Research on Structure understanding of Complex-Table in Form Applications

X.C. Wang, H. Ma, D.M. Liang, Z. Du
Department of Educational Technology
Capital Normal University
Beijing, China

***Abstract*—Form application is an important means in modern enterprise management, and its user interfaces pursue paper effect more and more. Complex-table is a common expressive way in Chinese forms. In the existing form applications development tools, complex-table can only be implemented by directly coding instead of visual design. In fact, it is not difficult to implement complex-table by visual design, but it is inconvenient to manipulate its inner data from the viewpoint of component when the complex-table has been designed. Hence, by surveying and analyzing form applications, a complex-table structure understanding method is put forward. Utilizing this method, the logical relationship between the cells in a complex-table is ascertained, and the business rule implementation and data operation become convenient. Thus, the development of form applications is accelerated, and the developer's burden is reduced.**

Keywords-enterprise management; form application; complex-table; table structure understanding; table structure model; hypothesis operation

I. INTRODUCTION

With the increasing demand about systematic and scientific management in modern enterprise, the form, as one of the important means in enterprise management, has been played a significant role. On the one hand, the form makes user interface more systematic and standardized; on the other hand the form can cover the whole management process in enterprise, making the management in the order. The form application [1], along with business logic management, promotes information sharing and improves working efficiency, so the form application has the important commercial value [2]. With the development of human-computer interaction techniques, the end-user of form applications, have the increasing demand on the paper effect [3]. Complex-table [4] is a Chinese-style form, and often appears in various documents. Therefore, the developer should be able to easily and quickly develop software with complex-table. However, the current software development tools and application development toolkits doesn't include the complex-table component such as visual basic and java. Thus, developers who want to implement complex-table in the user interface can only be achieved through direct coding mode, which not only increases the burden on developers, but also reduces the efficiency of development. Therefore, we should develop a complex-table component to solve the above problems.

Complex-table component should be divided into two states: design time and run time. At design time, the main focus is on how to facilitate the design of the complex-table appearance, and set label area and entry area, these functions can be realized by tabulation with the tools such as Microsoft Word and Excel. In run time, the data in the complex-table is processed so that the corresponding business logic can be implemented. The intelligent manipulation of the data in the complex-table from the viewpoint of the logical structure will make the development easier, however the complex-table should be designed in a natural way, and the designer of the table should focus on the description of business complex-table, and does not consider the logic relationship between the cells, therefore the complex-table structure should be understood by the development tools. Table structure to understand [5] is not a new emerging research field, which is researched and used by many field such as data extraction from Web documents and scanned image, information retrieval, document understanding and Web mining [5,6,7,8]. In these applications, the complex-table is stored in the system in feature beforehand. When a complex-table instance is inputted to the system, the system will identify its type and structure. The designer of the development tools cannot know the appearance of complex-table which is tabulated by the user of development tools, so this method can not be used to understand complex-table component.

In this paper, by investigation and analysis about the logical structure of the application form and the complex-table, a method is proposed to understand the structure of the complex-table, which not only makes complex-table easy to design, but also operate its internal data from the logic view, so that we can develop a stand-alone component which is applied to software development tools.

In the rest of the paper, section 2 proposes a complex-table structure understanding method. Section 3 describes the application of the method. Section 4 concludes the paper.

II. COMPLEX-TABLE STRUCTURE UNDERSTANDING

Complex-table structure understanding is defined as the process that the complex-table layout structure is mapped to the logical structure. The complex-table structure understanding in this paper utilizes rule-based pattern combination method to recognize complex-table structure, and then verify the correctness. The whole process begin from the cell, the then produce structural hypothesis by combination rules [9, 10], the error structure hypothesis will removed and rectified by hypothesis reasoning and user.

For more effectively support structure hypothesis generation and reasoning, the complex-table structure understanding is defined as:

U = (Box, Rule, Operator, Result)

Where, Box is the set of the cells in a complex-table. Rule is the predicate set of structure hypothesis, and these rules will produce the structure hypothesis from cell to a complex-table. Operator is a set of operators about hypothesis operation, where Operator = { ○ , |}, both of which are the binary operator, and the priority of the operator" ○ " is over the "|". The operator " ○ " represents the hypothesis can be combined to a new hypothesis; the operator "|" indicates an optional relationship between hypothesis. Result is the result of hypothesis computing. In the following part, we will utilize this reasoning system to describe the process of complex-table structure understanding.

