Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 13 (18): 3668-3675, 2013
ISSN 1812-5654 / DOI: 10.3923/jas.2013.3668.3675

Application for Legacy System Migration Based on UCM Components

[1]Li Aiping, [1]Wang Zhenghua, [1]Duan Liguo, [2]Li Xueping and [1]Wang HongXiang
[1]Department of Computer Science and Technology,
[2]Polytechnic Institute of Taiyuan University of Technology, Taiyuan University of Technology,
Taiyuan, 030024, China

Abstract: Most existing component extraction methods extract component after abstracting the system into UML diagrams by reverse engineering, involving very tedious steps. To solve this problem, a new type of component extraction scheme combined with the expression method of Use Case Maps (UCM), the procedure of component of extraction, the knowledge of component reusable measurement is proposed which is based on the stub and interface of UCM and a specific component extraction algorithm which is called CEA algorithm based on the analysis of UCM demand-dependent is designed to avoid the reverse engineering in this paper. For the algorithm model, we design component extraction model of legacy systems and extract information from the XML model to RDB (relational database) to verify the model component extraction.

Key words: Component extraction, use case maps (UCM), XML model, RDB

INTRODUCTION

Component Based Software Engineering (CBSE) is widely used. Now most existing component extraction methods extract components after converting the legacy system to UML diagrams by reverse engineering, involving very tedious steps. Therefore, a simple and efficient component extraction proposal is essential to avoid reverse engineering of component extraction.

Use Case Maps (UCM) can describe the system behavior, its composition structure and relationships between them in a diagram. On the one hand, UCM avoids excessive detail requirement in systems analysis and design at an early stage; on the other hand, unlike UML Use Case diagram which hides all the details, UCM contains appropriate ones. Therefore, there are obvious advantages in UCM to compare with other similar tools.

With the development of WEB applications, more and more applications will be expressed by the form of XML. There are huge amounts of XML documents on the WEB. They need to be stored in Relational Database (RDB) for effective management. How to extract useful information from the XML to be stored in RDB has become an important research topic.

RELATIVE BASIC DEFINITION

Project supported by the Science Research Project in Shanxi Province, China (20120321030) and Youth Foundation of Taiyuan University of Technology (No. 2012L067) and the open project of State Key Laboratory (SKLSE2012-09-30).

Definition 1: UCM requirement specification Component based Software Engineering (1) RS = (D, C, H, δ, Bc); D is UCM domain composed of the following elements:

$$D = SP \cup EP \cup R \cup AF \cup AJ \cup OF \cup OJ \cup S$$

SP is collection of the start point, EP is collection of the end point, R is collection of the responsibility, AF is collection of the And-Fork, AJ is collection of the And-Join, OF is collection of the Or-Fork, OJ is collection of the Or-Join and S is the collection of Stub.

C is the set of RS members. H is collection of the hyperedge connecting the various parts of the UCM (a hyperedge with one or more of the source and target). δ is collection of transferring relations, recorded as $\delta = D \times H \times D$. $Bc = D \times C$ (Hassine *et al*., 2005).

Definition 2: UCM demand dependency (Xue, 2009):

$$\text{Contain dependency: } S1 \xrightarrow{ConD} S2$$

S1 and S2 represents the static or dynamic stub. If S1 contains all of the features of S2, S1 is called to contain dependency on S2. Contain dependency is used in the

Corresponding Author: Duan Liguo, Department of Computer Science and Technology, Taiyuan University of Technology, Taiyuan, 030024, China

calling process of the stub and which sub-module to be called is determined by the specific operating environment or other modules' output:

$$\text{Function dependency: } F1 \xrightarrow{FunD} F2$$

Given two function modules F1 and F2, if there exists output association through the existing interface, F2 is called to have function dependency on F1. In UCM, individual module or component completes a task through the cooperation between them. Therefore, there is mutual dependency between various functional modules, realized through interfaces:

$$\text{Temporary dependency: } S1 \xrightarrow{TempD} S2$$

If dynamic stub S calls sub-stub Si according to the operating environment of the system, there exists temporary dependency between Si and S. Temporary dependency is mainly reflected in the calling process of dynamic stub, choosing sub-modules to be called based on the operating environment of the system.

