Teaching Data Structures and Software Architecture while Constructing Curriculum Platform

Min CAO, Zhen CAO
School of Computer Engineering & Science
Shanghai University
Shanghai, China
mcao@shu.edu.cn

Abstract—Data Structures and Software Architecture are two important courses of the subjects allied to computer science. Both of them are abstract and need practical experience to understand well. This paper discusses how to apply the different knowledge to the teaching process of Data Structures and Software Architecture by constructing the curriculum platform, which is a web instructional system. An optimal algorithm for generating authority tree dynamically applied to the developed web system is proposed. The knowledge hidden in the details of the algorithm is applied to the teaching. It is meant to explore a new method of teaching, rich the teaching content, grow the students' interest of curricula, and improve their ability of practice.

Index Terms—Data Structures, Software Architecture, Curriculum Platform, TP-RBAC Model, authority tree

I. INTRODUCTION

Data Structures is one of the most important basic courses of the subjects allied to computer science, and Software Architecture is a fresher and more advanced course rooted in the software engineering. Therefore, they are designed for different students at different level. In recent years, almost every university has given more support to the curriculum reform and construction. This paper discusses how to apply the different knowledge to the teaching process of Data Structures and Software Architecture by constructing the curriculum platform. It's meant to grow the students' interest in curricula and improve their ability of practice. This teaching method has achieved good results.

II. THE PROBLEMS OF TEACHING AND CURRICULUM CONSTRUCTION

Data Structures includes the organizational logic and physical storage methods of data which computer deals with. As facts have shown that the achievement and quality of software system are both dependent on the appropriate data structures. Thus, Data Structures is one of most important fundamental courses of computer science, and also an infrastructure course in some subjects like communication and automation.

However, as a course, Data Structures is highly abstract and strict in logic. Students often feel having difficulty to grasp it. That is usually caused by these following points. First of all, Data Structures is one kind of mathematic and abstract presentation which are universally considered to be boring. Secondly, students were bewildered by how to apply it in practice. Finally, mastering one high level programming language is the basis of the data structure learning, so students those who have poor capacity of programming even feel scared of data structure curriculum[1].

Software Architecture is used to direct the series of abstract design of large software system. As a draft of a system, software architecture describes the abstract components which directly constitute a system. Then, at the implementation stage, the abstract components will be translated into some practical ones. Appropriate architecture of software can improve its production efficiency and reduce the degree of difficulty of maintenance. With software system is becoming more and more complex, software architecture has caused more attention and become one advanced and indispensable course for senior software developers and specialized learner of Software Engineering.

As Software Architecture is an abstract course, the practical experience of project development is generally necessary for one to comprehend it deeply. However, it is just practical experience that the undergraduates usually lack of. Furthermore, it often infeasible for undergraduates to verify the accuracy of one software architecture design, as it is cumbersome and difficult. So, it's difficult for software architecture learner to understand the theory in class[2].

The purpose of curriculum construction is to improve the quality of education, stimulate students' interest, and increase teaching interaction. But in reality, lower grades often feel they have less opportunity to practice, while some senior students feel it is inefficient and difficult to realize the practice opportunities they have got.

For solving above-mentioned problems in the curriculum construction and teaching process, we propose a kind of model which combine some courses, like Data Structures and Software Architecture which have different difficulty levels, with one engineering practice in teaching process. By summing up knowledge at different levels of difficulty from the same engineering practice, teachers can make them corresponding to the knowledge points in teaching. By doing this, students can comprehend the knowledge preferably through combining with the practical examples.

978-1-4244-9718-8/11/$26.00 ©2011 IEEE

III. ACHIEVEMENT OF CURRICULUM CONSTRUCTION AND ITS APPLICATION IN TEACHING

In the process of curriculum construction, a web instructional system is designed and developed. The system consists of Course Introduction, Assignment Administration, Courseware Administration, Educational Administration, Interactive Teaching, and System Administration. These six subsystems separately illustrated as Fig. 1. Each subsystem can be divided into more small subsystem too. For instance, Assignment Administration is divided into Download Homework, Upload Homework and Correct Homework, while System Administration includes Userinfo Management, Password Management these two subsystems. Each subsystem in different level can be divided into more subsystems continuously if needed. Fig. 1 is the description of the software architecture which gives priority to system functions.

