An Analysis of Delegation Mechanism in Workflow Management System

Jian-Wei Wang, Chi-Hsieh Chang, and Feng-Jian Wang
Department of Computer Science and Information Engineering
National Chiao Tung University, Hsinchu, Taiwan, ROC
Telephone: (03)5712121 x 54718 Fax: (03)5724176
Email: {jwwang,chschang,fjwang}@csie.nctu.edu.tw

Abstract

Despite many research works in the features of Workflow Management System (WFMS), several issues are not researched yet. One of them is the delegation mechanism. Delegation mechanism ensures the continuity of processes. This paper presents an analysis of delegation mechanism in a Workflow Management System. Two scenarios in which delegations happen and several delegation rules are shown as the detail of delegation mechanism. Then several problems of delegation are discussed based on the delegations and delegation rules. In order to find out the problems with delegation mechanism, a method is discussed to transform the delegations with delegation rules into a graphic view. With the graphic method, algorithms of finding corresponding delegation problems are presented.

Keywords: workflow, delegation mechanism, delegation, delegation rules, delegation graph

1. Introduction

In recent years, many issues in Workflow Management System (WFMS) have been studied and implemented to commercial products. Delegation is one of the security mechanisms in Workflow Management System. This mechanism can ensure that a process session to be done in the appropriate time. Even one of the process members is unavailable during a short time. One can temporally assign the job to another member in the company. The process will be done on time and the company operates normally, continuously.

Our research topic is study the phenomena of delegation in a Workflow Management System. Categorize the detailed delegation rules, delegation types and scenarios of different delegations. Find possible problems according to the mixed delegation rules and scenarios. Finally, a method which can solve these problems will be proposed to provide a systematic solution to administrators of Workflow Management System.

2. Background

2.1. Workflow Management System (WfMS)

Recently, Work Flow Management is a fast evolving technology which is exploited by businesses and industries [6]. The primary characteristics of Workflow Management are the automation of processes which combine the human and machine-based activities [6]. The management should define each model that is related to the workflow, the basic process definition model, role model, related application,

the interaction and data interchange with the model.

Workflow Management System, WfMS, is a system that implements all features that WFM defines. The definition of process model, role model, and other features that WFM defines will be combined as one of the WfMS's functionality. The WfMS possibly offers a GUI for users to define their own specific processes, the roles and relevant data of this process. With the detail description of process, the WfMS controls the execution and interoperation of process instances.

2.2. AgentFlow

AgentFlow [8], which is based on the concept of Process-centered Software Engineering Environment (PSEE) [2][5], is a software that contains lots of tools and lets end users to use the tools to define the enterprise's specific workflow management. AgentFlow contains the tools that correspond to all necessary components and interfaces in the workflow model. AgentFlow has five components, which includes PDE, ORG, FormDesigner, FlowEngine and Agenda [3]:

Figure 1 illustrates the overview of AgentFlow.

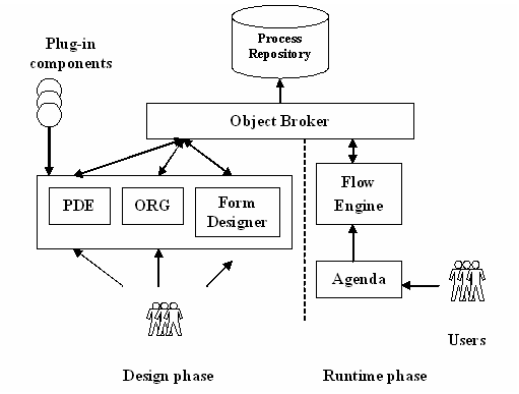


Figure 1 AgentFlow System overview

With AgentFlow, designers can easily construct specific process and define each relevant data to process activities. All process related data will be stored in the process Repository, such as Database. During the process execution, system administrator can monitor the execution of each process with administrator access right in Agenda and system execution logs [3]. End users interact with Agenda client program to do the jobs in their own work-list. With flexible characteristics of flow modification, these data can be viewed as the treasure to tune up the business processes more effectively, and achieve the BRP(Business Process Reengineering) [7][9][10].

3. Delegation phenomena

3.1. Scenarios of delegation

3.1.1. Delegation in regular processes

This scenario indicates that a delegation rule can be applied in an e-office or e-organization. One participant could own multiple roles in an organization. For example, the RD department supervisor participant owns a supervisor role obviously. This supervisor could own another role, like the engineer or the supervisor of project B; multiple roles have different delegation rules, one corresponding to each role. The delegation rules might be applied role by role in regular processes, the related problems within this scenario will be discussed in chapter 4.