Definition 3: XML logical data model (Wang *et al*., 2008): Given an XML document D, it can be abstracted as an ordered tag tree T = (V, V^0, E, Σ, type, lab, val, ≤) (Jagadish *et al*., 2002), in which V is a collection of XML nodes; V^0 is the root node of the tree;the binary relation $E \subset V^2$ is the set of edges, if u is the parent node of v, then $(u, v) \in E$; Σ is a finite alphabet and the collection of the document D's elements and attributes; function type: $V \rightarrow \{elem, attr, text\}$, determines the node type, if v is the element, then type (v) = elem, if v is the attribute, then type (v) = attr, if v is the text, then type (v) = text; $Ve = \{v|v \in V \wedge type(v) = elem\}$ is a collection of element nodes, $Va = \{v|v \in V \wedge type(v) = attr\}$ is a collection of attribute nodes, $Vt = \{v|v \in V \wedge type(v) = text\}$ is a collection of text nodes; function lab: $Ve \cup Va \rightarrow \Sigma$, returns the name of the elements or attribute nodes; function val: $Va \cup Vt \rightarrow String$; returns the value of the attributes or text nodes, String is the collection of all legal documents in XML; The binary relation $\leq \subset V^2$, defines the order of XML document, if u appears v before or u = v, then $(u, v) \in \leq$ or recoded as $u \leq v$.

COMPONENT EXTRACTION

Component and component model: A compent can be described as a four-tuple:Component (CIdentity, CDescription, CSubject, CInterface).

Cidentity refers to Component identifier, CDescription component information description, in favor of retrieval and reuse of component, CSubject represents subject or entity that contains the program code and design documents and CInterface component interface, including external physical interface and external logic interface which is designed to provide users with non-formal component description by Component Definition Language (CDL) and Component Manipulation Language (CML).

Component model is divided into internal statute and External interface. The internal statute takes semantic model of its structure and service features into consideration, such as interaction protocols between components, the function set it can provide and so on; the external interface consists of two categories: Join point, the external interface used by the component and function Statute, component interface for external use.

Introduction to component extraction process: The purpose of component extraction is mining reusable component from existing legacy systems. Currently, there are a large number of reusable components, including relational databases and network controls like VBXs and ActiveX, JavaBean, DLL, interfaces libraries and API, which can effectively improve the efficiency of system development; however, common components oriented toward business of specific areas are relatively scarce. Currently, the main task of component extraction is to identify and measure the reusable module, then extract relatively generic parts into high-quality reusable components.

Extracting reusable components from the legacy systems generally goes through the following phases, as shown in Fig. 1.

Representation of XML tree structure of component extraction for generalized list: The main contents in the process of component extraction include the basic information, feature set, design patterns. Generalized List

