Title: An information presentation method based on tree-like super entity component

Authors: Ruijun Zhang, Jie Lu, Guangquan Zhang

PII: S0164-1212(11)00093-8
DOI: doi:10.1016/j.jss.2011.04.001
Reference: JSS 8696

To appear in:

Received date: 15-9-2010
Revised date: 31-3-2011
Accepted date: 4-4-2011

Please cite this article as: Zhang, R., Lu, J., Zhang, G., An information presentation method based on tree-like super entity component, *The Journal of Systems and Software* (2010), doi:10.1016/j.jss.2011.04.001

An information presentation method based on tree-like super entity component

Ruijun Zhang[a,b,*], Jie Lu[b], Guangquan Zhang[b]

[a] *School of Management, Wuhan University of Science and Technology, 947 Heping Road, Wuhan, PR China*

[b] *Decision Systems & E-Service Intelligence Research Laboratory, Centre for Quantum Computation & Intelligent Systems, School of Software, Faculty of Engineering & Information Technology, University of Technology, Sydney, PO BOX 123, Broadway NSW 2007, Australia*

Abstract: Information systems are increasingly oriented in the direction of large-scale integration due to the explosion of multi-source information. It is therefore important to discuss how to reasonably organize and present information from multiple structures and sources on the same information system platform. In this study, we propose a 3C (Components, Connections, Container) component model by combining white-box and black-box methods, design a tree-like super entity based on the model, present its construction and related algorithm, and take a tree-like super entity as the information organization method for multi-level entities. In order to represent structural, semi-structural and non-structural data on the same information system platform, an information presentation method based on an editable e-book component has been developed by combining the tree-like super entity component, QQ-style menu and 1/K switch connection component, which has been successfully applied in the flood protection project information system of the Yangtze River in China.

Keywords: E-book; Tree-like super entity; Component; Information presentation; Multi-source

1. Introduction

Information systems today are increasingly becoming the nerve centre of any business. Entities such as power companies (Sforna, 2000; Tsamenyi and Cullen, 2006), banks (Ravichandran and Banerjee, 1994; Waema and Walsham, 1990), transport departments (Taylor et al.), finance companies (Terpsiadou and Economides, 2009), and manufacturing industries (Coppus and Strashok, 1995) cannot operate without clear and accurate information systems. With the integration of information systems and the diversity of the requirement for information, a single data source cannot meet the information needs of businesses. Several data acquisition techniques and communication protocols related to the bar code (Manthou and Vlachopoulou, 2001) and radio frequency technology (Ko, 2009) are undergoing continuous development, and communication and network techniques improve frequently.

It is vitally important to organize and present information from multi-source, multi-regional, multi-field and multi-structured data for information systems (Yang et al., 2007; Lu et al., 2007), but it is equally important to direct the focus of intelligent information systems (Ma et al., 2010).

The Flood Protection Project Information System of the Yangtze River (FPPISYR) in China plays a very significant role in flood risk management. It also provides an important decision support tool to prevent flooding and provide a rapid response to danger for the Yangtze River (Jiang et al., 2007; Li et al., 2007; Wu et al., 2007; Xu et al., 2005). The Ministry of Water Resources of China needs to manage the geological information of more than forty culvert-brakes, dams, consolidating projects and regulating projects, and to build a digital library of the Yangtze River embankments for the purpose of sharing information. The FPPISYR is a part of that digital library. The data types involved in the system vary and include not only conventional project data, but also a large amount of geographic information, maps, photographs, reports, and so on for a variety of purposes. The system is complex and the information requirements of users vary, making it difficult to meet the demands of the system by using a single database management model.

There are various methods of organizing and representing information. Typically, traditional information systems can only process structured information but cannot effectively manage audio, images, video and other semi-structured and unstructured information. Several techniques, such as Web pages (Ettredge et al., 2001; Klitzman et al., 2009), XML (Combi et al., 2005; Manvi and Venkataram, 2005), PDA terminal (Chang et al., 2006; Kundu et al., 2007) and so on can clearly organize and represent information, but they are less interactive and not as easy to update and expand.

As an information carrier, e-book has the advantage of optionality of contents, easy retrieval, easy spread and substantial content when compared with traditional books. Its content is also well organized, which is a feature other carriers cannot achieve, but it is not convenient for editing and expanding information (Han and Jin, 2009; Kang et al., 2009; Shelburne, 2009).

Modern cognitive psychology and ergonomics indicate that different styles of information presentation have a great effect on people's ability to acquire information, their learning knowledge and decision-making skills. Niederhauser et al. (2000), Khalifa and Kwok (1999), and Quentin-Baxter (1998) compared and analyzed hypertext and linear text. It is generally believed that hypertext supports human associative thinking, but its flexible structure does not provide the user with a clear learning route, and

* Corresponding author. Tel.: +862788512213.
E-mail addresses: zrjun@wust.edu.cn (R.Zhang), jielu@ it.uts.edu.au (J.Lu), zhangg@it.uts.edu.au (G.Zhang).

as a result, the user can easily get lost. Although hypertext is conducive to browsing and searching information resources, it does not contribute to learning. Speier studied the impact of space maps, charts, tables and other different types of information on effective decision-making (Speier, 2006). All the theories above discussed the coverage and depth of information, but did not study the interaction and editability of information. Teuvo Kohonen, a Finnish academician, proposed Self-Organized Feature Maps (SOFM) according to how the brain processes information (Sun Microsystems, 1997). He considered that the human brain is a typical, self-organized system which can learn in a non-directive way and developed a model aimed at the characteristics of information presentation.

On the basis of an in-depth analysis of component technologies, this study constructs an editable e-book component by combining the advantages of management information systems and the e-book. The component is packed into an intelligent information system to represent structured, semi-structured and unstructured information on the same platform, in which users can organize information according to their own roles and present information from different dimensions to meet their personal information requirements. The FPPISYR has also been developed on the basis of an editable e-book component.

This paper is organized as follows: A component model and its formal description are given in Section 2. The concept of a tree-like super entity is presented in Section 3 with the aim of solving the problem of the connection of multiple layer entities in relational databases. An editable e-book, which mainly consists of a tree-like super entity component, QQ menu component and 1/K switch connection component, is outlined in Section 4. Conclusions are discussed in Section 5.

2. The component and its connection

In this section, we will give the definition of an atomic component, component and the connection between components.

2.1. Component

There are essentially three traditional types of component: COM (Feng et al., 2007), CORBA(Sun et al., 2009) and JavaBeans (Bruce, 2004), but no unified definition of a component has yet been proposed. Liu et al. defined components as re-usable software modules which can be packed in object classes, a set of functional modules or a software framework (Liu et al., 2000). Cox and Song (2001) considered that a component was a software unit with a certain independent function which could be a basic component or a complex component having a combination of several basic components. D'Souza and Wills (1997) defined a component as a coherent package of software that can be independently developed and delivered as a unit. Leach (1997) stated that a software component is a unit of composition with contractually specified interfaces and explicit context dependencies only. These definitions only describe the internal attributes of a component, such as encapsulation, re-usability and so on, from the perspective of the white box method, which does not indicate its external attributes of environmental adaptability, component interaction, and so forth.

