International Journal of Database Theory and Application
Vol.7, No.3 (2014), pp.255-264
http://dx.doi.org/10.14257/ijdta.2014.7.3.22

Research on Consistency Maintenance of Right Management Operations in Real time Collaboration Environments

Liping Gao[1], Qiongqiong Fu[1], Lili Gao[2], Yuben Zhang[1] and Qingkui Chen[1]

[1]School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai, 200093, China; [2]Pre-school and Special Education Normal School, WeiFang University, Shandong, 261021, China
lipinggao@fudan.edu.cn;qiongqiongfu@usst.edu.c;snowgao@163.com;yubenzhang @usst.edu.cn;qingkuichen@usst.edu.cn

Abstract

As for the lack of the constraint on operations in traditional real-time collaborative environment, the paper introduces to adopt right allocation technique to support the document management of the shared document of different work groups. The paper gives formal definition of the Tree Document Model supporting role management, describes the configuration process of the roles, defines the formats of the right operations, proposes conflict detection and resolution strategies for Apply/Apply, Apply/Delete and Apply/Update conflicts. Besides, the paper improves the address space transformation strategy to guarantee the correct execution of the right operations in the distributed sites. Efficiency analysis of the improved algorithm is given and the application of this strategy in the Co-AutoCAD system proves the validity of the whole strategy.

Keywords: *Real-time Collaboration; Right Management; Layered Document; Address Space Transformation*

1. Introduction

Real-time collaborative editing systems adopt a replicated architecture to build a P2P connection between the distributed sites [1-5]. And the copies of the shared document are replicated at all sites in the architecture. The operation produced on the local site are executed immediately on the local document copy, whose effect is fed back to users at once, which ensure response time of the groupware systems is comparable with that of the single-user application. Positive concurrency control techniques such as Operation Transformation (OT) [3, 5] or Address Space Transformation(AST)[4] *etc*. are adopted in real-time collaborative editing systems to guarantee the consistency of the distributed documents. The system achieves high concurrency by discarding negative concurrency control strategies such as locking, token, transaction etc. However, if the architecture was used for collaborative design environment to support the collaboration and communication among different design members, the following problems will appear:1) Since every user owns the equal editing right of the sharing documents, operational disturbance among different teams may occur frequently (*e.g.*, Pipeline designers make modification on structure design); 2)Transforming the document into linear document, and implementing concurrent control on the entire document increases the complexity of the consistency maintenance algorithm;3)unpublicized documents cannot be provided confidentiality support. That is to say, if some designers don't want to share some parts of the document, the only way he/she can do is to delete them.

ISSN: 2005-4270 IJDTA

This paper firstly proposes a tree document model to describe a document structure of CAD editing field, analyzes the necessity of right allocation in the process of collaborative design and gives definitions for right operations as well as tree document model supporting right operation. The Address Space Transformation strategy is improved to support consistency maintenance of concurrent right operations with other editing operations. Conflict definitions and resolutions of Apply/Apply, Apply/Delete and Apply/Update are detailed. Processes of the set-up of the site ids are also discussed.

The following of the paper is organized as follows. Section 2 analyses the related work and Section 3 describes the tree document model which supports right allocation. Section 4 gives definitions for right operations. Section 5 details the improved AST strategy to achieve consistency maintenance of concurrent operations while Section 6 describes the definitions and resolution strategies of Apply/Apply, Apply/Delete and Apply/Update conflicts. Section 7 concludes the paper and discusses the future work.

2. Related Work

Right management is hot spot in the field of information system, network security, etc. In recent years, domestic and foreign scholars do a lot of work in the field of the access control policy (RBAC, TBAC, *etc.*) [6, 7], supporting technology (based on Hibernate, based on SOA, etc.) [8-10], uniform structure[10], *etc*. But these studies are mostly based on C/S or B/S structure whose right allocation usually has been completed by one or more super-users, which cannot satisfy the requirements of fast responsiveness in replicated architecture who can provide access to fine-grained entity in collaborative systems.