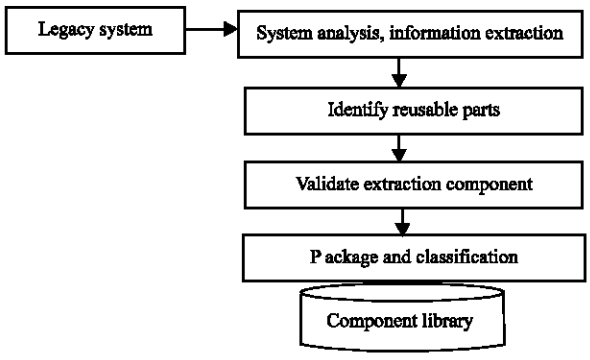

Fig. 1: Process of component extraction

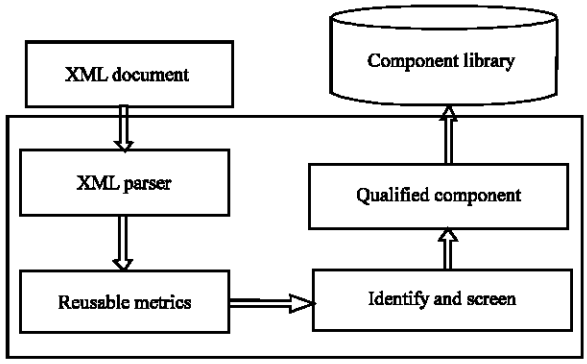

Fig. 2: Component extraction model based on UCM

of Tree Structure can be represented as: Component (Basic information (version, language, operating system, publisher), Feature set (attributes (the number of writing methods, the number of reading methods, the number of parameters), Method (Inheritance, premise, result, return type), Event (event type, monitor type)), Design mode (affiliated component, connection, combination)).

UCM component extraction mode: UCM extraction process is measuring the reusability of legacy system described by the UCM and comparing the results and particle size of the component with user requirement for further identification and screening, as shown in Fig. 2.

In Fig. 2, XMLParser generates UCM show from XML file, which is then measured by reusable gauges metrics and compared with user's input conditions for component identification and screening (CEA algorithm) so as to extract reusable components and store them in the library.

Component extraction method based on UCM: Component extraction is extracting high quality reusable module elements from existing software systems for the purpose of component reusability. High quality component is extracted based on its association with UCM component association-the contain independency and function independency have to satisfy the condition. With a clearer level during the process of stub calling, UCM is much better than the UML diagram. This paper takes the approach of UCM top-down to slice candidate component, namely screening big particle widget and then screening small widget:

- **Question description:** Based on the feature of UCM demand dependency, we further proposed to measure the reusability of contain independency, function independency, temporary dependency and identify and screen qualified components or modules as reusable components in the process of UCM component extraction contain independency is that the software component can realize the extent of their functions on its own which does not depend on the operating system, other components, connectors, or runtime environment constraints. It can use the following formula to measure the contain independence of stub:

$$C\_ind = \frac{1}{N + \sum_{i=1}^{k-1} (NumChild(i) - 1)} \quad (1)$$

N indicates the sub-stub which root stub directly contains, k represents the number of dynamic stub and NumChild (i) the number of stubs that the i-th dynamic stub contains. The lower the C_ind value is within the range of (0,1), the less independency the dynamic stub contains and the better reusability the component has.

Function independency: The function dependency interacts with other modules via interface, function independency measurement also through Interface to be calculated. The formula is:

$$F\_ind = k * ID(i) + (1 - k) * ID(0) \quad (2)$$

F_ind is at the range of (0,1); k, as a balance factor value, is the proportion of the input interface component accounted for the interface of the total number. The lower the F_ind value is, the higher the component reusability is.

Temporary dependency formula is:

$$Tempd = \frac{1}{n} \quad (3)$$

n is the number of sub-stubs contained in the measured stub; the larger the Tempd value is within the range of (0,1), the lower the temporary dependency is and the better the quality of extracted component is.

Taking the reusability metrics model of UCM-based component, this paper defines objective function s (μr) to reflect reusability based on UCM:

$$S(\mu r) = \frac{1}{\mu 1 * C\_ind + \mu 2 * F\_ind + \mu 3 * Tempd} \quad (4)$$

μ1, μ2, μ3 are impact coefficients of corresponding metrics on component reusability within the range of (0, 1).

The target of the model function: given values of μ1, μ2, μ3 within the range of (0,1), measure UCM-based system in terms of C_ind, F_ind, Tempd so as to maximize the value of function S (μr).

- Question solving: The core of component extraction is the UCM stub. Component extraction assembles high quality reusable component with hyperedge and responsibility of the strong correlation of the core stub

Component extraction algorithm is measured by meeting the condition of contain independency and function independency. Contain independency for dynamic stub which measured by sub-stub contained. Function independency is measure by the standard of input and output interfaces.

The UCM component extraction process: If traversing UCM to stub, it need to measure the stub which meets the condition as the core stub which will be connected to the stub's hyperedges and responsibility as qualified component.

