)$;

// Obtain the executive collection of homogeneous activity

$userList \leftarrow GetUserList(san.Role, thisUnit) - suserList$;

If ($san.smode \in \{SequenceDown, SequenceUP\}$) // Serial mode mapping

$graph \leftarrow CreateSequenceGraph(san, userList, suserListt)$;

ElseIf ($san.smode \in \{ParallelAny, ParallelAll\}$) // Parallel mode mapping

$graph \leftarrow CreateParallelGraph(san, userList)$;


```
            EndIf
        // Map sub workflow for each homogeneous activity
        If (HasSubSWF(san.))
            For (all user ∈ userList) do
                MapSWFtoWF(user, san.subswf, ORG);
            For (all suser ∈ suserList) do
                MapSWFtoWF(suser, san.subswf, ORG);
        EndIf
    Replace(wf, san, graph);        // Replace san in wf with graph
Endfor
```

Algorithm 2 Mapping algorithm of serial mode *CreateSequenceGraph*

Input： homogeneous activity *san*, executives collection *userList*, privileged executives collection *suserList*

Output: basic workflow model fragment with serial structure *graph*

```
SortUserListByLevel(userList, san.smode);
startAn ← Pre(san);
endAn ← Succ(san);
// Obtain parallel privileged activity node graph if there is privileged user
If (suserList ≠ φ)
    graph ← CreateParallelGraph(san, suserList);
Else
    graph ← CreateEmptyGraph();
EndIf
// If dynamic selection mode and serial mode coexist
If (san.dmode ≠ AutoAll)
// Create a process variable to record the selected users
    graph.WD ← graph.WD ∪ {userListVar}
    SetSelectionTask(startAn, userList ∪ suserList, san.dmode
                , san.IN, san.EX, {userListVar}); // Set selection program
EndIf
lastAn ← startAn;        // Initialize the current activity node
For (all user ∈ userList) do    //Obtain serial activity node graph

an. ← CreateActivity(san.TList, san.MList, san.PD, san.LD, san.LED, san.AD)
    //Modify the task start condition
    ModifyTaskStartCondition(an.TList, {userListVar});
    SetSequenceCollectData(an, san.CD); //Set data used in serial mode
    link ← CreateLink(an, lastAn); //Add activity node to serial structure
graph
    graph.AN ← graph.AN ∪ an;
    graph.FC ← graph.FC ∪ link;
    lastAn ← an;
EndFor
link ← CreateLink(lastAn, endAn);
graph.FC ← graph.FC ∪ link;
```

Return $graph$;

Algorithm 3 Mapping algorithm of parallel mode *CreateParallelGraph*

Input： homogeneous activity san, executives collection $userList$

Output： basic workflow model fragment with parallel structure $graph$

$startAn \leftarrow Pre(san)$;
$endAn \leftarrow Succ(san)$;
$graph \leftarrow CreateEmptyGraph()$;
// If dynamic selection mode and parallel mode coexist
If ($san.dmode \neq AutoAll$)
// Create a process variable to record the selected users
$graph.WD \leftarrow graph.WD \cup \{userListVar\}$
$SetSelectionTask(startAn, userList, san.dmode, san.IN, san.EX, \{userListVar\})$; // Set selection program
EndIf
For (all $user \in userList$) do // Obtain parallel activity node graph

$an. \leftarrow CreateActivity(san.TList, san.MList, san.PD, san.LD, san.LED, san.AD)$
// Not set transfer condition if multi-selection mode
If ($san.dmode \in (ManualMultiple \cup AutoAll)$)
$preLink \leftarrow CreateLink(startAn, an)$;
Else //Set transfer condition if single selection mode
$preLink \leftarrow CreateLink(startAn, an, an.id \in userListVar)$;
EndIf
//Add activity node to serial structure graph
$graph.AN \leftarrow graph.AN \cup an$;
$succLink \leftarrow CreateLink(an, endAn)$;
$graph.FC \leftarrow graph.FC \cup preLink \cup siccLink$
EndFor
If ($san.dmode \in (ManualMultiple \cup AutoAll)$)
// Modify the task start condition
$ModifyTaskStartCondition(an.TList, \{userListVar\})$;
$SetParallelCollectData(an, san.CD)$; // Set data used in parallel mode
// Set the starting conditions of the convergence activity as synchronous merge
$SetStartCondition(endAn, ALL)$;
Else
$SetStartCondition(endAn, ANY)$;
EndIf
Return $graph$

The time complexity of the algorithm *MapSWFtoWF* (include the time to call algorithm of *CreateSequenceGraph* and *CreateParallelGraph*) is O(n) in which n represents the number of activity nodes in the final generated basic workflow model wf.