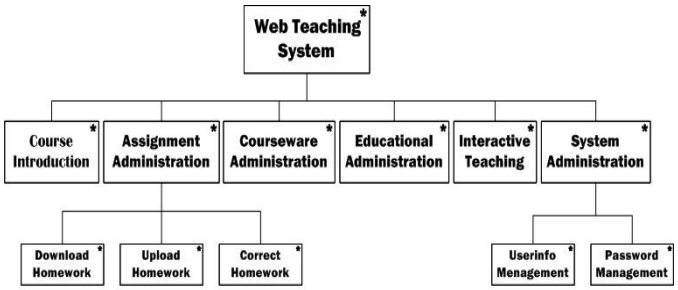

Figure 1. Software architecture of Web Instructional System

A. TP-RBAC Model and Software Architecture

Because there are different user roles in this web instructional system, it is necessary to judge whether a user is authorized to access some function or modify some data. In other words, access control and management should be considered in the system.

In computer systems security, RBAC(role-based access control) is an approach to restricting system access to authorized users. It is a newer alternative approach to MAC(mandatory access control) and DAC(discretionary access control). When defining a RBAC model, the convention of permissions is useful. Permission means an approval of a mode of access to a resource. However, the description of the resource in a software system relates to the software architecture. For example, Fig. 1 shows the software architecture of the web instructional system. One problem is whether a function, such as upload homework, is the same permission for the teacher and for the student. Therefore, this paper proposed TP-RBAC(tree page-based RBAC) model to ensure system security.

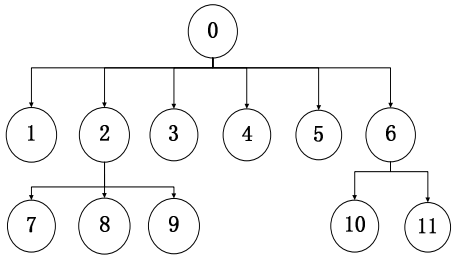

Figure 2. Abstract tree of function page

According to the proposed access control model, Fig. 1 is abstracted to Fig. 2, where each subsystem can be divided into some web pages practically, and the path from login page to every sub-page is unique. The upperclassmen attend the architecture design of the system. They adequately taste different description of software architecture from different point of view. For example, one kind of module design is as follows: both download homework and upload homework call database access module, as shown in Fig. 3.

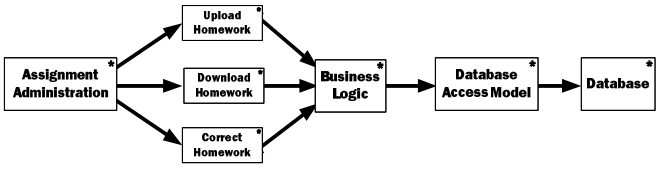

Figure 3. Module composition of assignment sub-system

The access administration system under TP-RBAC model makes it easy to modify function pages or change users' authorization dynamically. All the above modifications can be quickly implemented though adding, deleting or modifying sub-tree in Fig. 2. At the same time, TP-RBAC makes it possible to separate logic design, page display of a software system from its database design. When the different tasks are accomplished by different teams, the whole system is built by assembling the tasks with each other according to the function tree. It can greatly improve the efficiency of development. Meanwhile, the clear structure is convenient for system maintenance and system reuse.

In the teaching process, this kind of model design let software architecture learner realize the general design idea of complex system and comprehend some abstract concepts, like dynamic design, concurrent development and so on, in software architecture course more deeply. It also shows the usage of data structure knowledge to the computer beginners at the same time, such as the application of tree topology in software model design.

Though the underclassmen master hardly any knowledge of software architecture, database and user access control, once the system is abstracted to a tree structure shown in Fig. 2, they can accomplish data storage and interrelated algorithms. The next sections detail the improved algorithm for generating authority-tree dynamically based on TP-RBAC model, and show its application in teaching of Data Structures.