2.1.1. The concept of component

According to the history of concept formation and the challenges of component-based development (Crnkovic and Larsson, 2002), we provide the definition of 'component' as follows:

Definition 1 (Atomic component): An atomic component is an integral software unit with input and output interfaces which fulfills the corresponding functions when triggered under special conditions.

Definition 2 (Component): A component is a detachable unit with a certain function for application software (Cox, 2001). It is assembled by $n(n \geq 1)$ units, called atom components, through the interface by following a certain linkage mechanism. It can interact with other components given certain environmental support.

This provides a comprehensive definition for 'component' in terms of a combination of the white-box and black-box methods. 'Interface' defines the service that the component can provide for the environment and other components from the perspective of the external service.

2.1.2. Formal Description of Component Model

According to Definitions 1 and 2, the formal descriptions of an atomic component and component are provided as follows:

Definition 3 (The formal definition of an atomic component):

$c = (c_D, In, Out, Func/Event, Triggers)$ (1)

Where,

c_D is an identifier of atomic component;

In is the input of atomic component, $In = (i_1, i_2, ..., i_n), n \geq 0$; (2)

Out is the output of atomic component, $Out = (o_1, o_2, ..., o_n), n \geq 0$; (3)

Triggers are the trigger conditions for the atomic component, which are hidden knowledge in the system, $Triggers = (tr_1, tr_2, ..., tr_n), n \geq 0$; (4)

Func/Event is the function of atomic component and n-maps from *In* to *Out* under the trigger conditions.

$$Func / Event : \tau_{Triggers}(i_1, i_2 ... i_n) \rightarrow \tau(Out) \quad (5)$$

Definition 4 (Component model): a basic component can be described as follows (Rousseau et al., 2006):

$C=(C_D, Components, Connections, Container)$ (6)

Where,

C_D is a component identifier;

Components is the set of sub-components;

$Components = (com_1, com_2, ..., com_n), n \geq 1$; (7)

Connections indicates the connection method between components, and is divided into three methods: parallel connection, series connection and mixed connection,

$Connections = (Par\ Con \cup Ser\ Con \cup Mix\ Con)$; (8)

Container is a component container.

From the formal description above, we know that a component is essentially a similar onion-skin structure and its 'core' is made up of so-called atomic components. Following the formal interface standard in a certain context, these atomic components are packed layer by layer to constitute a basic component.

2.2. Connection of components

When low-level components are assembled into high-level components, the connection method is divided into parallel connection, series connection and mixed connection.

2.2.1. Parallel Connection

Parallel connection indicates that high-level components are the aggregation of low-level components, in which 'input' is the input of all low-level components, while 'output' is the output of all low-level components. The parallel connection is shown in Fig. 1.

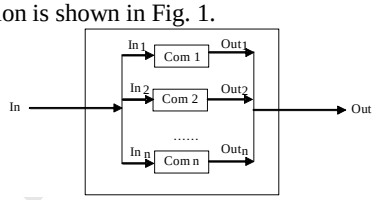

Fig. 1. Parallel connection

The parallel connection relationship is formally described as follows:

Par Con=(In, Out, Triggers) (9)

Where,

$In=(In_1 \times Triggers_1, In_2 \times Triggers_2, \ldots, In_n \times Triggers_n)$ (10)

$Out = (Out_1, Out_2, \ldots, Out_n)$; (11)

$Triggers = (Trigger_1, Trigger_2, \ldots, Trigger_n)$ (12)

$In_1, In_2, \ldots In_n$ are respectively input interfaces of *component 1, component 2,…, component n*;

$Out_1, Out_2, \ldots, Out_n$ are respectively output interfaces of *component 1, component 2,…, component n.*

$Trigger_1, Trigger_2, \ldots, Trigger_n$ are respectively the trigger conditions of *component 1, component 2,…, component n.*

2.2.2. Series Connection

Series connection indicates that the high-level component is composed of a sequence of low-level component nodes in which the input of the first node is the input of the high-level component, while the output of the end nodes is the output of the high-level component. The output of the predecessor nodes changes into the input of the successor node, as shown in Fig. 2.

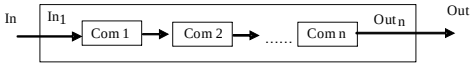

Fig. 2. Series connection

A series connection relationship could be described as follows:

$$Ser\ Con=(In, Out, Triggers) \quad (13)$$

Where,

In is the input of *component 1*, $In = In_1$;

Out is the output of *component n*, $Out = Out_n$;

$Triggers = (Trigger_1, Trigger_2, \ldots, Trigger_n)$; (14)

$In_1, In_2, \ldots In_n$ are respectively input interfaces of *component 1*, *component 2*,…, *component n;*

$Out_1, Out_2, \ldots, Out_n$ are respectively output interfaces of *component 1*, *component 2*,…, *component n*.

Furthermore,

$$In_i = (Out_{i-1} \times Trigger_i) \quad i=2,\ldots,n \quad (15)$$

There are also other mixed connection methods, such as series/parallel connection, parallel/series connection, network, etc. These details are not given here.

3. Tree-like super entity

In this era of information explosion, the best method of organizing information is by DBMS (Database Management System). The data model, as the core and foundation of DBMS, has undergone three phases: hierarchical model (Greenfield and Schneider, 1977), network model and relational model (Cardenas, 1985). The relational database has been the most popular database since the 1970s.

3.1. A problem exists in a traditional tree structure E-R diagram

In relational databases, there are three ways to describe the relationship between entities: one-to-one, one-to-multi, multi-to-multi (Mannino, 2009). An entity-relationship (E-R) exists based on the hierarchical structure; for example, the administrative organization of a university in Fig. 3 illustrates the tree structure.

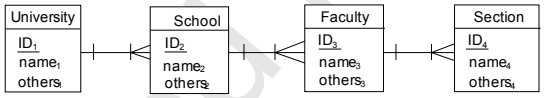

Fig. 3. The traditional organization chart of a university with tree structure

Where, ID_i is an identifier which identifies the *i*-th layer entity uniquely as the primary key, $name_i$ is the main property and $others_i$ is the set of other properties. The *i*-th layer entity can be transferred into the following relational model:

$$enti\ (\underline{ID_i}, name_i, ID_{i-1}, others_i) \quad i=2,..n$$

Where, if i=1, there is no ID_{i-1} property.

Obviously, each entity is presented with a type of tree structure and each tuple of an entity has the same depth in the tree. The model is usually determined by the tuple semantics and data dependencies. The relationship of the relational model will change at different times because the real world constantly changes in different conditions (Yeh et al., 2008). The semantics of tuple are ambiguous; therefore, the integrity constraints of the relationships can easily be challenged. For example, the tree structure of Fig.3 will change when a faculty is directly under the university, and accordingly, the E-R diagram changes from tree to graph, as shown in Fig. 4.