5. Case Study

The mapping process of manual multi-selection synchronous merge mode is illustrated by the example. Manual multi-selection synchronous merge mode is shown in the left of Figure 2 which contains a homogeneous activity A with redundancy of 3, its precursor node S and it successor node F. Two activity nodes are selected to execute task from the three redundant activities.

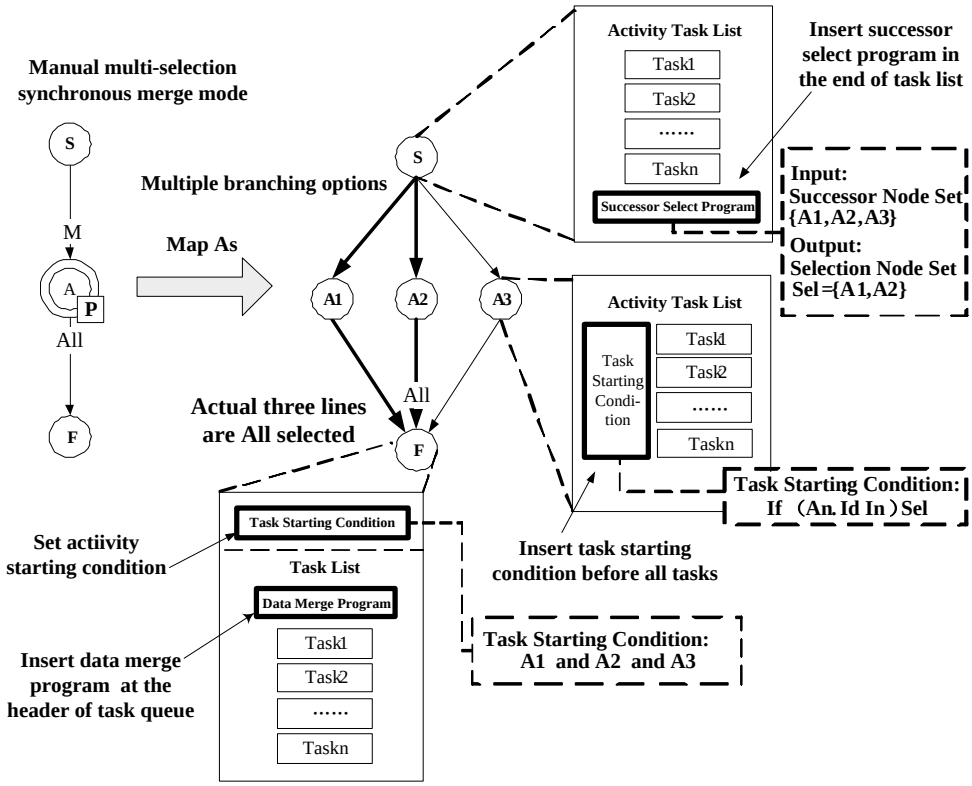

Figure 22. Mapping Result of Dynamic Selection Mode

Respectively processing are needed in both workflow definition phase and workflow running phase in the mapping of manual multi-selection synchronous merge mode.

(1) Workflow definition phase

a. Set the successor of activity node S as A1, A2 and A3.

b. Set activity attributes of A1, A2 and A3, and set the switch variables A1.isSelectedUser=false, A2.isSelectedUser=false, A3.isSelectedUser=false as the preconditions of task execution to indicate whether the current user is the selected user by the precursor. Activity node A1, A2 and A3 have the same configuration since they do the same work.

c. Set the starting condition of converge node F as A1, A2 and A3 all arrived.