Few researches have been done in the real-time collaborative environment as for right management. Fang *etc*.[11] puts forward a multi-level security access control strategy of CAD model for the collaborative environment. The strategy uses a model of multi-level right, simplifies the definition of right and transfer procedure, which enriches the ability of right expression and achieve multi-granularity access control of the product model. However, the work is focused on solving the right division and dynamic migration issues during downstream passing of the product models during working stream. Fu *etc.* [12] has proposed the access control strategy based on RBAC in collaborative environment. This strategy introduces the role mechanism to the complex granting issue of shared resource access control strategy and the coarse-grained granting issue. And the strategy takes the role of concurrent access control strategy applied in the system. However, the method needs to make the pre-allocated right before collaboration, whose dynamic property is poor. Jiang *etc.* [13] have proposed the flexible management strategy and dynamic allocation technology of the user' right so that the right of members are be dynamically adjust when the enterprise organization or the staff need some adjustments. Though the strategy provides dynamics, it still cannot meet the demand of the dynamical right allocation for the entity in real-time system adopting replicated architecture. Feng *etc.* [14] have put forward to use the Lock scheme combined with the AST technology to achieve the semantic maintenance of editing operations in the field of collaborative editing, with Lock ensuring that parts of document cannot be modified by other users so as to realize the right control of modifying. Though this method can provide the access control in the replicated architecture, whose functions are simple(only the right management of modification operations) and can only fit in the environment with sequential addresses of the entities and cannot fit in the CAD environment with discrete addresses of the entities in the same group. Moreover, the Lock scheme also reduces the concurrency of the real-time collaborative system.

Our research group [15] has proposed the real-time collaborative mechanism to support the team allocation in the previous work. The mechanism divides the document to mutual disjoint

task areas according to editing requests from different teams and through modifying the broadcast procedure of local operations the and receiving procedure of remote operations for the consistency maintenance of sub-documents. The strategy is better to meet the needs of allocation and cooperation among different members, however, it can be used to handle with only linear document model, and the way the document division is based on to realize cooperation cannot be suitable for the CAD environment with the feature of discrete address of the entity. Besides, we [16] have ever proposed the hierarchical structure document model supporting the right division for the features of document model in the CAD field. But the solution is just to solve the conflicts of the part of the users' operations, and the concurrency of real-time collaborative system has not been improved obviously.

3. Tree Document Model Supporting Right Allocation

3.1. Tree Document Model

The OT and AST strategies in the real-time collaborative field both format the document model into a linear structure [2-5]. This linear document model does not accord with the characteristics of the document in the design field, and may lead to high complexity of the algorithm (Sine the Insert/Delete operations can lead to the position changes of document elements from the current node). For this, according to the characteristics of CAD document model, we abstract it into the tree structure (as shown in Figure 1). Each node consists of several elements. Each element are appended with a linear history buffer storing the executed operations targeting at the element by the order of the time-stamps[2-3] of the operations. The node elements in the first layer can be marked with its positions in the linear table. The node elements in the i^{th} layer can be marked by the form of $<P_1,P_2,\ldots P_i>$ （1<=i<=5） , with P_j(1<=j<=i-1) indicating the sequence number of its ancestor on the j layer , and P_i indicates the sequence number of the node element located in the current node. Take Figure 1 as the example, P_1 indicates the sequence number of the Document node, P_2 indicates the sequence number of the Layer node, P_3 indicates the sequence number of the block node, P_4 indicates the number of the entity node, P_5 indicates the sequence number of the vertex node. The node element of V 2,1,1,2,4 indicates a node element who is the 4^{th} vertex of the 1^{th} block in the 1^{th} layer of the 2^{th} document. The true structure model of Tree-Doc is described in Definition 1.

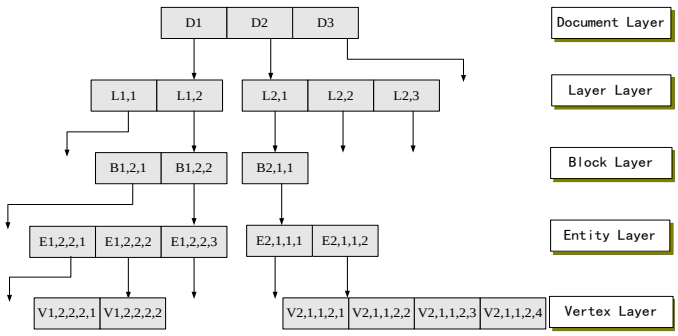

Figure 1. The Tree Document Model

Definition 1: Tree Document Model (Tree-Doc)