Adding the component obtained to the output set, then the same way for traversing the stub of contain stub. When all of the contain stub traversed finished or does not have contain stub, using this stub as a start point to continue follow-up traversal until UCM traversal finished:

- Implementation of Component Extraction based on UCM

Algorithm: Traverse UCM tree of legacy systems expressed by UCM for the purpose of maximizing the target function S' value, identifying and extracting the better reusable components until the entire UCM traversal finishes, then the algorithm ends.

Description of algorithm steps:

Step 1: Traverse the UCM tree until no stubs contain any stub or all the contained stubs are traversed and then turn to step (3), Otherwise, calculate reusability of stubs and turn to step 2

Step 2: Screen the core stub by parameters of reusability produced by Step 1 and upper and lower bounds of entered component and record the one that connects the stub's hyperedges of the core stub as qualified component and turn to step 3

Step 3: Start with this stub, continue to traverse the next elements and turn to step (1). The algorithm exits until all the stubs have been traversed

The pseudo code of Component Extraction Algorithm:

```
/*
d1: field value of contain independency
d2: field value of function independency
d3: field value of temporary dependency
c1: the maximum size of component
c2: the minimum size of component
*/
CEA(ucm,d1,d2,d3,c1,c2){
// traverse UCM, record the current element as u
for(each u ? UCM)
 /*
Trafinish[u] judges whether u has completed a full traversal with the initial value of 0 and value of 1 when traversal completed
 */
if (!isStub(u)||(isStub(u)&&trafinish[u]))
        start=u->next;
else{ // calculate contain independency
        c_ind[u] <- C_ind(u);
// calculate function independency
        f_ind[u] <- F_ind(u);
        // calculate temporary dependency
        tempd[u] <- Tempd (u);
        S[u]=S(C_ind,F_ind,Tempd);
if(c_ind[u]<d1 && f_ind[u]<d2&&tempd<d3 &&S[u]<c1&&S[u]>c2){
        comp <- composeComponent(u);
// store as attributes and component
        compf <- f;
        component <- component + comp;
        CEA(ucm,d1,d2,d3,c1,c2);
} }}
```

In the above algorithm, it is more accurate to determine parameters and extract component according to users' requirement and actual situation. When extracting reusable component from multiple legacy systems whose system function is similar or the same, the probability is high for screening high-quality reusable component. To compensate for the defects of other algorithms, the algorithm extracts component with higher reusability by describing and comparing the contain independency, function independency and temporary dependency:

- Algorithm analysis

Analysis of time complexity of the algorithm: Calculating traversed elements of contain independency-assuming UCM contains n stubs and each dynamic stub contains an average number of sub-stubs k, then the complexity of contain independency of each stub is O (k!) and the independency of the total stubs is recorded as O(nk!); the number of input and output of the computation of function independency is denoted as e, then the complexity of function independency is O (e); therefore, the time complexity of the entire algorithm is O (nk! + e). Compared with (Luo *et al*., 2004) of which the time complexity is $O(mn^k logN)$ and literature by McCall *et al*. (1977) of which the time complexity is O (2elogN + logN), as shown in Fig. 3, UCM is much more efficient and has practical values.

Algorithm	Time complexity	Complexity analysis
Literature(5)	O (mnklogN)	Related with k,as k increases, complexity exponential grows
Literature(6)	O (2elogN+logN)	Cohesion metric algorithm complexity is O (2elogN), segmentation algorithm is complexity O (logN)
Extraction component basedonUCM	O (nk!+e)	Contain independency is O (nk!),function independency is O (e)