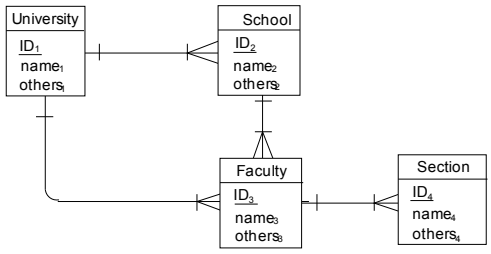

Fig. 4. The changed organization chart of a university with graph structure

Furthermore, when we visit the tree in Fig. 3, a concatenation operation of multi-level entities is required and the computation result is written to the temporary files in the memory. If the depth of the tree is high and the tuple number in each level is large, the computation will consume a large amount of CPU and system memory resources, causing the system to run at reduced efficiency.

3.2. The tree-like super entity and its data structure

Scholars have conducted in-depth studies on how to describe the data structure and related algorithm in Fig. 4, and how to present the data. Freeman and Yin (2005) proposed a new method termed 'Treeview self-organizing maps' (Treeview SOMs) for clustering and organizing text documents by means of a series of independently and automatically created, hierarchical one-dimensional SOMs. Hu et al. (2001) discussed how to express hierarchical data and information by TreeView Control, but did not achieve effective separation between program and data. Liu (2001) achieved the separation but did not unify multi data tables. Ji et al. (2003) proposed a tree-structure storage model of a bill of material (BOM) in MRPII. Two link fields (parent and child) were used in the table to present the relationship between the levels of a product tree, but the related algorithms were not provided.

We therefore adopt the parent representation method of tree structure to store the above multi-level entities with irregular one-to-multi relationships in the form of a database table, and present the concept of tree-like super entity.

Definition 5 (Tree-like super entity): As far as the multi-level entities with one-to-multi relationships are concerned, we extract their keywords and the main attributes with a common semantic, uniform names, and add an attribute to identify the keyword of their parent nodes. This entity is called a tree-like super entity.

For example, in Fig. 5, the keywords (ID_1, ID_2, ..., ID_n) and the primary attributes ($name_1$, $name_2$, ..., $name_n$) are named differently, but their semantics are consistent with each other. They can be normalized and have an attribute added to identify the parent node, to form the following relational model.

sup_ent(ID□name□upID)

Where, *ID* is the keyword, it is used to uniquely identify a node entity in the tree; *name* is the node name; *upID* is the code of the parent node which is the foreign key associated with the parent node. This relational model adopts a static table to realize a tree-like data structure. Fig. 6 presents the tree.

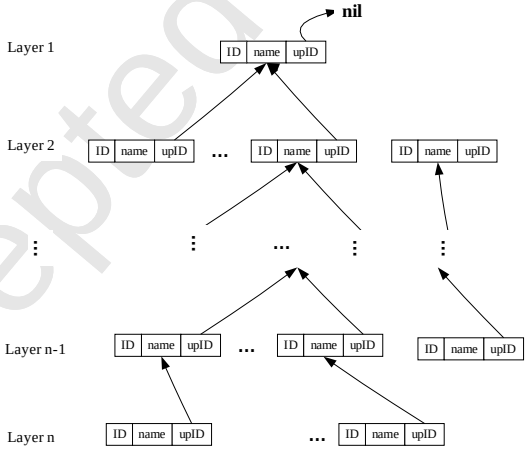

Fig. 5. Data structure of a tree-like super entity

To clearly understand the concept of a tree-like super entity, Fig. 6 provides an example of the data structure and a static storage table for a super entity. The relationship between a super entity and general entities is also presented by combining with other detailed data.

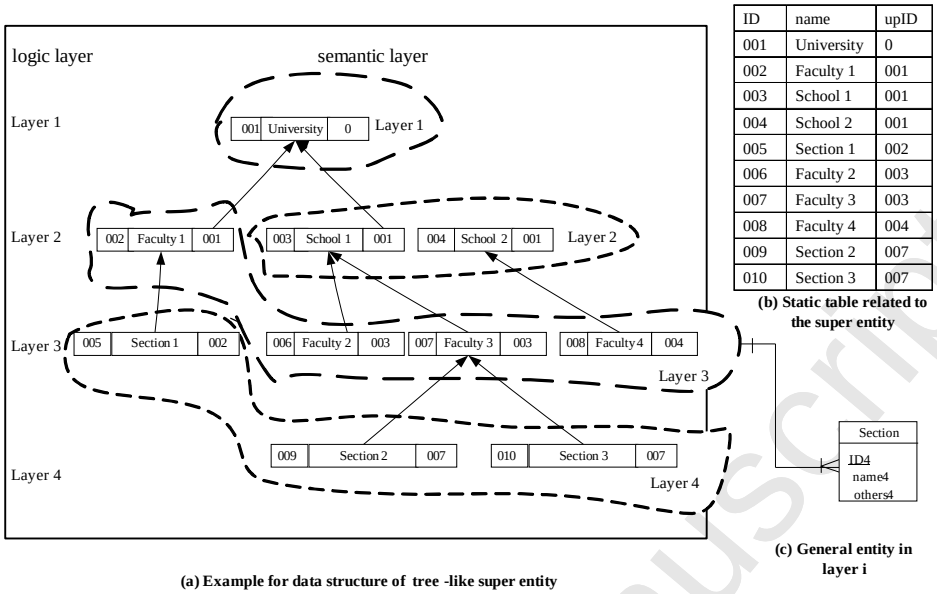

Fig. 6. An example of the data structure of a tree-like super entity

Definition 6 (Logic layer): Logic layer is each layer of the super entity tree, and accordingly, the depth of each node which exists in the logic layer is the one that exists in the tree.

Definition 7 (Semantic layer): Semantic layer is each layer with the same narrow semantic in the super entity tree. If the nodes in the same semantic layer are not in the same logic layer, they should be reduced to the deepest layer of the logic layer in the super entity tree.

From Fig. 6, we know that each node of the tree-like super entity is exactly the tuple in the data table. The depth of each node in the logic layer and the semantic layer is not always the same; in fact, the former is not greater than the latter. Taking Fig. 6 as an example, in the case of the node with ID '006' and the name 'Faculty 2', the depth in the logic layer is the same as that in the semantic layer. However, the node with ID '005' and the name 'Section 1', has a depth of 2 in the logic layer, and less than 3 in the semantic layer.

Definition 8 (Path): Path is the reachable node sequence between any two nodes in the super entity tree (Kruse et al., 1997).

According to the characteristic of the tree, the path from any node to the root is unique in the tree-like super entity.

3.3. Algorithms of tree-like super entity

The tree-like super entity has a dual role. First, it is a data table for physical storage. Second, it is logically expressed as a tree. It performs several operations as a common tree, such as *CreateTree (& T, structure)*, *InsertChild (&T, p, c)*, *SetValue (T, cur_e, values)*, *GetValue (T, cur_e, values)* , *DeleteChild (& T, p)* , *Parent (T, cur_e)* , *Son (T, cur_e).* It also performs two main operations as follows:

(1) *TraverseTree (T, Visit_stra ())*

Initial conditions: tree *T* exists, *Visit_stra* is the traversal strategy.

Description：access each node of the tree once, and only once, according to traversal strategy *Visit_stra.*

There are two traversal strategies: sequential traversal and breadth-first traversal.