3.1.2. Delegation in specific processes

In this scenario, the delegation rules can not be applied in other processes. In other words, the roles and the corresponding participants' delegation rules are established in the organizational role model during the process specification.

The life time of such a process instance is limited. So are that of relevant data and temporary new roles in the same project. The new process needs new relevant data and temporary new roles. The relevant data and temporary new roles will vanish after the

specific processes completes.

3.2. Delegation rules

1. Delegation Forbiddance

Delegation forbiddance is the first rule which is defined to prevent important process activities to be delegate to unrelated members. If a process activity is defined associated with delegation forbiddance rule alone, this activity won't allow any delegation to be activated.

If there are too many forbiddance rules set in a process, the process might lose flexibility in execution time. In delegation forbidden activities, the participant will stall the whole execution because of his unavailability. Thus, deciding the delegation forbiddance rule of activities depends on the policies of organizations.

2. Single delegation

Single delegation is one of the major rules for members in the organization to define their own delegation. There is one or more than one choice, and this most common method provides a one-to-one delegation of a process activity for the participants, especially; the target participant is the same and single for all common source members in a process definition.

3. Multi-conditional delegation

This delegation rule allows user to delegate the job to one or divide the delegation into a group of members each time. Multi-conditional delegation provides a flexible delegation rule for multiple members in the organization.

For example, member A has a multi-conditional delegation rule with a process, where such a rule may be associated with multiple values. The value could be composed of the following data: process name, activity subjects, job deliver time, the job sender, etc. During runtime of process activities, if the multi-conditional delegation mechanism is activated, the system calculates the above data to get the corresponding target participants for the delegation.

4. Multiple-conditionally general delegation

The most general specification model of delegation is to allow multiple value-conditions where a value points to its own target and the target is either a single member or a group of members.

4. Delegation Analysis

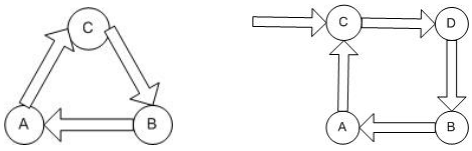
4.1. Defects in delegation phenomena with delegation in regular process

After introducing different rules and types of delegation, there are several problems associated with delegations in regular processes. To simplify the problem, here the delegation rules and delegations are assumed to be fixed under process specification for us to analyze. The analysis of dynamic delegation will also be discussed.

4.1.1. A general delegation loop

A general delegation loop will possibly happen under the rules of single delegation, multi-conditional delegation and multiple-conditionally general delegation.

In **Figure 2**(a), the delegation loop is the simplest loop. A, B, C indicates the participants in enterprise. With the delegation D(A, B) D(B, C) D(C, A), the delegation sequence will end up with a deadlock. In **Figure 2**(b), a sequence of delegation delegate to target participant D, and the delegation rules D(D, E) D(E, B) D(B, C) D(C, D) also end up in a deadlock.



(a) (b)

Figure 2 Delegation loop

4.1.2. Mutually Exclusive Roles

In RBAC [11], the most common constraints in RBAC are mutually exclusive roles. Assume that a user can be assigned only one role in the mutually exclusive role set. This constraint can assure the separation of duties and avoid the assignment fraud.

In the process specification, participants of this process may not conflict the mutually exclusive role constraint. But it may conflict the constraints after the delegation is activated. Delegation conflicts the mutually exclusive constraint are defined as two types:

1) Direct delegation conflict

In **Figure 3**(a), participant E_A, owns role R_1, delegates the jobs to another target participant E_B who owns role R_2. R_1 and R_2 are in the mutually exclusive roles set S. Participant E_A plays two mutually exclusive roles and conflicts the constraint.

2) Indirect delegation conflict

In **Figure 3**(b), Participant E_C plays two mutually exclusive roles and conflicts the constraint.

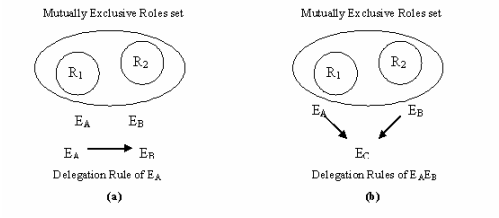


Figure 3 Mutually exclusive roles

4.1.3. Work unbalance

With the delegation mechanism, the mechanism might break the work balance. In **Figure 4**, participants E_A, E_B, and E_C are working on the project process P. If E_A and E_B assign their delegation rule to E_C, E_C shall do all the jobs in this project process. And then a work unbalance problem appears. The delegation appears always in WfMS, but the process execution might be inefficient if the plan is not good as in **Figure 4**.