Fig. 3: Comparison of extraction methods

```
<?xml version="1.0" encoding="UTF-8"?>
<tables>
    <table id="T_SYSTEM_LICENSE">..
    <table id="T_USER">..
    <table id="T_USER_INFORMATION">..
    <table id="T_USER_IMG">..
    <table id="T_FRAME_MODEL">..
    <table id="T_THEME">..
    <table id="T_LOGIN_LOG">..
    <table id="T_WORKSITE">..
    <table id="T_UNIT">..
    <table id="T_UNIT_INFORMATION">..
    <table id="T_USER_REGISTER_NUM">..
    <table id="T_USER_UNIT_RELATION">..
    <table id="T_ROLE_BUSINESS">..
    <table id="T_USER_ROLEBUSINESS_RELATION">..
    <table id="T_ROLE_SYSTEM">..
    <table id="T_USER_ROLESYS_RELATION">..
    <table id="T_ROLE_RESOURCE_RELATION">..
    <table id="T_RESOURCE_SYSTEM_IMG">
        CREATE TABLE T_RESOURCE_SYSTEM_IMG  (
           ID                   VARCHAR2(30)                    not null,
           RESOURCE_ID          VARCHAR2(30)                    not null,
           IMG_NAME             VARCHAR2(100),
           IMG_TYPE             INTEGER,
           constraint PK_T_RESOURCE_SYSTEM_IMG primary key (ID)
        )
        <initData>
            INSERT INTO T_RESOURCE_SYSTEM_IMG ( ID, RESOURCE_ID, IMG_NAME,IMG_TYPE )
               VALUES ('1', '1', 'system.png', 1)
        </initData>
```

Fig. 4: Fragment of XML

INSTANCE OF MIGRATION VALIDATION

Storage of XML document: The W3C formulates standard programming interface-Document Object Model (DOM) to process XML documents in a uniform manner. However, DOM is only a logical but not persistence model, whose establishment means to define the persistence of XML node. This process involves firstly parser scans XML document and generates logic nodes, which are then converted into physical nodes by indexer which add up to constitute persistent DOM. Persistent DOM is responsible for storing and accessing XML-based B+tree index. The record of the data page is physical nodes of XML, with B+tree mapping node id to the record store address. The following is the XML-based algorithm description of document storage.

Document storage algorithm: Given XML document D and parameter depth, return the paging file of D. Set reader as XML document parser, dom is persistent DOM; pseudo code of the algorithm is as follows:

```
/* read() in document order, if read successfully, return true; if read the end of document, return false. */
while(reader.read())
{ // return the current logical node
LogicNode = reader.currentNode;
// covert to physical node
PhysicalNode = LogicNode.toPhysical();
// stored as a new node
NewNode = dom.addRecord(PhysicalNode);
if(type(LogicNode) = elem and LogicNode.Depth = depth)
// establish B+ tree index
dom.addKey(id(LogicNode),NewNode); }
return dom;
```

The verification of Migration of XML model into RDB: Supported by the algorithm of persistent DOM-based XML document storage and based upon component extraction, experiments that are based on component and applies XML model in the migration of Oracle are designed in this section, of which the results verify the process of extracting the information from XML model to RDB based on UCM.

The fragment of parsing XML is shown in Fig. 4. The code of creating table of extracting information from XML to ORACLE is as shown in Fig. 5.

```
private void createTablesFromOpereateFile(){
    final File[] operateFiles = getOperateFiles();
    for(int i=0;i<operateFiles.length;i++){
        Logger.info("------"+(i+1)+"......" );
        Logger.info("--------文件 ["+operateFiles[i].getName()+"]开始读取操作......" );
        List<Table> tables = getOperateFileTables(operateFiles[i], "//tables/table");
        Logger.info("--------文件 ["+operateFiles[i].getName()+"]共有表格["+
                    tables.size()+"]个表格进行写入......" );
        for(int j=0;j<tables.size();j++){
            Table t = tables.get(j);
            try{
                isystemDao.createTable(t);
                Logger.info("------------表格 ["+t.getTableName()+"] 写入成功......" );
            }catch (SQLException e) {
                Logger.error("------------表格 ["+t.getTableName()+"] 写入失败......" );
                e.printStackTrace();
            }
        }

        if("Tables_system.xml".equals(operateFiles[i].getName())){
            Logger.info("------------******执行系统管理插入xml，特殊执行次内容...开始..." );
            createPRO_REGISTER_LOGOUT_STATUS();
            createPRO_REGISTER_LOGOUT_STATUS_IntervalRun();
            Logger.info("------------******执行系统管理插入xml，特殊执行次内容...结束..." );
        }

        Logger.info("--------文件 ["+operateFiles[i].getName()+"]结束读取操作......" );
    }
}
```