- Sequential traversal

Access sequentially the data table corresponding to the super entity tree from beginning to end.

- Breadth-first traversal

Establish an index for the data table corresponding to the super entity tree according to an *upID* attribute, and access the table once.

Only the child node can point to its parent node, but a parent node cannot point to its child nodes so there is no depth-first traversal algorithm.

(2) *Connect (T, Ent)*

Initial conditions: tree *T* exists, *Ent* is an entity.

Description: join the tuple in the *k*-th semantic layer with the entity in the *k+1*-th semantic layer of the tree *T*.

For example, Fig. 6. provides the relationship between the *Faculty* tuple in the 3rd semantic layer and the *Section* entity in the 4th semantic layer.

3.4. Analysis of computational complexity

The most important operation is the *Join-Query* operation in many operations of the tree-like super entity, in which two situations are important. One is to visit from the root to the node in the deepest semantic layer (the *k-1*-th layer), that is, the leaf node. The other is to connect the leaf node with the common entity in the *k*-th layer. In the following, we will discuss the algorithm complexity when comparing the traditional tree structure and the super entity.

Here are the two entity modes:

The entity in the *i*-th layer of the traditional tree-like structure: *enti (IDi, namei, IDi-1, others)*

The tree-like structure super entity: *sup_ent (ID, name, upID)*

Obviously, *IDi-1* and *upID*, *ID* and *IDi* have the same semantic. This is conveniently described as follows: we assume that the length of the field *name* of *super_ent* and the one of the field *namei* of *enti* are the same, so that entity *enti* has one field more than entity *super_ent*; that is, field *others*. We obtain the storage capacity ratio for each tuple between these two entities:

$$c_{enti} / c_{\sup\_ent} = (1+\alpha_i)/1 = 1+\alpha_i \quad (16)$$

We assume that the capacity for each tuple in *super_ent* is 1, then the capacity for each tuple in *enti* is $1+\alpha_i$.

Assume that the number of tuples for the *i-th* layer entity in the traditional tree structure is $n_i(i=1,2,\ldots,k-1)$, the capacity of related data for the *k-th* layer entity is *C*. We will discuss the space complexity of the Cartesian product as follows.

There are three steps for two tables to make a Cartesian product. Firstly, several blocks of tuples from these two tables should be alternately read from the memory according to certain rules. Secondly, the tuples make connections and are written to temporary files. Finally, the temporary files are read to make selection and project operations, according to the query demand. During the whole operation, most of the time and space are spent on the connecting operation, and its complexity is $O(n_i \times n_j)$.

When data tables of the prior *k-1* layers in the traditional tree structure connect with each other and connect with the *k*-th data table, the size of the generated temporary file is:

$$C_{ent} = C_{ent1} \times C_{ent2} \times \ldots \times C_{entk-1} \times C = n_1(1+\alpha_1) \times n_2(1+\alpha_2) \times \ldots \times n_{k-1}(1+\alpha_{k-1})C \quad (17)$$

As far as the tree-like super entity is concerned, we can create an index according to *upID*. After being visited once, the nodes in the *k-1 th* layer connect with the *k-th* data table, and the size of the generated temporary file is:

$$C_{\sup\_ent} = n_1 + n_2 + \ldots + n_{k-1} \times C \quad (18)$$

Obviously, $C_{ent} >> C_{sup\_ent}$ (19)

Suppose that the memory can read and write *m* blocks of data per second, then the time to read and write is:

$$T_{ent} = n_1(1+\alpha_1) \times n_1(1+\alpha_1) \times \ldots \times n_{k-1}(1+\alpha_{k-1})C / m \quad (20)$$

$$T_{\sup\_ent} = (n_1 + n_2 + \ldots + n_{k-1} \times C) / m \quad (21)$$

We can also obtain that: $T_{ent} >> T_{sup\_ent}$ (22)

In the FPPISYR, the background database is SQLServer2000 and the capacity of the whole database is 40G. Under the same hardware and software environment, we designed two methods to create a data table structure: one is the traditional table with one-to-multi relationship, and the other is a tree-like super entity. When retrieving the fifth layer data table, the first method takes 21.23 seconds, while the other takes only 0.562 seconds (Zhang et al., 2007).

4. An editable e-book component-based information presentation method

In a number of highly integrated information systems, text information is usually managed through the database system and other non-text information (such as sound, images, or video) is read and written by special software. There are two ways to integrate these. The first is to perform operations, such as adding, deleting and updating text information from the database, by constructing a powerful system framework, and the non-text information can be read and written by several readers. It is easy to modify all kinds of information in this way. However, there is no consistent thread running through the entire system, the cognition field is independent each other, the effect that the system displays is relatively poor, the object

renders large distance, and it is difficult for an user to quickly acquire the related information from the system. The second option is to manage involved content through e-book – its powerful catalogue features provide the system with a sense of organization and hierarchy in the management of content, and it is easier for users to learn compared to the first method. However, because of the customizable nature of the information, it does not facilitate the expansion of the theme and content, and the interaction is weak. In order to solve these shortcomings, we developed an information presentation method based on an editable e-book component.

The editable e-book component consists of the tree-like structure component, QQ menu component, various kinds of media player components and an *1/K* switch connection component. In this section, we will introduce the composition mechanism of these components by combining the FPPISYR.

4.1. Tree-like structure component

4.1.1. Architecture of tree-like structure component

There are different representations when the tree-like super entity component connects with different components and is encapsulated in different languages and in a different environment. The granularity of components is also different (de Jonge, 2009; Oussalah, 2003). The common representation is a tree-like structure which looks like Windows Explorer in the C/S (Client/Server) mode. As we know, several classic C/S front-end development tools, such as PowerBuilder, Delphi, or VB, support tree-like components and have plenty of events, functions and methods. In the following content, we use PowerBuilder as the programming environment to describe the components. Fig. 7. provides the architecture of the tree-like structure component.

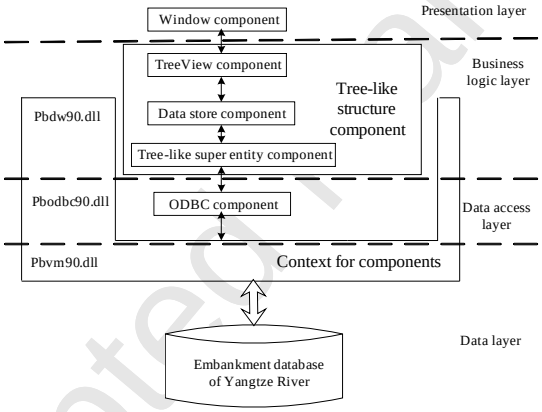

Fig. 7. Architecture of tree-like structure component

From Fig. 7, we can see that the tree-like structure component is composed of three sections from bottom to top: the tree-like super entity component, data store component and tree view component. It connects with the database through the ODBC component downward, and uses the window component as the representation platform upward.

When planning the system tree of FPPISYR, we designed the hierarchical structure of *Middle and lower reaches of Yangtze River- Province- Embankment- Subembankment/Brake*. The following super entity mode is given:

t_diduan (code, name, upcode)

Where, *code* is the primary key which identifies the node in the tree, *name* is the name of the node and *upcode* points to its parent node.