A. *The Basic of Complex-Table Structure Understanding*

In the process of complex-table structure understanding, the logical information of the cell and the adjacent relationship between the cells is needed at the same time, which constitutes the initial conditions and the basis of reasoning about the complex-table structure understanding.

1) *Information about cell:* In the complex-table, the cell is the basic unit of a logical structure, and is also the minimum geometry unit, which has the basic physical information and logical information. The physical information refers to the position of the top-left corner, and its height and width about a cell. The logical information refers to the type of cell functions: label, entry and ULC (Up-Left Cell), as shown in Figure1.

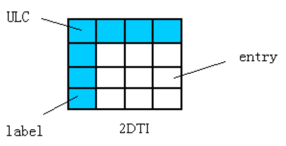


FIGURE I. INFORMATION ABOUT CELL.

The function of the label cell is to explain the entry next to the label cell; the function of entry cell is to receive user's input. The ULC cell is the upper-left cell in 2DTI (two-Dimensional Table-Item) item. In the process of structure hypothesis generation, the relationship between the label cell and entry cell, as well as between the label cells, is firstly considered, ULC is not considered until 2DTI hypothesis is generated.

2) *Adjacency relationship between the cells:* In the complex-table, each cell has four sides. Each side is shared with another cell, or share with the complex-table borders. Adjacency Relationship between the cells can be summarized as three cases (Figure 2). Figure 2 shows the relationship in horizontal direction, denoted with "H". The adjacency relationship between cells in vertical direction and horizontal direction is similar, denoted with "V".

All adjacency relationship operators shown in Figure 2 is asymmetric, the exchange of the location of variable will change the meaning. For example, Such as: $a\xleftrightarrow{H}b$ denote that cell *b* is at right side, adjacent with the same height in horizontal direction with *a*. $b\xleftrightarrow{H}a$ denote that cell *b* is at left side, adjacent with the same height in horizontal direction with *a*.

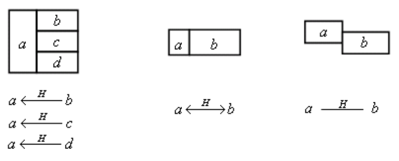


FIGURE II. ADJACENCY RELATIONSHIP BETWEEN CELLS IN A COMPLEX-TABLE.

B. *Combination Rule*

Complex-table structure hypothesis is generated by the utilization of logical and physical information. First, the basic structure hypothesis is produced. The basic structure hypothesis is divided into two classes: the type Ⅰ hypotheses and the type Ⅱ hypothesis (Figure 3 is their examples). Type Ⅰ structure hypothesis is divided into type Ⅰ-1 structure hypothesis and type Ⅰ-N structure hypothesis (N≥2), and both can be expressed in a unified form as follows:

For the label cell a and entry cell $e_1, e_2 \cdots e_n$ (n⩾2), if there is

$a\xleftrightarrow{x}e_1\xleftrightarrow{x}e_2\cdots\xleftrightarrow{x}e_n \wedge \exists[e_j[e_n\xleftrightarrow{x}e_j)]$ where $x\in\{V,H\}$, n ⩾1, type Ⅰ structure hypothesis $B=(a\ ,\ e_1,e_2\cdots e_n)$ can be generated.

Type Ⅱ structure hypothesis are used to represent multi-level label mode, which can be produced as follows:

For the label cells in a complex-table $a,d_1,d_2\cdots d_n$ (n⩾2), if there is

$a\xleftarrow{x}d_1\wedge a\xleftarrow{x}d_2\cdots\wedge a\xleftarrow{x}d_n\wedge\neg(\exists d_j(a\quad_{\text{x}}\ d_j))$ where $x\in\{V,H\}$, n >1, type Ⅱ structure hypothesis $G=(a,d_1,d_2\cdots d_n)$ can be generated.

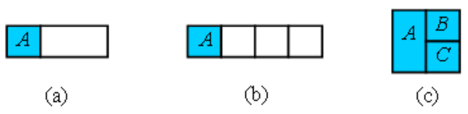


FIGURE III. THE EXAMPLE OF BASIC STRUCTURE HYPOTHESIS. BLUE CELL IS THE LABEL, AND BLANK CELL IS THE ENTRY.