Fig. 5: Code of creating table of extracting information from XML to ORACLE

```
public void initData() {
    Logger.info("==开始执行数据写入操作......" );
    final File[] operateFiles = getOperateFiles();
    for(int i=0;i<operateFiles.length;i++){
        Logger.info("------"+(i+1)+"......" );
        Logger.info("--------文件 ["+operateFiles[i].getName()+"]开始读取操作......" );
        List<Table> datas = getOperateFileTables(operateFiles[i], "//tables/table/initData");
        Logger.info("--------文件 ["+operateFiles[i].getName()+"]共有数据["+datas.size()+"]即将进行写入......" );
        for(int j=0;j<datas.size();j++){
            Table t = datas.get(j);
            try{
                isystemDao.operate(t.getTableCode().replaceAll("'", "'"));
                Logger.info("------------数据 ["+t.getTableCode().replaceAll("'", "'")+"] 写入成功......" );
            }catch (SQLException e) {
                Logger.error("------------数据 ["+t.getTableCode().replaceAll("'", "'")+"] 写入失败......" );
                e.printStackTrace();
            }
        }
        Logger.info("--------文件 ["+operateFiles[i].getName()+"]结束读取操作......" );
    }
    Logger.info("==结束执行数据写入操作......" );
}
```

Fig. 6: Code of initialization of table records from xml to oracle

Extract the initialization information from XML table and transplant it to the Oracle database to create table records, as shown in Fig. 6.

EXPERIMENTAL RESULTS

The results of Initialization to create tables are shown in Fig. 7.

```
开始执行数据写入操作.....
--开始获取执行需要操作的文件夹中的文件.....
--已获取执行文件 [2]个.....
----1.....
------文件 [Tables_system.xml]开始读取操作.....
------文件 [Tables_system.xml]共有数据[26]即将进行写入.....
----------数据 [
                INSERT INTO T_RESOURCE_SYSTEM_IMG ( ID, RESOURCE_ID, IMG_NAME,IMG_TYPE ) VALUES ('1', '1', 'system.png', 1)
            ] 写入成功.....
----------数据 [
                INSERT INTO T_RESOURCE_SYSTEM_IMG ( ID, RESOURCE_ID, IMG_NAME,IMG_TYPE ) VALUES ('2', '2', 'bullet_green.png', 1)
            ] 写入成功.....
----------数据 [
                INSERT INTO T_RESOURCE_SYSTEM_IMG ( ID, RESOURCE_ID, IMG_NAME,IMG_TYPE ) VALUES ('3', '3', 'bullet_green.png', 1)
            ] 写入成功.....
----------数据 [
                INSERT INTO T_RESOURCE_SYSTEM_IMG ( ID, RESOURCE_ID, IMG_NAME,IMG_TYPE ) VALUES ('4', '4', 'bullet_green.png', 1)
            ] 写入成功.....
----------数据 [
                INSERT INTO T_RESOURCE_SYSTEM_IMG ( ID, RESOURCE_ID, IMG_NAME,IMG_TYPE ) VALUES ('5', '5', 'bullet_green.png', 1)
            ] 写入成功.....
----------数据 [
                INSERT INTO T_RESOURCE_SYSTEM_IMG ( ID, RESOURCE_ID, IMG_NAME,IMG_TYPE ) VALUES ('6', '6', 'bullet_green.png', 1)
            ] 写入成功.....
----------数据 [
                INSERT INTO T_RESOURCE_SYSTEM_IMG ( ID, RESOURCE_ID, IMG_NAME,IMG_TYPE ) VALUES ('7', '7', 'bullet_green.png', 1)
            ] 写入成功.....
----------数据 [
                INSERT INTO T_RESOURCE_SYSTEM_IMG ( ID, RESOURCE_ID, IMG_NAME,IMG_TYPE ) VALUES ('8', '8', 'bullet_green.png', 1)
            ] 写入成功.....
----------数据 [
                INSERT INTO T_RESOURCE_SYSTEM_IMG ( ID, RESOURCE_ID, IMG_NAME,IMG_TYPE ) VALUES ('9', '9', 'bullet_green.png', 1)
            ] 写入成功.....
```