Tree document model (Tree-Doc) is a limited set of N nodes. It is either empty set (n=0) or consists of a root Node and k mutually disjoint sub-trees. Tree-Doc is used to represent the

unique identifier of the Node in Document layer, and Tree-Doc<P_1, P_2, …, P_i> indicates the unique identifier of the $(i+1)^{th}$ layer($1<=i<=4$), in which $P_j(1<=j<=i)$ indicates the sequence number of the ancestor of the Node on the j^{th} layer. For example, Block1,2,1 and Block1,2,2 are located in the Node whose unique identifier is Layer-Doc<1,2>.

3.2. Tree Document Model Supporting Right Allocation

Different parts of the document should be set different editing modes according to practical requirements: visible/invisible, updateable/read-only. Because the confirmed editing results are not allowed to be modified by the latter operations, their right modes are set to read-only (i.e., editors are not allowed to modify these parts of the document); in addition, products among different teams can only be allowed to refer to but not be changed mutually. Moreover, if some products are confidential and do not want to be seen by other participants, they should be set to be invisible. So, two right modes are designed: visible/invisible, updateable/read-only.

First, two fields are added to every Node Element: Visible/invisible and Updateable/read-only, which are used to represent whether the current Node is visible and Updateable (including Delete) or not respectively. The modified structure of the Document Node Element is shown in Figure 2.

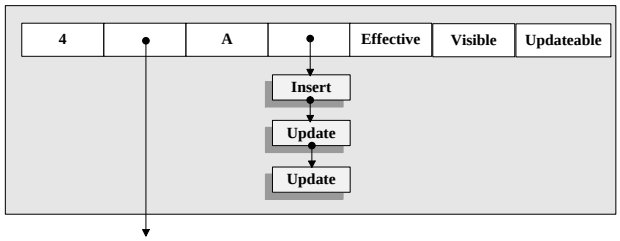

Figure 2. The Structure of Node Elements Attached with Right

4. Definition of the Right Operations

According to the previous analysis, we define two types of rights: Visible right and Updateable right. The two parameters can be True or False. The meanings of which are described as follows:Visible = True: element is visible; Visible = False: element is invisible. It is important to note that the difference between Visible field and Effective field. Visible field just indicates the visibility of entity object on the interface of the users; while Effective field expresses the validity of entity in the real document space.Updateable = True: element can be modified; Updateable = False: element cannot be modified. Updateable=false means that the object can only be readable but not modified.The right operations have 2 types: Apply and Grant. Users in the lower layer send Apply operations to users in the upper layer; users in the upper layer send Grant operations to users in the lower layer. The rights reply Operations responding to the Apply operations has 2 types: Accept and Deny. These four types are described as follows:

1) Apply([right], list[object], [role], Initiator, SV);The statement indicates the Initiator site want to apply [right] on list[object] in site [role] in SV document status. *E.g.*, Apply(Updateable=false, [<1,1>,<1,2>], <1>, 3,SV), shows that site3 requests to set the entity in the 1^{th} and 2^{th} layer of the 1^{th} document at site1 as only readable.

2) Grant ([right],list[object],[role],Initiator,SV); The statement indicates to execute the [right] operation from Initiator site to the list[object] in SV document status. The definitions

of [right], [object], [role] are the same as that in Apply. *E.g.*, Grant(Visible=False,<1,2>,<3>,2,SV) indicates that at site3 executes operation of setting invisible in the 2th layer of the 1th document at site2.

3) Accept(Initiator, Grantor, Apply);Grantor site accepts the Apply operation from Initiator site, namely, the Apply operation succeeds. *E.g.*, Accept(3,1,Apply(…)) indicates that site1 accepts the Apply operation from site3.

4) Deny(Initiator, Grantor, Apply);Grantor site denies the Apply operation from Initiator site, namely, the Apply operation fails. *E.g.*, Deny(3,1,Apply(…)) indicates that site1 denies the Apply operation from site3.