After the basic structure hypothesis is generated, the high-level complex-table structure hypotheses will continue to further produced. Vertical-wise VRI (Variable-length Rectangle Item) structure hypothesis can be generated as follows: For type 1-N vertical-wise structure hypothesis $B_1^V,B_2^V,\cdots B_{\text{n}}^V$ (n ⩾ 2), if there is $B_1^V.e_j\xleftrightarrow{H}B_2^V.e_j\xleftrightarrow{H}B_3^V.e_j\cdots\xleftrightarrow{H}B_n^V.e_j$ for all

$j\in\{1,2,\min(B_1^V.E.getCount(),B_2^V.E.getCount(),\cdots B_n^V.E.getCount())\}$

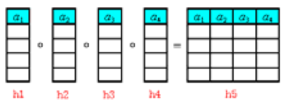
, structure hypnosis $VVRI(B_1^V,B_2^V,\cdots B_n^V)=B_1^V\circ B_2^V\circ\cdots B_n^V$ can be generated, where $B.E.getCount()$ is denoted as the number of entry cell in hypothesis *B*. Figure4.a is an example of VVRI hypothesis generation.

Horizontal-wise VRI structure hypothesis can be produced as follows:

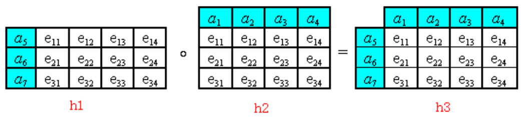
For type 1-N horizontal-wise structure hypothesis $B_1^H, B_2^H, \cdots B_m^H$ (n > 2), if there is $B_1^H.e_i \xleftrightarrow{V} B_2^H.e_i \xleftrightarrow{V} B_3^H.e_i \cdots \xleftrightarrow{V} B_m^H.e_i$ for all $i \in \{1,2, \min(B_1^H.E.getCount(), B_2^H.E.getCount(), \cdots B_m^H.E.getCount())\}$ structure hypnosis $HVRI(B_1^H, B_2^H, \cdots B_m^H) = B_1^H \circ B_2^H \circ \cdots B_m^H$ can be produced.

2DTI structure hypothesis can be produced as follows: For structure hypothesis *VVRI* and *HVRI*, if $B_1^V.e_1$ and $B_1^H.e_1$ is the same cell, the hypothesis $2DTI = VVRI \circ HVRI$ can be generated. The graphic example of the computing process is shown in Figure 4.b.

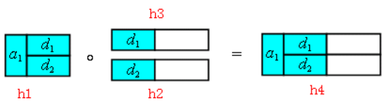
Tree item structure hypothesis is generated from the basic structure hypothesis. Set 1-level tree item structure hypothesis as $T^1 = (a,e)$ which is type Ⅰ-1 structure hypothesis, ***N**-level* tree item structure hypothesis T^N $(N \geqslant 2)$ can be described as: $T^N = (G, T_1, T_2, \cdots T_k)$, where $N \geqslant 2$, $K \geqslant 2$, *G* is a type Ⅱ structure hypothesis. When T^N $(N \geqslant 2)$ is produced, It should meet the following conditions: for each *d* in *G*, there is only one T_i $(1 \leqslant i \leqslant k)$ satisfied $d = T_i.a$, and for each T_i $(1 \leqslant i \leqslant k)$, there is only one *G.d* satisfied $T_i.a = G.d$, and there is at least one T_j $(1 \leqslant j \leqslant k)$ which is N-level tree items hypothesis. Its generation process is as shown in Figure4.c.



(a) The examples of VVRI structure hypothesis generation. h1, h2, h3 and h4 are type Ⅰ-N structure hypotheses, h5 is the VVRI structure hypothesis.



(b) The examples of 2DTI structure hypothesis generation. h1 is HVRI structure hypotheses and h2 are VVRI structure hypotheses, h3 is the 2DTI hypothesis.



(C) The examples of 2-level tree item structure hypothesis generation. h1 is type Ⅱ structure hypotheses, and h2 and h3 are type Ⅰ-1 structure hypotheses, h4 is the2-level tree item structure hypothesis.