Fig. 7: Initialization of creating tables

```
开始执行生产操作.....
--创建用于建表的存储过程.....开始.....
--创建用于建表的存储过程.....成功.....
--开始获取执行需要操作的文件夹中的文件.....
--已获取执行文件 [2]个.....
----1.....
------文件 [Tables_system.xml]开始读取操作.....
------文件 [Tables_system.xml]共有表格[19]个表格进行写入.....
----------表格 [T_SYSTEM_LICENSE] 写入成功.....
----------表格 [T_USER] 写入成功.....
----------表格 [T_USER_INFORMATION] 写入成功.....
----------表格 [T_USER_IMG] 写入成功.....
----------表格 [T_FRAME_MODEL] 写入成功.....
----------表格 [T_THEME] 写入成功.....
----------表格 [T_LOGIN_LOG] 写入成功.....
----------表格 [T_WORKSITE] 写入成功.....
----------表格 [T_UNIT] 写入成功.....
----------表格 [T_UNIT_INFORMATION] 写入成功.....
----------表格 [T_USER_REGISTER_NUM] 写入成功.....
----------表格 [T_USER_UNIT_RELATION] 写入成功.....
----------表格 [T_ROLE_BUSINESS] 写入成功.....
----------表格 [T_USER_ROLEBUSINESS_RELATION] 写入成功...
----------表格 [T_ROLE_SYSTEM] 写入成功.....
----------表格 [T_USER_ROLESYS_RELATION] 写入成功.....
----------表格 [T_ROLE_RESOURCE_RELATION] 写入成功.....
----------表格 [T_RESOURCE_SYSTEM_IMG] 写入成功.....
----------表格 [T_RESOURCE_SYSTEM] 写入成功.....
----------******执行系统管理插入xml，特殊执行次内容...开始...
```

Fig. 8: Initialization table records

Results of the Initialization of table records are shown in Fig. 8.

CONCLUSION

Based on UCM, this study has proposed to extract useful information from XML model to store into RDB, using information extraction from XML to Oracle as an example for verification after designing component extraction model. Future researches include improving the model of component extraction and putting it into the actual business system.

REFERENCES

Hassine, J., J. Rilling, J. Hewitt and R. Dssouli, 2005. Change impact analysis for requirement evolution using use case maps. Proceedings of the 8th International Workshop on Principles of Software Evolution, September 5-6, 2005, Canada, pp: 81-90.

Jagadish, H.V., S. Al-Khalifa, C. Adriane, V. Laks and S. Lakshmanan *et al*., 2002. TIMBER: A native XML database. Int. J. Very Large Data Bases, 11: 274-291.

Luo, J., W. Zhao, T. Qin, R.K. Jiang, L. Zhang and J.S. Sun, 2004. A decomposition method for object-oriented systems based on iterative analysis of the directed weighted graph. J. Software, 15: 1292-1300.

McCall, J.A., P.G. Richards and G.F. Walters, 1977. Factors in Software Quality. Vol. 3, Preliminary Handbook on Software Quality for an Acquisiton Manager, Springfield.

Wang, X., X.J. Yuan, C.Y. Wang and H.W. Zhang, 2008. XN-STORE: A storage scheme for native XML databases. J. Comput. Res. Dev., 45: 1211-1219.

Xue, Y., 2009. Research and application of correlation measurement and abstraction of component based on UCM. M.Sc. Thesis, Ocean University of China.