5. Intention Preservation of Grant Operations

Grant ([right], list [object], [role], Initiator,SV) should be executed in the corresponding context, namely, we should also attach the SV to Grant operation. If there exists the operations of editing document(Insert, Delete) that are concurrent with Gran operation, the list[object] parameter of the Grant operation will be not explained correctly. So, before executing Grant operation, the Retrace procedure must be called to retrace the document state to the state when the Grant operation is released.

5.1. Tree-AST Algorithm

During the Retracing process of the document with the tree structure, there is no need to retrace the statuses of all the entities but only retrace the ancestors of the NODE attached with O and itself from top to down. The aim of the Retrace procedure is to retrace the ancestor Nodes from 1 to level (NODE)-1 layer in the tree document to the timestamp SV of the current operation. And then compare SV with the timestamp of operations attached on nodes, determine whether the nodes are effective or not. This algorithm only need retrace the linear table of one node at every layer, which improves the algorithm's execution efficiency greatly. Retrace procedure hides all execution effects of the concurrent Insert and Delete operations (except Update and Grant operations, because Update and Grant cannot change the absolute position of entity element and unique identifier for the element), and only preserve the executing effect of operations causally preceding [2,5]the current operation O. For every Element E of a Node, suppose that Insert and Delete operation in the operating history list are timestamped by SV_{ins} and SV_{del}. The algorithm judges whether the node N is valid or not under timestamp SV_o [3, 4, 14] .Retracing procedure under the Tree-Doc is shown in Procedure 1.

Procedure 1 Retracing_Tree-Doc (Tree-Doc, level, SV_o, O)
Retracing Tree-Doc to the state of O produced

```
//check the effectiveness of each node in the tree document model
//If O.pos={p1,p2,…,pn-1, pn}
1: If level==length(O.pos)
2:   Return;
3: for every element Ei of Layer-Doc {
4:   Set Ei ineffective;
5:   Consider the Insert Oins of Ei, if Oins is timestamped by SVOins and SVOins≤ SVo {
6:      set Ei effective;
7:   }
8:   For each Delete Odel of Ei {
```

```
9:    if O_del is timestamped by SVO_del and SVO_del ≤ SV_o {
10:     set E_i ineffective;
11:   }
12: }
13: count No.p[level] effective element from left to right in Layer-Doc, marked E[p[level]];
14: Tree-Doc=E[p[level]].childPtr;
15: Retracing (Tree-Doc, level+1, SV_o, O );
16: }
```

5.2. Intention Preservation of Grant Operation based on Tree-AST

When a Grant([right],list[object],[role],Initiator,SV) operation is received by local site from upper site if there are Insert or Delete operations that are concurrent with it, the document status may be different with that when the Grant operation is released. Thus, in order to guarantee the correct execution of the Grant operation, the Retracing procedure should firstly be invoked to retrace the document status to that of SV. Then Grant is executed under SV and the document status is re-retraced to current status. Procedure 2 shows the execution process of the Grant operation in remote sites.

```
Procedure 2 Execute(Grant, Tree-Doc, SV_0 ),
Execute Grant with SV_0
1. If Grant is not causally-ready {Queue(Grant);}
2. Else {
3.  Retracing_Layer-Doc (Tree-Doc, 1, SV_o, Grant )
4.  Execute(Grant);
6.  SV_0[Grant.Initiator]= SV_01[Grant.Initiator]+1;
7.  Grant.SV=SVo;
8.  Retracing_Tree-Doc(Tree_Doc, 1, SV_o, Grant);
9. }
```

5.3. Efficiency Analysis

Function Retracing_TreeDoc (Tree-Doc, level, SV_o, O) retrace the document status of Tree-Doc to that indicated by SV_o. During the Retracing process, only one node's status in the same level of Tree-Doc is needed to be compared rather than all nodes of the tree. Therefore, the execution efficiency of this algorithm has been increased compared with AST algorithm.

Assume that the average length of the linear table in the hierarchy document node is l. In another word, the Retracing_TreeDoc process only needs to compare the operation statuses of l elements in each layer so as to determine the effectiveness of the node elements. Suppose that the number of operations attached with a node element is d, the time cost for checking the node's effective/ineffective status is O(d). Then the efficiency of the Retracing process on each layer can be expressed as $O(d \cdot l)$. Therefore, the time complexity of Retracing is $O(l \cdot level \cdot d)$, and the value of level depends on the layer of O.