(2) Workflow running phase

a. After the completion of task T in activity node S, a user selection program is inserted into the end of the task list, allowing users to dynamically set the successor as A1 and A2. At this time the switch variable T.isSelectedUser on activity node A1 and A2 are set to true and the switch variable T.isSelectedUser on activity node A3 is set to false. The above configure is delivered to successor A1, A2 and A3 when task T is transferred.

(3) (b) After the transferred task T from the precursor node S are received in activity node A1, A2 and A3, the switch variable in activity node A1 and A2 are reset to open state while the variable on activity node A3 is still in shutdown state. The variable in open state means the starting condition of the task is true and it can be executed, otherwise the task can't be executed. So the task in activity node A1 and A2 will be executed while it on activity node A3 will be skipped and be directly transferred to the successor node F.

a. The task in converge node F will be started after the tasks from A1, A2 and A3 all arrived which realizes the merging of data from the above nodes.

(4) The uncertainty of dynamical selection mode is transferred from model description level to instance implementation level so as to not affected by the static deployment of the distributed workflow architecture which disadvantage is additional workflow instances is added to the traffic flow. However, the traffic flow in the case is usually negligible compared to the traffic flow of business application data.

6. Conclusions

A workflow model mapping algorithm oriented homogeneous activity is proposed which supports the mapping process of homogeneous activity model with serial mode, parallel mode and sub-flow mode to basic workflow model. The mapping mechanism transfer the uncertainty of dynamical selection mode from model description level to instance implementation level, that is to say the uncertainty problem of topological relationships of redundant activities in workflow definition phase is postponed to workflow running phase and task can be flexibly processed according to the actual situation, thus to realize the simulation results of dynamic selection model on static deployment of distributed activity nodes. Experiments show that the method is effective.

References

[1] Y. Z. Zhu, H. Yang and B. L. Yin. Computer Modeling and New Technologies. 18, 1 (**2014**).
[2] J. X. Liu and J. M. Hu, Future Generation Computer Systems. Vol. 23, no. 3 **(2007)**.
[3] J. X. Liu, Y. P. Wen, T. Li and X. Y. Zhang, Concurrency and Computation: Practice and Experience. 17, 8 **(2011)**.
[4] B. Cao, J. X. Wang, J. Fan and T. Y. Dong, Journal of Software, vol. 26, no. 3 **(2015)**.
[5] Y. P. Wen, J. X. Liu and Z. G. Chen, Journal of Software. Vol. 26, no. 3 **(2015)**.
[6] Y. Yu , Y . Wang, X. M. Liu and J. Chen, Journal of Software. Vol. 26, no. 3 **(2015)**.
[7] S. Ganesan, Y. Yoon and H. Jacobsen. NIÑOS take five: the management infrastructure for distributed event-driven workflows. Proceedings of the Fifth ACM International Conference on Distributed Event-Based Systems, **(2011)** July 11-15; New York, USA.
[8] L. Pufahl and M. Weske, Batch activities in process modeling and execution. Proceedings of the 11th International Conference on Service Oriented Computing, **(2013)** December 2-5; Berlin, Germany.
[9] J. Pflug and S. Rinderle-Ma, Dynamic instance queuing in process-aware information systems. Proceedings of the 28th An Annual ACM Symposium on Applied Computing, **(2013)** March 18-22; Coimbra, Portugal.
[10] H. C. Li and Y. Yang, Electronic Commerce and Application. Vol 4, no 2 **(2005)**.
[11] R. Z. Sun and M. L. Shi, Journal of Software.vol. 3, no 16 **(2005)**.
[12] H. H. Lu, L. J. Min and Y. S. Wang, Journal on Communications. Vol. 31, no. 1 **(2010)**.
[13] F. Yang, Y. Z. Zhu and Y. Luo, International Journal of Control and Automation. Vol. 8, no. 10 **(2015)**.

Authors

F. Yang. She received her PhD in Computer Sciences and Engineering Department(2011) from Beihang University. Now she is full lecturer at Computer Department, Beijing Institute Petrochemical Technology. Her current research interests include workflow management, enterprise application integration and distributed system.

Y. Z. Zhu. Ph.D. candidate in Computer Sciences and Engineering Department at Beihang University.