A system tree example is illustrated in Fig. 8. in an interactive way.

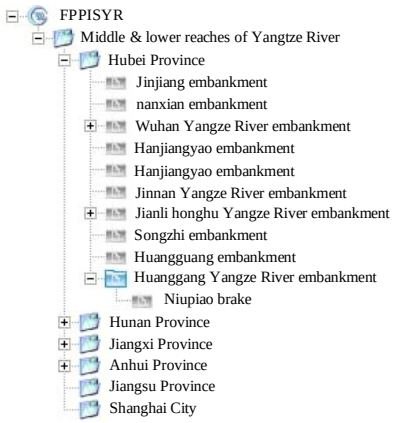

Fig. 8. A real system tree example

4.1.2. Composition mechanism of tree-like structure component

According to the component and its connection model outlined in Section 2, we provide a formal description of the relevant components for the tree-like structure component as follows:

- Tree-like super entity component

c_D = sup_ent001;
In = <data table structure, data of each tuple>;
Out = <data table, stored procedure, trigger>;
Func/Event = <CreateTree, InsertChild, SetValue, GetValue, DeleteChild, Parent, Son, TraverseTree, Connect>.

Description: This component usually exists in the form of a stored procedure or trigger, and is activated by the triggers under certain conditions after loading data though the data tables. The algorithm for the functions is described in Section 3.3. For example, *CreateTree* is used to construct the tree; actually, it is a static table illustrated as Fig. 6 (b). *Connect* is used to connect the leaf nodes in the tree and the related common entities.

- Data store component

c_D = datastore002;
In = <dataobject, object>;
Out = <invisible data view>;
Func/Event = <Create, Destroy, DBError, ItemChanged, RetrieveStart>.

Description: Data store is a type of special component in PowerBuilder which is used to convert the data table in the back-end database to the invisible data view in the foreground application program. It has plenty of events and functions; for example, event *Create* is used to create the data store, event *Destroy* is used to destroy the data store, event *DBError* gives an error message when extracted data is wrong, event *ItemChanged* is triggered when the data item is changed and event *RetrieveStart* is triggered when retrieving data.

- Tree view component

c_D = treeview003;
In = <datastore>;
Out = <tree>;
Func/Event = <Constructor, Destructor, Itemexpanded, SelectionChanged, Clicked, RightClicked, DoubleClicked>.

Description: The tree view component is also a component in PowerBuilder which is used to fetch the data in the data store item by item and write to the nodes of tree view. It mainly has the events *Constructor, Destructor, Itemexpanded, SelectionChanged, Clicked, RightClicked, DoubleClicked*, and so on. These events are triggered when the user executes the relevant operations.

- Tree-like structure component

C=(C_D, Components, Connections, Container);
C_D = tree01;
Components = <Sup_ent001, datastroe002, treeview003>;
Connections = <Series Connection>;

Container = <Window>.

Description: The tree-like structure component is a composite component which combines the three components in series and presents to the user by using the window as the container and display mode. In this way, a user only creates a tree-like super entity data table in the background and uses it as an entrance to the tree-like structure component to output the visible system tree.

4.2. QQ Menu component

As a popular online chatting tool, QQ is welcomed by the majority of users in China (Gao and Cao, 2010) because its menu is designed with icons and text, more information is provided and its functions are strongly interactive. Nowadays, more and more user-friendly presentation modes of information systems are welcomed by users, and the popularization of large screen and high-resolution display means that the QQ menu is used as a trend for the menu (Chen, 2005).

Fig. 9 provides the structure of the QQ menu, which consists of three levels of assembled components. The components from the inside to the outside are as follows: second level menu item components (LeafMenu), first level menu item components (SonMenu) and the menu components (Menu). There are four atomic components: Text Label components (TextLabel), PictureLabel components (PictureLabel), UpButton components (UpButton) and DownButton components (DownButton). All models of these components are described as follows:

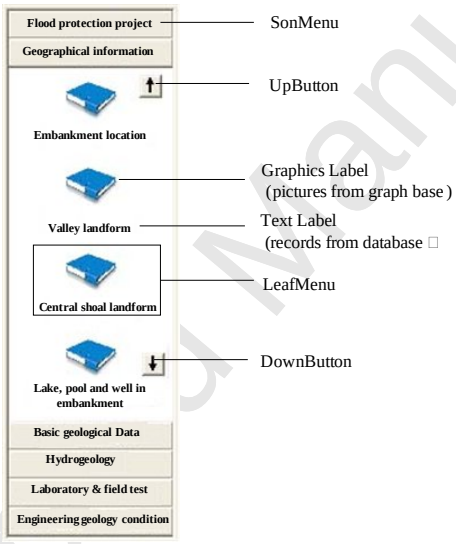

Fig. 9. QQ menu

- Text label component (TextLabel)

c_D = TextLabel001;

In = <Database table rows>;

Out = <A window representing data objects>;

Func/Event = <Clicked, RightClicked, DoubleClicked>.

Description: This component is an atomic component, which mainly includes *click*, *double click*, *right-click* and other events. These events are triggered when the user performs the relevant operations.

- Picture label component (PictureLabel)

c_D = PictureLabel 002;

In = <Picture in graph base>;

Out = <A window representing data objects>;

Func/Event = <Clicked, RightClicked, DoubleClicked>.

Description: This component is an atomic component, which corresponds to the text label component.

- Second level menu item component (LeafMenu)

C_D = LeafMenu 003;

Components = <TextLabel, PictureLabel>;

Connections = <Parallel Connection>;

Container = <UserObject>.

Description: The component consists of text labels and picture labels in parallel, and it is packaged by the user's object in PowerBuilder.

- First level menu item component (SonMenu)

C_D = SonMenu 004;
Components = <LeafMenu, UpButton, DownButton>;
Connections = <Parallel Connection>;
Container = <DataWindow>.
Description: The component consists of second-level menu item components and UpButton, DownButton components in parallel, and it is packaged by a datawindow in PowerBuilder.

- Menu item component (Menu)

C_D = Menu 005;
Components = <SonMenu>;
Connections = <Parallel Connection>;
Container = <Window>.
Description: The component consists of several first-level menu item components in parallel.

4.3. 1/K switch connection component

In order to switch seamlessly between different kinds of media in editable e-books, a 1/K switch connection component is proposed between the previous component and several next components, and looks like the single-pole multi-throw switch in a physical circuit. In the component, the high level component is the aggregation of a previous component and several next components. The input of the component is the input of the previous component, and the output is the output of one of the several next components, as shown in Fig. 10.