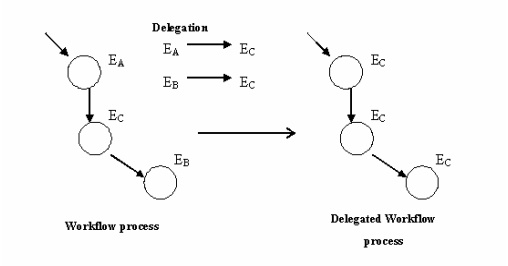


Figure 4 Work unbalance

4.1.4. Inappropriate delegation

Several delegation problems are special because the delegations in these problems are always legal. Inappropriate delegation problem is one of them. Inappropriate role delegation problem will damage the access control policy in some enterprises and leads to assignment fraud.

For example of inappropriate delegation, there are two roles, CEO and assistant engineering in an organization structure. These two roles are not mutually exclusive, but CEO should not normally delegate his or her jobs to the assistant engineer. The delegation does not make a fraud but is not allowed actually.

4.2. Solutions to problems in scenario of delegation in regular process

4.2.1. A graphic method

In previous section, a delegation sequence is defined as: Let a role S has a delegation rule D whose target is T, the D(S, T). A sequence of continuous delegations, can be defined as $D(S_1, S_2)$, $D(S_2, S_3)$, $D(S_3, S_4)$ … $D(S_i, S_{i+1})$ …, etc. To form a delegation graph from delegation sequences in a process, an algorithm to generate is defined as follows:

A delegation graph G, delegation set S in process P
P has fixed delegations and delegations rules
A delegation or a delegation sequence D in S indicates a delegation of ***source*** *->* ***target*** *or a sequence of* ***source -> target combinations***

Generate_delegation_graph(G,D,P)
Begin

For *each D in process P*
For *each existed nodes in graph G*
If source *= one of the exist nodes in G*
Then *Connect this delegation D to exist node in G*
Else *G = G U D*
End

- **Delegation loop and unbalance**

Delegation loop and work unbalance could be observed on the delegation graph. Mutually exclusive set problems and inappropriate delegation can also be observed in addition to the add-on on delegation graph.

Delegation loops and work unbalance are shown in **Figure 5**. In **Figure 5**(a), a delegation loop is found between participants A, B, C. In **Figure 5**(b), a work unbalance problem is found under participants A, B, C, D.

The algorithm of delegation loop and unbalance are shown as follows:

//the begin of algorithm with delegation loop use DFS algorithm
Check_delegation_loop_work_unbalance (G)
Begin
DFS(G)
If *any edge marked as* ***back edge***
Then *claim a delegation loop will occur*
If *any edge marked as* ***Forward edge*** *or* ***cross edge***
Then *a work unbalance will occur*
End

In above algorithm, a DFS(G) will traverse all vertices in a delegation graph. A back edge indicates a participant points to its ancestor which means a loop occur. A forward edge and cross edge indicates the participant points to a traversed participants which means the participant might have more than one delegated job to do and it means a work unbalance will happen.

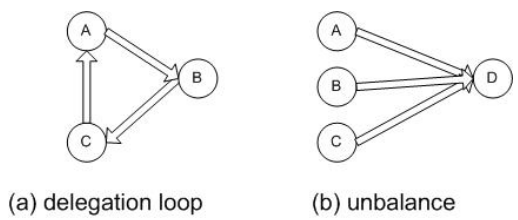


Figure 5 A delegation loop and work unbalance

- **Mutually exclusive roles**

To find out mutually exclusive role sets problem, a mutually exclusive role list has to be added in the delegation graph. Participants A and B are in the mutually exclusive role sets. In **Figure 6**(a), there is a delegation from A to B and might cause a direct conflict if this delegation activated in process execution. In **Figure 6**(b), there is an indirect delegation conflict because of A and B delegate jobs to the same participant C.

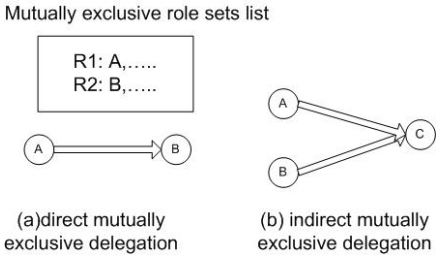


Figure 6 Mutually exclusive role sets problem

The example in **Figure 6** presents a possible direct and indirect delegation problem in mutually exclusive role sets. In **Figure 7**(c), participant A might not direct delegate jobs to target participant B but in a indirect way and same as in **Figure 7**(d), but participants A and B also delegate jobs to target participant C and conflict mutually exclusive role sets problem.