The Retracing procedure of AST need retrace all nodes in the last layer. According to the former assumption about the average length of the linear table, we can infer that the number of the node elements on the lowest hierarchical tree is supposed to be $h^{level-1}$. Therefore the time complexity of Retracing in AST algorithm is $O(h^{level-1} \cdot d)$. Obviously, the execution efficiency of Tree-AST is much higher than that of AST.

6. Conflicts Resolution of Right Operations

As for Apply([right], list[object], [role], Initiator, SV), if more than one upper sites send the Grant operations, there exists an unmentioned situation that different rights may be required by the Apply operations. Moreover, the intention of the Grant operation may be destroyed by that of the concurrent editing operations (such as Delete or Update). Thus, it is necessary and urgent to maintain the operating intention in this situation. Firstly, the definitions of two kinds of conflicts are expressed as follows:

Definition 2: Apply- Apply Conflict

If O1, O2 are both Apply operations, there will be a conflict between O1 and O2, iff:(1) two operations are concurrent;(2) O1 and O2 apply to set different values of the same right of the same object at the same site.

Definition 3: Apply-Edit Conflict

If O1 is an Apply operation, O2 is a Delete operation or an Update operation, there will be a conflict between O1 and O2, iff: (1) two operations are concurrent; (2) O1 grant (Updateable=false); (3) O1 and O2 aim at the same object, or an object in O1 is an ancestor node of the object in O2.

6.1. Resolution Strategy

For the above two kinds of conflicts, the following three conflict resolution strategies(Grant-Grant Conflict resolution strategy based on Upper-Role-Priority, Apply-Delete Conflict resolution strategy and Apply-Update Conflict) are detailed in the following.

1) The Apply-Apply Conflict resolution strategy is based on Upper-Role-Priority strategy. That's to say, if two Apply operations are conflict with each other, the operation released by Upper Roles will win. This procedure is described in Procedure 3.

Procedure 3 Solve_Apply-Apply (O1, O2, RoleTable)
Solve the conflict of two Apply operations

```
//Suppose two operations O1 and O2 are both Apply operations
1. For each object1 in O1list[object], object2 in O2list[object], role1 in O1[role], role2 in O2[role]
2. {
3. If (object1=object2 & role1=role2 & O1[Updateable=true] & O2[updateable=false])
4. Then { compare the positions of initiator1(O1) and initiator2(O2) in RoleTable;
5.      If (they are sibling nodes )
6.      Then { Accept(O2), Deny(O1);}
7.      If (they are not sibling nodes)
8.      Then { According to the principle of upper role priority, the upper receives an accept, the lower receives a deny;}
9.    }
10. }
```

2) The Apply-Delete Conflict resolution strategy is described in Procedure 4.

Procedure 4 Solve_Apply-Delete(O1,O2, HB),
Solve the conflict between a Apply and a Delete operation

//Suppose O1 is a Apply operation, O2 is a Delete operation, HB is history buffer of operating sequence.
1. For each object1 in O1list[object], object2 is the object of O2{
2. If (object1=object2 || object1 is an ancestor node of object2) & O1[updateable=false])
3. Then { Retracing_Tree-Doc(O2);// the new value of the node is updated by ineffective=true
4. Execute an inverse operation of O2($\overline{o2}$);
5. Append the HB of the target node element of O2 with {$\overline{o2}$, O1;} //$\overline{o2}$ indicates the reverse operation of O2
6. }

3) The Apply-Update Conflict resolution strategy is described in Procedure 5.

Procedure 5 Solve_Apply-Update(O1,O2, HB),
Solve the conflict between a Apply and an Update operation

//Suppose O1 is a Apply, O2 is an Update operation, HB is history buffer of operating sequence
1. For each object1 in O1list[object], object2 is the object of O2{
2. If (object1=object2 || object1 is an ancestor node of object2) & O1[updateable=false])
3. Then { Retracing_Tree-Doc(O2);
4. execute an inverse O2($\overline{o2}$);
5. Append the HB of the target node element of O2 with {$\overline{o2}$, O1;} //$\overline{o2}$ indicates the reverse operation of O2
4. }

6.2. Example Analysis

According to the set-up of the user roles described in section 3, suppose that there is a project director, several project leaders, who take charge of one team with some designers, an example analysis is given to solve the Apply-Apply conflict as well as the Apply-Edit conflict.