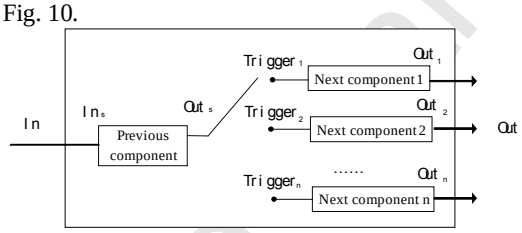

Fig. 10. 1/K switch connection component

The connection component can be formally described as follows:
1/k_switcher_Con = (In, Out, Triggers)
Where, $In = In_s$
$Triggers = (Trigger_1, Trigger_2, \ldots, Trigger_n)$

$$Trigger_i = \begin{cases} 1 & \text{if previous component connects to next component}_i \\ 0 & \text{if not} \end{cases} \quad (23)$$

$$\sum_{i=1}^{n} Trigger_i = 1 \quad (24)$$

$$Out = \sum_{i=1}^{n} func_i(Out_s \times trigger_i) \quad (25)$$

Where, In_s is input of the system tree.
Out_s is output of the system tree.
$Out_1, Out_2, \ldots, Out_n$ are the outputs of $component_1, component_2, \ldots, component_n$, respectively.
$Trigger_1, Trigger_2, \ldots, Trigger_n$ are the triggers of $component_1, component_2, \ldots, component_n$, respectively.

4.4 Architecture of the editable e-book

The editable e-book component takes a system tree component as the root, the 1/K switch connection component as the branch, and a number of browser and player components as the nodes. In this way, a tree-like architecture is formed as shown in Fig. 11.

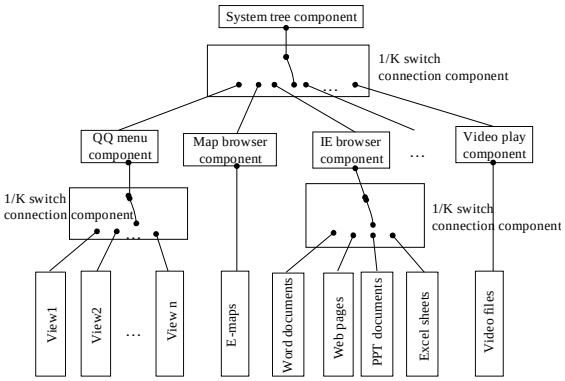

Fig. 11. Architecture of editable e-book

The tree formed by an editable e-book has the following properties:

Property 1: The path from the previous node to the next node is one-way mapping.

Property 2: The root can only reach one leaf node at a given time; that is, there is only one path between all the paths from the root to each leaf node that can be activated.

Property 3: 1 / k switch connection component is usually coupled with its previous node.

4.5. Multi-style information presentation mode based on editable e-book component

Taking the FPPISYR as an example, the system interface is divided into two parts: the system tree and workspace. The system tree on the left side is made up of the tree-like structure component; it controls the flexible switching and intelligent presentation of the various media information on the right side, through the 1/K switch connection component. Figs. 12, 13 and 14 respectively illustrate multi-style presentation modes with database information, video information and image information under the same system tree.

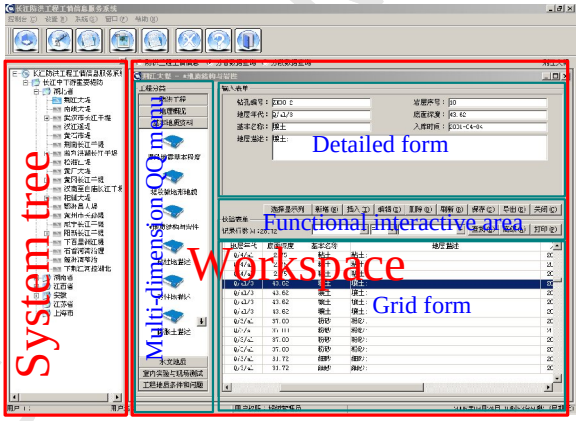

Fig. 12. Database information presentation mode

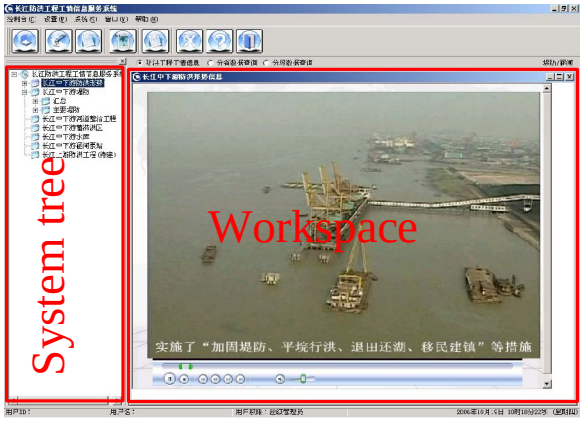

Fig. 13. Video information presentation mode

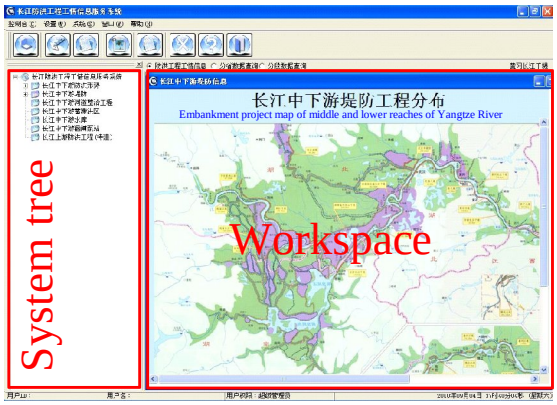

Fig. 14. Image information presentation mode

A user can also edit the related node information in the system tree according to the right details, which cannot be achieved with an ordinary e-book. Fig. 15 provides an example interface for editing the system tree.

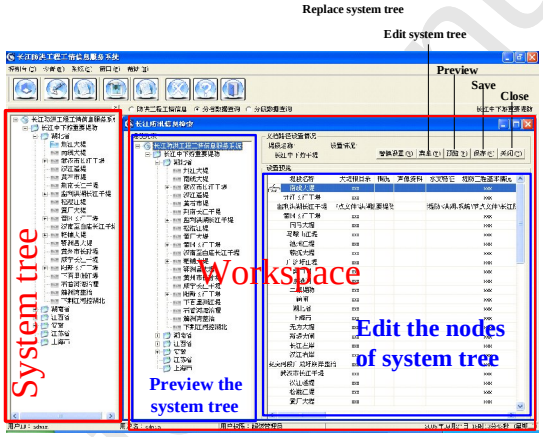

Fig. 15. Edit the system tree

When the user clicks the Edit system tree button, he or she can add, delete and edit the information of nodes in the tree and then save them in the background database by clicking the Save button. If the user clicks the Replace system tree button, the system will pop-up a window and ask the user to select a data table with a tree-like structure to replace the current system tree. When clicking the Preview button, users can preview the system tree in the middle section.

4.6. Comparison between the editable e-book and other information presentation methods

From the example interfaces mentioned above, we know that the editable e-book provides a new representation method to display information from different aspects, depth and coverage. Here we design a questionnaire survey with twenty questions and related example information presentation interfaces by combining the evaluation model for human-computer interaction given by Wang (2006), Hu et al. (1999) and Speier (2006). The questionnaire concerns document, hypertext, button, e-book and other combined information presentation methods, and evaluates the presentation effect from four aspects: Coverage, Depth, Dimension and Interaction. A 5-point scale is adopted in the questionnaire: 5-excellent, 4- very good, 3- good, 2-medium, 1- poor, 0- very poor. In the survey, a hundred and twenty-one questionnaires are delivered, and a hundred and two effective questionnaires are received. Table 1 illustrates the evaluation weights for several general combined information presentation methods within a 95% confidence interval.