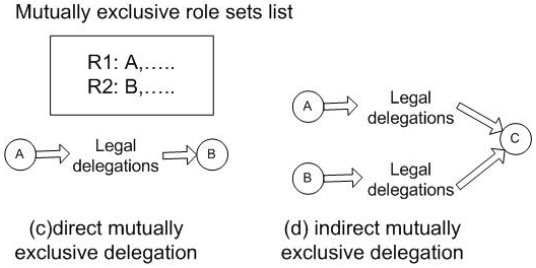


Figure 7 Another mutually exclusive role sets problem

The algorithm of checking mutually exclusive role sets are shown as follows:

Check_mutually_exclusive_problems (G)
Begin
//first check the direct conflict using DFS
For *each node in DFS progress*

Record the ancestors of current node
If *current nodes and any of traversed nodes in this path are all in the mutually exclusive role sets list*
Then *this delegation will cause mutually exclusive role sets problem*
Else *continue DFS*
//Second check the indirect conflict
For *each node's ancestors list(s)*
If *in ancestors list has two of the ancestors are in the mutually exclusive roles set*
Then *this node has an indirect conflict*
Continue until all nodes' list are checked
End

- **Inappropriate delegation**

Inappropriate delegation is defined as legal but has an essential delegation problem. To resolve the problems in delegation graph, a level constraint can be used on the participants in processes. For example, the organization can separate participants' power into 10 levels, and the delegation have to make a constraint of "cannot delegate the jobs to power level lower or higher than 3 to 5 levels. **Figure 8** illustrates the inappropriate delegation in delegation graph. Participant A has a power level of 2 and B has a power level of 7.

The algorithm is similar in delegation loop test, also shown as follows:

Check_inappropriate_delegation (G)
Begin
For *each traversed nodes in DFS*
If *this node's class level is much lower or higher than any of the nodes in traversed path*
Then *warning an inappropriate error will occur and mark this edge as* ***inappropriate edge*** *in graph G*
Else *continue DFS*
End

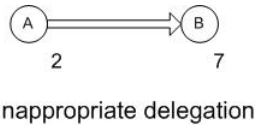


Figure 8 Inappropriate delegation

4.2.2. Using method in dynamic delegation

When this participant's defined delegation or delegation rules change, the previous static analysis in process specification time can not apply on this situation.

In **Figure 9**, there is a delegation graph which is calculated by our algorithm in process specification time and an inappropriate edge is found. But in the execution time of this process, participant D decide to change his delegation rule from single delegation to multi-conditional delegation which delegates his job to the other two participants J and K.

Therefore the previous delegation graph will be changed into **Figure 10**. Notice that the delegation of participants J and K should be included in this new changed delegation graph. Then the whole algorithms of detecting the delegation problems will be applied into this new delegation graph. There will be a back edge from participant J to C, cross edge from participant K to F, and still the inappropriate edge from H to I.

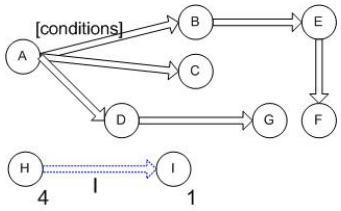


Figure 9 A previous analyzed delegation graph

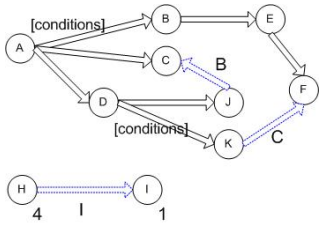


Figure 10 An analysis of changed delegation graph

4.3. Problems of delegation in scenario of a specific process

- Delegations from participants to other participants in specific roles of the process

The delegation is allowed in this specific process. Delegations will also be revoked with

the destruction of this specific process. When a specific process has ended and destroyed, the delegations of this specific process will also be destroyed. Because of the delegation's source and destination are in the specific roles. Therefore, the delegation will not cause any errors in this delegation type.

- Delegations from other source participants to target participants in this specific process

In this situation, other source participants who do not participate in the specific process may define the delegation to any of the participants in this process using the multiple-conditionally general delegation rule. Other jobs maybe delegated to specific participants by using this delegation rule. Problems will occur when this specific process has been done and destroyed.

4.4. An example

4.4.1. Using method on an example flow

Let us illustrates the example flow graph. The participants are the RD department engineers and RD department supervisor.

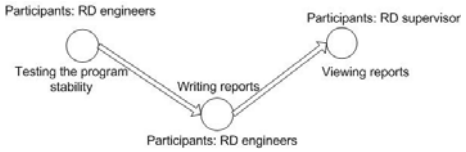


Figure 11 An example flow graph

Figure 12 illustrates role tree of the participants of this flow graph. In this figure, related roles of delegation are also listed.