B. Algorithm for Dynamically Generating Authority Tree

In TP-RBAC model, the judgment of authorization is abstracted to judge if a user can access those nodes in Fig. 2. The traditional method is usually executed as follows: When a user accesses a function node on the lower branch layer from the upper layer, the software system judges if the user is authorized to access such node. If yes, access will continue, otherwise the user will be refused. However, there is a situation that a user may have to access many upper layer nodes to reach a lower one before being refused owing to lack of authorization to access the node. Then, the user may have to access those upper nodes again along previous path while backing out the original position. In this instance, it is usually a waste of time

to users and makes the system inefficient. So, this paper presents an algorithm for dynamically generating an authority tree after the user logging in the system. Namely, once the user log into the system, he is able to access each page which appears in his menu or desktop, and will never be refused through the whole access process. It can reduce the invalid operations and make the system user-friendly[3].

1) The thoughts of the algorithm: It is clearly that an authority tree of a user is a sub tree with the same structure of the tree in Fig. 2. Each node of the tree represents a page and can be accessed. There is one user marked as user A. Assume that the user A is able to access nodes 1,2,5,6,7,8,10, his authority tree is illustrated in Fig. 4.

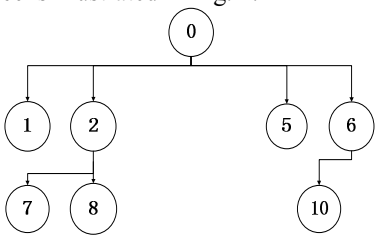

Figure 4. User A's tree of function page

a) Database design scheme and its characteristics: Firstly, a table named tbPages is designed to store the page information of the whole system. As shown in Table I, each row of tbPages corresponds to a page of the system, i.e. a node of the function tree. Besides some essential information(such as URL, name and so on) of a node, tbPages stores PageID, ParentID, and Depth such three key fields for representing the whole information of one node in tree structure. PageDd field represents the unique identifier a node, ParentID field is used to store its father node, and Depth field to the depth of the node in tree structure. Though the underclassmen lack of knowledge about database, the design of tbPages is essentially the "parent array" storage of the tree structure. Parent array is the concept of data structure course. By combining this application with teaching in class, teacher can help student understand this kind of data structure better.

TABLE I. TBPAGES FOR STORAGE OF FUNCTION PAGE

PageID	PageInfo	ParentID	Depth	Url
1	……	0	2	……
2	……	0	2	……
3	……	0	2	……
4	……	0	2	……
5	……	0	2	……
6	……	0	2	……
7	……	2	3	……
8	……	2	3	……
9	……	2	3	……
10	……	6	3	……
11	……	6	3	……

Secondarily, another table named tbUserPage shown in Table II is created to store the corresponding relationship between user and page.

As we have seen, table tbUserPage just stores the leaf nodes of one user's authority tree rather than stores all nodes of the tree. It is designed for saving memory space, as well as quickening up the storage access speed and the search efficiency in a database. That is a very good example for students to study and practice the concepts of storage structure and space complexity [4].

TABLE II. TBUSERPAGE

UserID	PageID
A	1
A	5
A	7
A	8
A	10

b) Tree-search algorithm design: As the access authority has been abstracted to nodes in the tree by TP-RBAC model, we can convert authority search to tree search abstractly. There are two conventional algorithms for tree search: depth-first search and breadth-first search. In our system design scheme, the breadth-first search is preferred, because we'd like to dynamically generate an authority tree through searching the leaf node stored in table II and connotative path stored in table I. When all the nodes of an authority tree are searched from bottom to top by breadth-first search algorithm, the queue structure helps us to take turn getting nodes in same layer enter the queue from its rear and delete from the queue in its front to realize tree traversal .

Definition 1 Queue: The queue is one kind of FIFO(first in first out) linear list. In the queue, the end allowed to insert component called rear and the end allowed to remove component called front. When a component has been inserted into the queue, it can be removed only when all the components enter queue earlier have been removed.

c) The optimization of tree search algorithm: Traditional search methods include sequential search, binary search and so on. In order to enhance the efficiency of tree search, the algorithm is optimized by combining with the thought of hash search algorithm.

Definition 2 Hash Table: The hash table is a kind of array which contains keywords and has fixed size. We can get the projective relationship from keywords to components of an array by using the hash function. One component can be found from the hash table quickly when we provide this component's keyword to the hash function.

Based on the characteristic of TP-RBAC model, the application of the queue and the hash table in proposed algorithm is introduced briefly. Instead of taking all authority nodes out from database to form the tree, we firstly get leaf nodes of current user from table tbUserPage and insert them into a queue in sequence according to their depth. On the other hand, the hash table will be used to store the sub trees and each sub tree includes a node and a key pointing its father node.