Table 1

Evaluation weights for several general combined information presentation methods

Information presentation method		Coverage	Depth	Dimension	Interaction	Description
Text	Common editable document	0-4	0-4	0-4	3	focus on one aspect of converge, depth, dimension or interaction according to the context
	Linkable hypertext	1	4	1	1	tend to be confused
	Menu+ embedded form+ buttons	1	2	5	3	tend to be confused
	Menu+ embedded form+ buttons	5	0	5	5	no sense of hierarchy
	Tree structure + form+ buttons	3	5	0	5	cannot reflect multi-aspect of the object
	Menu+ tree structure + form+ buttons	3	5	5	5	reflect multi-level and multi-aspect of the object comprehensively
Non-text	Web page	0-5	0-5	0-5	0-5	emphasize differently according to the context of the web page, less security, low access speed
	E-book	4	3	4	0	a strong overall sense, less interaction
	Map browser	5	1	1	1	a regional sense, less sense of hierarchy
Editable e-book		4 -5	4 -5	4 - 5	5	reflect different aspects, levels, and coverage of the object

From the comparison in Table 1, we see that the editable e-book provides a satisfactory method of presenting information. In the method, two-dimensional tables, documents, e-maps, videos and other media are used to represent the object in detail, the system tree is utilized to illustrate the different levels of the object, the QQ menu is adopted to describe the object from a different profile and the interaction between the user and object is performed in the functional interactive area.

5. Conclusions

Aiming to solve the problem of information presentation in the same information system platform for multi-structure and multi-source information, this study proposes a tree-like super entity based multi-layer entity organization mode, and encapsulates the tree-like super entity with the component mode we have designed. An editable e-book component-based information presentation method was developed, which has been successfully implemented in the FPPISYR. The main contributions of this study can be summarized as follows:

(1) A tree-like super entity was proposed to solve the ambiguity problem whereby the semantic of some tuples change when constructing multi-layer one-to-multi entities. The data structure and related algorithms for the tree-like super entity were given and the architecture of the tree-like structure component was also illustrated. We compared the tree-like super entity with the traditional tree structure entity and found that the former had the advantage of time complexity in *Join-Query* operation and space complexity when it was saved as a data table.

(2) An editable e-book component was designed, consisting of a tree-like structure component, QQ menu component, 1/K switch connection and other components. The component can present multi-dimensional and multi-level information in an interactive way by combining with table form, video, web page, editable document, and other presentation methods. We developed and deployed the editable e-book component in real life integrated information systems and achieved good results.

Acknowledgements

This research is supported by the National Natural Science Foundation under Grant 90924026 from the China Government, by the Australia Research Council (ARC) under Discovery Grant DP0557154, by the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry, and by the Research Program of Humanities and Social Sciences, State Education Ministry, China.

References

Bruce E., 2004. Thinking in Java (second edn.), Prentice Hall, Upper Saddle River, NJ.

Cardenas A., 1985. Data Base Management Systems (second edn.), Allyn and Bacon, Boston, MA.

Chang, Y., Lin, C., Lee, Y., Lai, H., 2006. Optimized PDA orientation and screen layout for Chinese vocabulary learning by young

children. Displays 27(2), 73-79.

Chen, Z., 2005. Dynamic design of QQ panel menu by Delphi. Journal of Youjiang Teachers College for Nationalities Guangxi 18(6), 67-69.

Combi, C., Oliboni, B., Rossato, R., 2005. Merging multimedia presentations and semistructured temporal data: a graph-based model and its application to clinical information. Artificial Intelligence in Medicine 34(2), 89-112.

Coppus, G., Strashok, A., 1995. Manufacturing information systems for the process industry: responding to the plant management challenges of the 90's. ISA Transactions 34(2), 119-132.

Cox, P.T., Song, B., 2001. A formal model for component-based software. Proc of 2001 IEEE Symposium on Visual/Multimedia Approaches to Programming and Software Engineering, Stresa, Italy, 122-124.

Crnkovic, I., Larsson, M., 2002. Challenges of component-based development. Journal of Systems and Software 61(3), 201-212.

D'Souza, D.F., Wills, A.C., 1997. Objects, Components, and Frameworks with UML: the Catalysis Approach. Addison-Wesley, Boston, MA.

de Jonge, M., 2009. Developing product lines with third-party components. Electronic Notes in Theoretical Computer Science 238(5), 63-80.

Ettredge, M., Richardson, V.J., Scholz, S., 2001. The presentation of financial information at corporate Web sites. International Journal of Accounting Information Systems 2(3), 149-168.

Feng, S., Li, L.X., Duan, Z.G., Zhang, J.L., 2007. Assessing the impacts of South-to-North Water Transfer Project with decision support systems. Decision Support Systems 42(4), 1989-2003.

Freeman, R.T., Yin, H., 2005. Tree view self-organisation of web content. Neurocomputing 63, 415-446.

Gao, Y., Cao, T., 2010. Memory forensics for QQ from a live system. Journal of Computers 5(4), 541-548.

Greenfield, P.M., Schneider, L., 1977. Building a tree structure: the development of hierarchical complexity and interrupted strategies in children's construction activity. Developmental Psychology 13(4), 299-313.

Guoli, J., Daxin, G., Tsui, F., 2003. Analysis and implementation of the BOM of a tree-type structure in MRPII. Journal of Materials Processing Technology 139(1-3), 535-538.

Han, X., Jin, Y., 2009. Research progress on display technology of electronic book. China Printing and Packaging Study 1(4), 8-13.

Hu, J., Gu, Q., Wang, X., 2001. Application of tree view component in management information system. Computer Development and Application 14(8), 31-32.

Hu, P., Ma, P., Chau, P., 1999. Evaluation of user interface designs for information retrieval systems: a computer-based experiment. Decision Support Systems 27(1-2), 125-143.

Jiang, T., Su, B., Hartmann, H., 2007. Temporal and spatial trends of precipitation and river flow in the Yangtze River basin, 1961-2000. Geomorphology 85(3-4), 143-154.

Kang, Y., Wang, M., Lin, R., 2009. Usability evaluation of e-books. Displays 30(2), 49-52.

Khalifa, M., Kwok R.C., 1999. Remote learning technologies: effectiveness of hypertext and GSS. Decision Support Systems 26(3), 195- 207.

Klitzman, R., Zolovska, B., Folberth, W., 2009. Preimplantation genetic diagnosis on in vitro fertilization clinic websites: presentations of risks, benefits and other information. Fertility and Sterility 92(4), 1276-1283.

Ko, C., 2009. RFID-based building maintenance system. Automation in Construction 18(3), 275-284.

Kruse, R. L., Tondo, C. L., Leung, B. P., 1997. Data structures & program design in C (second edn.), Prentice Hall, Upper Saddle River, NJ.