According to Figure 5, specific operation at each site is given as follows:
Site6: Og1=Apply(Updateable=true,[<2,1>,<2,2>],<5>,6,<0,0,0,0,0,1>);
Site5: Ob=Delete(<2,1,1>),SV=<0,0,0,0,1,0>;
Site3:Og2=Apply((Updateable=true,[<2,1>,<2,2>],<5>,6,<0,0,1,0,0,1>);
Site2: Or1=Apply(Updateable=false,<2,2>,<4,5>,2,<0,1,0,0,0,0>);
Or4,5=Grant(Updateable=false,<2,2>,<4,5>,2,<0,1,0,0,1,0>).

In Figure 4, the conflict resolution process is described as follows: since Og1 and Ob are concurrent in Designer 6, they are detected as Apply-Edit conflict according to the Apply-Edit definition. Ob will be retraced according to the conflict resolution strategy and the node denoted by <2,1,1> is marked as inefficient. When designer 6 hands the Apply operation over to Team Leader 3, sine there is also a concurrent conflict operation Ob, the conflict resolution process will also be done just like that on Designer 6. Next, Team leader 3 hands Og2 over to the Project Director, and at the same time, he receives Or1 from Team leader 2. The Apply-Apply Conflict occurs on this site. Then Upper-Role-Priority strategy is used to solve this conflict. Because Team Leader 2 is in the second layer while Designer 6 is in the third layer, the Apply from Team Leader 2 will be accepted while that from Designer 6 will be denied. Team leader3 sends Deny information to Designer 6. Meanwhile Ob has gotten to the site6, since Ob is not conflict with Or1, Project Director sends a successful message for the Apply

and Ob is executed. In site2, the arriving of Ob and the generation of Or1 erupt simultaneously but not conflict. Ob is executed, and after Team Leader 2 receives Accept information, he is supposed to send Or4 and Or5 respectively to the Designer 4 and the Designer 5. When two sites receive Grant from Team leader 2 after executing Ob, they will achieve these requirements.

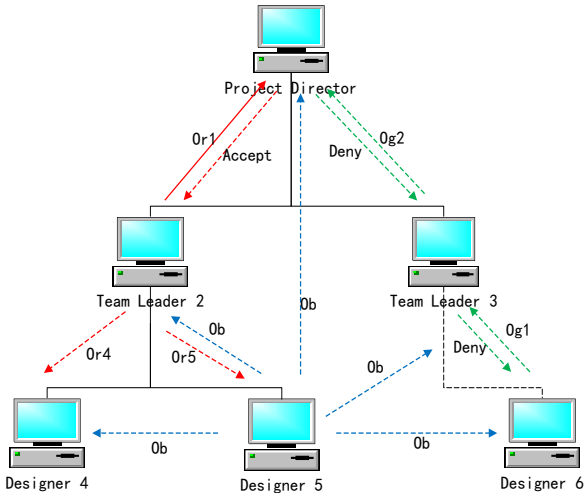

Figure 4. Resolution Processes of Apply-Apply Conflict and Apply-Edit Conflict

7. Conclusions and Future Work

Real-time collaborative systems usually adopts unlimited cooperative mode which increases the concurrency of the system but each member has a unified standard of permissions, which allow them to execute any edit operation for any object of the shared document. Thus, the implementation of right allocation can only rely on external agreement (for example: the oral agreement). If some users cannot execute the external agreement rigidly, the efficiency of the whole project will be damaged, resulting in reduction of the scope of the real-time cooperation. This paper proposes to maintain the relationships among different designers by building the hierarchical RoleTable of users, defines the tree structure model of the document supporting the right allocation and details the format of right operation, discusses the set-up of the site IDs as well as the initiating process of the system, and imposed optimized AST to preserve the intention of the right operation, discusses the conflict resolution of Apply-Apply, Apply-Edit operations. The right allocation strategy has been used in Co-AutoCAD real-time collaborative model we have ever developed, and the execution effect has obtained the recognition and affirmation from the cooperative members.