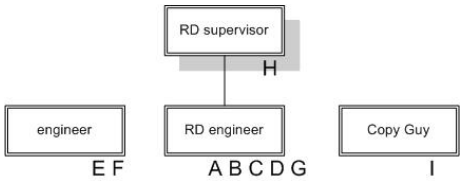


Figure 12 A role tree of the participants and related role

After the role tree and flow graph presentation, **Figure 13** illustrates the delegations of participants in this flow graph.

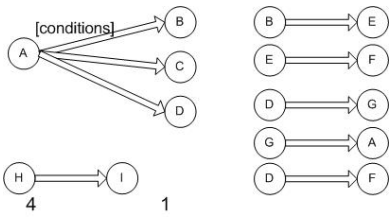


Figure 13 The delegations of participants of flow graph

After the delegations of participants are listed, a delegation graph can be easily formed by the method presented in previous section. When the delegation graph of this flow graph is formed, this graph can be analyzed in a static way to find out possible problems which are defined in previous section.

Figure 14 illustrates the delegation graph of this flow graph. In this figure, several problems are found as follows:

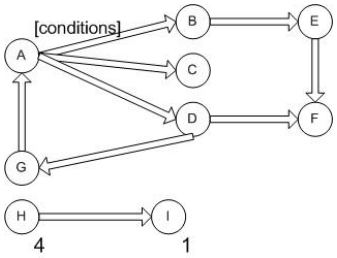


Figure 14 Delegation graph formed by delegations

After the transformation of delegations to delegation graph, we can now apply our algorithm to this delegation graph. First a DFS algorithm is applied on this graph with back edge, cross edge, forward edge, and a new inappropriate edge. The delegation graph will become as **Figure 15**. A back edge D(G, A), forward edge D(D, F) and a inappropriate edge D(H, I) are founded and a delegation loop, an inappropriate delegation problems will occur if these delegations are activated in process runtime.

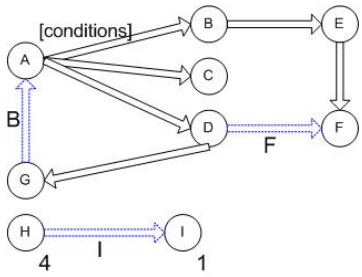


Figure 15 Algorithm applied delegation graph

5. Conclusion

In this paper, delegation, delegation phenomena, and several types of delegation rules are defined. The details of delegation rules are also presented to provide a clear view of delegation phenomena in WFMS. Then possible delegation problems are proposed and a static method is proposed to analyze the flow graph and delegation relation. This method provides a clear view of analyzing the delegation behavior in process specification time. Finally several examples are presented based on the role tree and flow graph with this graphical method.

In delegation phenomena, although possible delegation problems are listed., there are still many delegation phenomena which are not figured out or analyzed. There is still need works to deal with the delegation area in WFMS. Finding the whole problems or issues of delegation mechanism in WFMS is still a research goal in this area.

Reference

[1] [URL] http://www.wfmc.org

[2] M. F. Chen, B. S. Liang and F. J. Wang, “A Process-Centered Software Engineering Environment with Network Centric Computing”, 1997

[3] Y.S. Chen, F. J. Wang. “Building an Edit and Enactment System for Process Definition”, 2001

[4] M. F. Chen, B. S. Liang, and F. J. Wang, “Enacting a Software Development Process”, to appear in Proceeding of International Conference of Engineering Complex Computer System, 1997

[5] B. S. Liang, M. F. Chen, and F. J. Wang, “The Study of a Process-Driven Software Project Development Environment”, Technical Report CS85-0210-D009-014, Microelectronics and Information System Research Center, National Chiao-Tung University, 1996

[6] David Hollingsworth, “Workflow Management Coalition The Workflow Reference Model”.

[7] R. Marshak, “Software to Support BPR – The value of Capturing Process Definitions”, Workgroup Computing Report, Patricia Seybold Group, Vol,17, No.7, July, 1994

[8] [URL] http://www.flowring.com.tw

[9] Karagiannis, D. (1994, May). “Toward Business Process Management Systems”. Tutorial at the International Conference on Cooperative Information Systems (CoopIS’94). Toronto, May 1994.

[10] Soles, S. (1994). “Work Reengineering and Workflows: Comparative Methods”. In (White & Fischer, 1994), 70-105

[11] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, Charles E. Youman, “Role-Based Access Control Models”, *IEEE Computer*, February 1996