FIGURE IV. THE EXAMPLE OF COMBINATION RULE FOR HIGH-LEVEL STRUCTURE HYPOTHESIS

C. *Hypothesis Ambiguity Operation*

The example of complex-table structure understanding in Figure4 is typical. Because the structure hypothesis is produced by the geometric relationship between the cells using the corresponding logical relationship, and each cell forms hypothesis in both horizontal and vertical directions, and semantic information of the text in cell is not utilized during the process of hypothesis generation, so that ambiguity of complex-table structure understanding is inevitable. For example: in Figure5.a, both (a_1, e_1) and (a_1, e_2) are satisfied with the conditions of the hypothesis generation, but a_1, is not allowed to label e_1 and e_2 at the same time, therefore it should be determined which cell a_1 label.

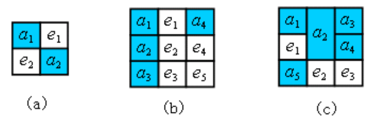


FIGURE V. THE EXAMPLE OF AMBIGUOUS HYPOTHESIS. ALL OF (A), (B) AND (C) ARE THE PART OF COMPLEX-TABLE. HYPOTHESIS (A1, E1) AND (A1, E2) ARE AMBIGUOUS IN (A). HYPOTHESIS (A2, E2, E4) AND (A4, E4, E5) ARE AMBIGUOUS IN (B). HYPOTHESIS (A5, E2, E3), (A2, E2) AND (A4, E3) ARE AMBIGUOUS IN (C).

To eliminate the ambiguity comprehension, two hypotheses operation are introduced: the intersection computing and optional computing. Hypothesis intersection is defined as follows: for the hypothesis D_1, D_2, if there is a cell *b* which belongs to D_1 and D_2 at the same time, D_1 and D_2 is intersecting. From the viewpoint of the layout structure, this situation will exist between the hypothesis in horizontal direction and the hypothesis in vertical direction. This hypothesis will be computed by the rules of the intersection hypothesis to eliminate incorrect hypotheses.

Optional hypothesis is defined as follows: for the hypothesis D_1, D_2, if there is a cell *b*, which belong to D_1 and D_2at the same time, then D_1, D_2 is optional, that is to say: $D_1 \mid D_2$. This suggests a cell is shared by several candidate hypotheses, and these candidate hypotheses remain after the intersection operation is completed. In fact, a cell must belong to the only hypotheses. However, the ambiguity can not be removed by hypothesis intersection operation, and user's arbitration must be used to resolve this problem. This problem should not occur in a well-designed user interface.

The following example illustrates the process of intersection operation of the structure hypothesis. For instance, after the combination rule is applied, an intermediate result of complex-table structure understanding is described as *Result* = $M_1 \circ M_2 \circ M_3 \circ M_4 \circ \cdots M_n$, where M_1, M_2, ... M_n is the results which is produced by the combination rules. If there is intersection between them, further computing is required. Suppose hypotheses M_1 and M_2 are intersecting. According to the feature that one cell belong to two hypotheses at most in a complex-table, suppose cell *e* is belong to M_1 and M_2 at the same time, thus $M_1 \circ M_2 = M_1 \circ M_{21} \mid M_{11} \circ M_2$, where M_{21} is the hypothesis M_2 which remove cell *e*, M_{11} is the hypothesis M_1 which remove cell *e*. Thus, M_{21} or M_{11} is not right hypothesis any more so that we can conclude $M_1 \circ M_{21}$ or $M_{11} \circ M_2$ is the result.

III. PRACTICAL APPLICATION

The complex-table structure understanding method which is proposed in the paper is applied into the complex-table component. In the design-time, the complex-table which is similar with appearance in Microsoft Word is designed by digit-pen, and then complex-table understanding algorithm is applied so that we can know about the corresponding relationship between cells, so that it can be easy to manipulate input data from a logical perspective. Figure 6 shows a result of complex-table structure understanding. Each box surrounded by red line is a logic block. With the results, it is possible to facilitate the manipulation of the input data. For example, suppose the instance name of the complex-table component is TeachingPlan, the program statement such as TeachingPlan. GetData ("goal.idea") or TeachingTable.SetData (" goal.idea ", 15) can be used to get or set the corresponding cell, which can greatly simplify the programming, especially for VRI items and 2DTI term operation.