Further work in this paper includes applying the right management strategy to the real-time collaborative environment to support the division of the work while not destroying the high responsiveness and concurrency characteristics of the environment.

Acknowledgements

The work is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61202376 and No. 61003031, "Chen Guang" project sponsored by Shanghai Municipal Education Commission and Shanghai Education Development Foundation under Grant No.10CG49, and Innovation Program of Shanghai Municipal Education Commission under Grant No. 13YZ075.

References

[1] C. Sun Agustina and D. Xu, "Operational Transformation for Dependency Conflict Resolution in Real-time Collaborative 3D Design Systems", Proceedings of the Conference on Computer Supported Collaborative Work, **(2012)**, pp. 1401-1410.
[2] C. A. Ellis and S. J. Gibbs, "Concurrency control in groupware systems", proceedings of the 1989 ACM SIGMOD international conference, New York, **(1989)**, pp. 399-407.
[3] D. Sun and C. Sun, "Operation Context and Context-based Operational Transformation", Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work, New York, **(2006)**, pp. 279-288.
[4] N. Gu, J. Yang and Q. Zhang, "Consistency maintenance based on the mark & retrace technique in groupware systems", proceedings of the international ACM SIGGROUP conference on Supporting group work, New York, **(2005)**, pp. 264-273.
[5] D. Li and R. Li, "An Admissibility-Based Operational Transformation Framework for Collaborative Editing Systems", Computer Supported Cooperative Work, vol. 19, no. 1, **(2010)**, pp. 1–43.
[6] M. Fan, H. Fan and X. Wu, "Universal Priviledge Management System Based on RBAC in ASP.net. Computer Engineering", vol. 36, no. 1, **(2010)**, pp. 143-146.
[7] Z. Jing, Y. Rui and J. Luan-sheng, "RBAC permission access control model based on data objects", Computer Engineering and Design, vol. 31, no. 15, **(2010)**, pp. 3353-3355.
[8] Z. Lei, Z. Ming-Hui, L. Tian-Cheng and M. Hong, "A Permission Management Model in Service -Oriented Architecture", Chinese Journal of Computers, vol. 28, no. 4, **(2005)**, pp. 677-685.
[9] Y. Fei and Y. Jian-hua, "Application of the Hibernate Based System of Acces Control", Microelectronics & Computer, vol. 24, no. 7, **(2007)**, pp. 206-208.
[10] L. Wei-ju, L. Lie-gen and Z. Yu, "A Universal Scheme of Authority Management Model", Microelectronics & Computer, vol. 15, no. 1, **(2009)**, pp. 1-3.
[11] F. Cui-Hao, Y. Xiu-zi, P. Wei and Z. Yin, "Multi-level and Dynamic Securtiy Access Control for CAD Models in Collborative Environment", Journal of Software, vol. 18, no. 9, **(2007)**, pp. 2295-2305.
[12] F. Xi-mei, "Access Control Strategy Based on RBAC in Collaborative Environment", Computer Engineering, vol. 35, no. 11, **(2009)**, pp. 140-142.
[13] J. Yong, J. Yu-ming and P. Si-da, "Research and Design of User Rights Management Model Based on Workflow", Computer Technology and Development, vol. 19, no. 1, **(2009)**, pp. 161-164.
[14] Y. Feng, L. Gao, N. Gu and F. Wang, "Locking Intention Preservation Based on Address Space Transformation Technique", Proceedings of the Pervasive Computing and Applications, **(2008)**, pp. 497-502.
[15] G. Li-ping, C. Qing-kui1 and Y. Yi-cheng, "Research on Consistency Maintenance Technique Supporting Group Division in Real Time Collaborative Environment", Journal of Chinese Computer Systems, vol. 34, no. 1, **(2013)**, pp. 33-40.
[16] L. Gao and T. Lu, "Achieving Transparent & Real-time Collaboration in Co-AutoCAD Application", Journal of Universal Computer Science, vol. 17, no. 4, **(2011)**, pp. 1887-1912.
[17] Research on real time collaboration technique supporting right management of layer-based document.Application Research of Computers, vol. 29, no. 5, **(2012)**, pp. 1690-1694.

Authors

Liping Gao, Qiongqiong Fu, Lili Gao, Yuben Zhang, QingKui Chen