Then, the concrete design and implementation of the algorithm is discussed. Considering that all the leaf nodes in some users' authority tree have entered the queue according to their depth, it ensures that the deeper nodes enter the queue preferentially. While the nodes in the queue are deleted one by one from the front of the queue, the algorithm judges if the node has a father node. If yes, its father node is inserted into

rear of the queue and this node is put into the hash table as its key points to its father node. Otherwise, put this node into the hash table too and make its key pointing to the root node. This process is repeated until the queue is empty. Finally, the authority tree of some user is built by traversing the hash table.

Comparing with breadth-first search , the characteristics of the optimized algorithm are only the effective nodes instead of all nodes enter the queue and every father node can be found by those existing nodes. As hash table can be used to search data efficiently by this algorithm, it is possible to get the authority tree from the hash table quickly when the whole process ends. Therefore, the designed algorithm has improved the efficiency of search and saved much storage space.

2) Implementation of the algorithm: The main methods of class Queue and Hashmap are as follows:

```
Class Queue {
    count(): get the numbers of the nodes in a queue.
    enqueue (): put a node into queue.
    dequeue(): remove the node at the head of the queue.
    ifExistParNode(Object): judge if one node has a father node.
    curNode(): get current node.
    curParNode(Object): get one node's father node.
}
Class Hashmap {
    emptyHashmap(): empty the hash table.
    AddNode(object1, object2): put one component into hash table
        and make its value being object1,its key being object2.
}
```

According to the methods and data structures above, the flowchart of the algorithm is illustrated by Fig. 5.

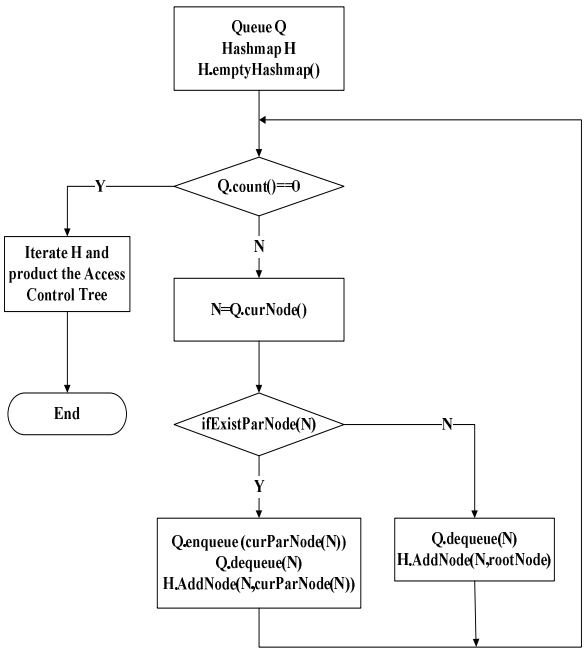

Figure 5. Flowchart of the algorithm

While optimizing the algorithm, we use two types of data structures, viz. queue and hash table, for realizing the data ordering, the data search, and the data traversal. The students may comprehend many boring and abstract concepts deeply and be impressive to some knowledge through above application[5].

C. Test and Verification of the Algorithm

The algorithm for dynamically generating users' authority tree is tested and verified with following experimental conditions and results.

1) Test environment: Its hardware includes IBM R60e, Intel(R) Celeron(R) M CPU 440 1.86GHz, 1G Memory, and software contains visual studio 2005, .NET framework 2.0, and SQL server2000.

2) Test data: According to the scale of normal web system, we in all add 100 nodes in database ten times and the number of nodes is different at each time. Assume that the number of user's access authority is increasing gradually according to the number of nodes. Test results in Table III shows the feasibility and efficiency of the proposed algorithm.

TABLE III. TEST DATA OF THIS ALGORITHM

Number of a user's access rights nodes	Number of total access rights nodes	Depth of access rights node	Execution time	number of lines in tbUserPage
5	5	3	0.0028	4
13	15	3	0.0044	7
20	25	3	0.0064	15
27	35	3	0.0097	20
36	49	4	0.0121	24
41	60	4	0.0172	24
47	70	5	0.0228	28
50	80	5	0.0250	29
57	90	6	0.0271	33
63	100	7	0.0288	37

3) Test results: Fig. 6 shows the relationship between the number of nodes in whole system and the executing time of the algorithm. When the number of nodes in the whole system tree is less, the execution time varies directly as the number of nodes, namely, there is a linear relation. As the number of nodes increases rapidly, execution time is going to be stable.