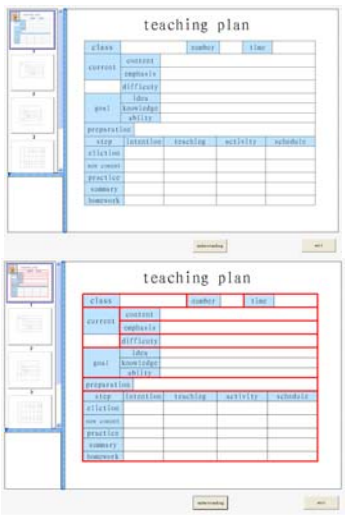


FIGURE VI. THE LEFT-SIDE FIGURE IS A COMPLEX-TABLE IN WHICH BLUE CELL IS THE LABEL AND BLANK CELL IS THE ENTRY. THE RIGHT SIDE COMPLEX-TABLE IS THE RESULT OF STRUCTURE UNDERSTANDING. EACH LOGICAL BLOCK SURROUNDED BY RED LINE IS A STRUCTURE HYPOTHESIS.

The method of Complex-table structure understanding in this paper is put forward for Complex-table in the user interface. if too many ambiguity appears in Complex-table structure understanding computing, we can conclude that the design of complex-table is often unscientific, which can increase user's cognitive load, even result users cannot correctly find the cell which the content will be filled to, so that to decrease the usability of the software. We should suggest the layout of complex-table should be reconsidered.

IV. CONCLUSION

Forms application is a scientific modern enterprise management tools, and complex-table is an indispensable component to achieve the pen-paper effect of the user interface in forms application. In this paper, a method for understanding the complex-table structure is proposed, which make form-designer can achieve a pen-paper effect of user interface, but also be able to easily manipulate their internal data from a logical view. This method will lay the foundation for the realization of intelligent business rules, accelerate the development of form application development, and reduce the burden on developers.

ACKNOWLEDGEMENTS

The research is supported by Science and Technology Plan of Beijing Municipal Education Commission under Grant No. KM201410028018.

REFERENCES

[1] J.Q. Xiong, Y.T. Fan, D.X. Teng, et al. An Online Synchronization Input Method Based on Paper Forms and Electronics Ones. *Journal of Computer-Aided Design & Computer graphics*, 24(9), pp.1125-1133, 2012.

[2] IBM Workplace form. http://www-142.ibm.com/software/workplace/ products/product5.nsf/wdocs/formshome

[3] Y.T. Fan, D.X. Teng, C. X. Ma, , et al. A Model Driven Development Method for Pen-Based Form Interface Software. *Journal of Computer Research and Development*, 49(12), pp. 2671-2685, 2012.

[4] Akira Amano, Naoki Asada. Graph Grammar Based Analysis System of Complex Table Form Document. In: Proceedings of the International Conference on Document Analysis and Recognition, pp. 916 -920, 2003.

[5] Jacek Siciarek. Semantics Driven Table Understanding in Born-Digital Documents, *Advances in Intelligent Systems and Computing*, Vol.233, pp. 153-160, 2014.

[6] T. Hassan. Towards a common evaluation strategy for table structure recognition algorithms. In Proceedings of DocEng, 2010.

[7] Guoliang Li. A Human-Machine Method for Web Table Understanding, *Lecture Notes in Computer Science Volume 7923*, pp.179-189, 2013.

[8] HE Nielson, WA Barrett. Consensus-Based Table-Form Recognition, In: Proceedings of the International Conference on Document Analysis and Recognition, Edinburgh (Scotland), pp. 906-910, 2003.

[9] R. Zanibbi, D. Blostein, J.R. Cordy. A Survey of Table Recognition: Models, Observations, Transformations, and Inferences, *International Journal of Document Analysis and Recognition*, 7(1), pp. 1-16, 2004.

[10] R. Zanibbi, D. Blostein, and J.R. Cordy. Historical Recall and Precision: Summarizing Generated Hypotheses, In: Proceedings of the International Conference on Document Analysis and Recognition, pp. 202-206, 2005.