Kundu, S., Mukherjee, J., Majumdar, A.K., Majumdar, B., Sekhar Ray, S., 2007. Algorithms and heuristics for efficient medical information display in PDA. Computers in Biology and Medicine 37(9), 1272-1282.

Leach, R.J., 1997. Software Reuse: Method, Models, and Costs. McGraw Hill, New York.

Li, L., Lu, X., Chen, Z., 2007. River channel change during the last 50 years in the middle Yangtze River, the Jianli reach. Geomorphology 85(3-4), 185-196.

Liu, H., Hu, J., Zheng, P., 2000. A development method based on architecture of component-based application system and its application. Journal of Wuhan University of Hydraulic and Electric Engineering 33(4), 83-85.

Liu, S., 2001. The application of tree structure in production management information system. Information Technology(9), 11-13.

Lu, J., Ma, J., Zhang, G., 2007. Warning message generation by information filtering technique. International Journal of Nuclear Knowledge Management 2(4), 435-448.

Ma, J., Zhang, G., Lu, J., 2010. A state-based knowledge representation approach for information logical inconsistency detection in warning systems. Knowledge-Based Systems 23(2), 125-131.

Mannino, M.V., 2009. Database Design, Application Development, and Administration. McGraw-Hill, New York.

Manthou, V., Vlachopoulou, M., 2001. Bar-code technology for inventory and marketing management systems: a model for its development and implementation. International Journal of Production Economics 71(1-3), 157-164.

Manvi, S.S., Venkataram, P., 2005. An intelligent product-information presentation in e-commerce. Electronic Commerce Research and Applications 4(3), 220-239.

Niederhauser, D.S., Reynolds, R.E., Salmen, D.J., Skolmoski, P. 2000. The influence of cognitive load on learning from hypertext. Journal of Educational Computing Research 23(3), 237-255.

Oussalah, M., 2003. Reuse in KBS: a components approach. Expert Systems with Applications 24(2), 173-181.

Quentin-Baxter, M., 1998. Hypermedia learning environments limit access to information. Computer Networks and ISDN Systems 30(17), 587-590.

Ravichandran, R., Banerjee, H., 1994. Support for information systems usage in banks. International Journal of Information Management 14(1), 5-12.

Rousseau, C., Bellik, Y., Vernier, F., 2006. A framework for the intelligent multimodal presentation of information. Signal Processing 86(12), 3696-3713.

Sforna, M., 2000. Data mining in a power company customer database. Electric Power Systems Research 55(3), 201-209.

Shelburne, W.A., 2009. E-book usage in an academic library: user attitudes and behaviors. Library Collections, Acquisitions, and Technical Services 33(2-3), 59-72.

Speier, C., 2006. The influence of information presentation formats on complex task decision-making performance. International Journal of Human-Computer Studies 64(11), 1115-1131.

Sun, H., Wu, Y., Wei, J., 2009. Research on CORBA pluggable protocols. Computer Knowledge and Technology 5(27), 7783-7787.

Sun Microsystems, 1997. JavaBeans for Java Studio: Architecture and API white paper.

Taylor, M.A.P., Woolley, J.E., Zito, R., Integration of the global positioning system and geographical information systems for traffic congestion studies. Transportation Research Part C: Emerging Technologies 8(1-6), 257-285.

Terpsiadou, M.H., Economides, A.A., 2009. The use of information systems in the Greek public financial services: the case of TAXIS. Government Information Quarterly 26(3), 468-476.

Tsamenyi, M., Cullen, J., 2006. Changes in accounting and financial information system in a Spanish electricity company: a new institutional theory analysis. Management Accounting Research 17(4), 409-432.

Waema, T.M., Walsham, G., 1990. Information systems strategy formation in a developing country bank. Technological Forecasting and Social Change 38(4), 393-407.

Wang, Y., Forgionne G., 2006. A decision-theoretic approach to the evaluation of information retrieval systems. Information Processing & Management 42(4), 863-874.

Wu, Y., Zhang, J., Liu, S.M., Zhang, Z.F., Yao, Q.Z., Hong, G.H., Cooper, L., 2007. Sources and distribution of carbon within the Yangtze River system. Estuarine, Coastal and Shelf Science 71(1-2), 13-25.

Xu, K., Chen, Z., Zhao, Y., Wang, Z., Zhang, J., Hayashi, S., Murakami, S., Watanabe, M., 2005. Simulated sediment flux during 1998 big-flood of the Yangtze (Changjiang) River, China. Journal of Hydrology 313(3-4), 221-233.

Yang, Y., Jing, Z., Gao, T., Wang, H., 2007. Multi-sources information fusion algorithm in airborne detection systems. Journal of Systems Engineering and Electronics 18(1), 171-176.

Yeh, D., Li, Y., Chu, W., 2008. Extracting entity-relationship diagram from a table-based legacy database. Journal of Systems and Software 81(5), 764-771.

Zhang, R., Chen, D., Shi, L., Zhou, S., Zhang, B., 2007. Research and realization of embankment information system of Yangtze River based on three modes. Journal of Wuhan University of Technology (Transportation Science & Engineering) 31(1), 27-30.

Ruijun Zhang is currently an Associate Professor and tutor of graduate students in the Information Management Department, School of Management, Wuhan University of Science and Technology (WUST), China. He received the Bachelor's degree in Computer Science from the Northern Jiaotong University in 1994, and Master's and Ph.D. degrees in Computer Application Technology from Wuhan University of Technology in 2002 and 2007 respectively. His research interests include information system engineering, decision support systems and intelligent information systems. He has published two textbooks and over 20 papers. Contact him at zrjun@wust.edu.cn.

Jie Lu is a Professor and Director of the Decision Systems and e-Service Intelligence Research Laboratory in the Faculty of Engineering and Information Technology at the University of Technology Sydney (UTS), Australia. She received her PhD from Curtin University of Technology in 2000. Her main research interests lie in the area of intelligent information systems, decision making modeling, decision support system tools, uncertain information processing, e-Government, and e-Service intelligence and personalization. She has published five research books and more than 250 papers in refereed journals and conference proceedings. She has won four Australian Research Council (ARC) discovery grants and many other research grants. She

served as Editor-in-Chief for Knowledge-Based Systems and as a guest editor of special issues for six international journals, and has also delivered four keynote speeches at international conferences. Contact her at jie.lu@uts.edu.au.

Guangquan Zhang is an Associate Professor in the Faculty of Engineering and Information Technology at the University of Technology Sydney (UTS), Australia. He has a PhD in Applied Mathematics from Curtin University of Technology, Australia. From 1979 to 1997, he was a Lecturer, Associate Professor and Professor in the Department of Mathematics, Hebei University, China. His main research interests lie in the area of multi-objective, bilevel and group decision making, decision support system tools, fuzzy measure, fuzzy optimization and uncertain information processing. He has published four monographs, four reference books and over 250 papers including more than 120 refereed journal articles. He has won four Australian Research Council (ARC) discovery grants and many other research grants. He has served, and continues to serve, as a guest editor of special issues for three international journals. Contact him at Guangquan.Zhang@uts.edu.au.