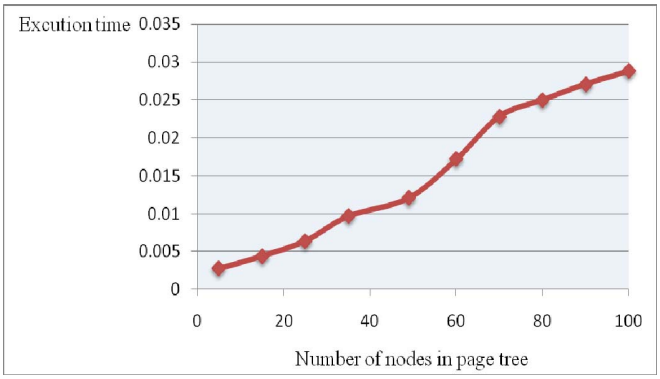

Figure 6. The execution time with the number of nodes

The relationship between the depth of the access authority tree and the execution time of the algorithm is shown in Fig. 7.When the depth of the tree is small, the execution time varies directly as the depth , with linear variety on the whole. As the depth is increasing to a certain extent, execution time is going to be stable too.

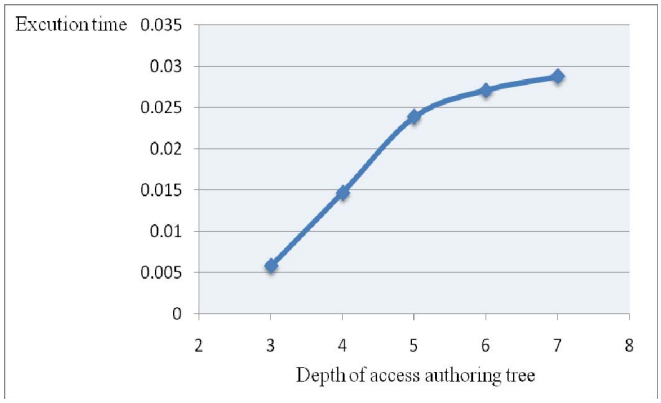

Figure 7. The execution time with the depth of authority tree

4) Test conclusions: By summarizing the graph and data in table III and Fig. 6-7, we find the efficiency of the algorithm is going to be rarely varied with the increasing of the total number of nodes and the depth of the authority tree. Both the execution time and efficiency of the proposed algorithm are acceptable.

The achievement and test of the proposed algorithm and the application of the Web instructional system make it convenient for teacher to combine practice with theory in teaching process. All the theories used in this article belong to the category of Data Structures and Software Architecture. The theories and design idea can be used to enrich teaching materials and cultivate students' interests in such two curricula.

IV. SUMMARY

The developed Web instructional system can be an example for Data Structures course and Software Architecture course to show their application in practice. It will rich the teaching content and train students' practical working ability better. Meanwhile, this method of teaching also can enhance teachers' research and interaction abilities. In the process of curriculum construction, we can let students take part in practice by building the web instructional system.

ACKNOWLEDGMENT

This work is supported by Shanghai Fifth Leading Academic Discipline Project(Project Number: J50103), and by Shanghai Education Committee Fourth Education Eelevation Project.

REFERENCES

[1] Y.Yang, D.K.Yu. The Explore and Research on Teaching of Data Structure[J]. Medical Information. 2007,.20(7):p41-43.

[2] Yousheng, ZHANG. Software Architecture[M]. Beijing:Tsinghua University Press, 2006: 12-15.

[3] Y.Liu, B.Wang, Tree-like database design model for expansible application[J]. Computer Engineering and Design. 2006, l.27(22):4074-4080.

[4] Min CAO, Shu LUO, Huaikou MIAO. Tree-based Database Design[C]. 2009 4th International Conference on Computer Science & Education, 978-1-4244-3521-0/09/$25.00 ©2009 IEEE, 1572-1575.

[5] XING Tian-yang, CAO Min. Research and application of algorithm for generating authority-tree based on TP-RBAC model[J]. Computer Engineering and Design. 2010,31 (5):950-953.