TR-88-19

FORMALIZATION ON VISUAL PROGRAMMING LANGUAGES

Formalization on Visual Programming Languages

D. H. Lu, M. S. Hwu, and K. Y. Cheng

Department of Computer Science and Information Engineering
National Taiwan University
and
Institute of Information Science Academia
Taipei, Taiwan, R. O. C.

ABSTRACT

The programming technique that utilizes interactive devices for programmer- and nonprogrammer-professional experts to develop their own application programs has been seen in many up-to-date personal computer systems and workstations recently. The distinguished feature of this programming technique is that the designers need only to express their expertise in some sort of visual forms and flows in programming. Nowadays, programming in visual becomes even popular in many applications such as office automation, computer-aided instruction, database interface, computer animation, computer graphics, pattern recognition and cognition, Chinese computer design, morphological language understanding, and artifical intelligence. The purpose of this paper is aiming to a study on this programming technique from formal language theoretical point of view.

I. INTRODUCTION

The programming technique that utilizes certain interactive devices such as screen cursor, joystick, mouse, touch screen, etc. and certain screen operations such as menu, window, form, diagram, icon, etc. has gained widespread popularity in personal computer systems and workstations recently. According to different application purposes, there are a few visual objects which can be described in terms of form-based, diagram-based or icon-based representations. They have been used extensively in a variety of applications such as office information[5, 28, 30, 31, 32], computer-aided instruction[22, 23, 39], computer-aided design[26], database interface[3, 16, 29], computer graphics[9, 10], image processing[12, 33, 36], scene understanding[37], ideographic language understanding[35], ..., etc. This reveals a great portion of human knowledge representation can be expressed visually.

Instresting enough, most visual applications are based on the visualization of patterns. For examples, an office information system is a visualization of office pattern by using forms as visual objects and their handlings as object relations, a Chinese input is a visualization of Chinese character pattern by using radicals as visual objects and spatial relations as object relations, a flow-chart or a diagram is a visualization of graph, ..., etc. From user's point of view, visual programming languages may be viewed as pattern description languages. On the other hand, from system's point of view, the realization of a pattern description is the inference of a grammar that generates the pattern description language. In this paper, we give visual languages a formal definition and discuss some properties of the languages and their programming styles.

II. VISUAL LANGUAGE DEFINITION

In this section, we shall give the visual language a formal definition which is based on the notion of labeled graphs.

[Definition 2.1] The structured graph $S_c(S)$ is a labeled directed graph, i.e., $S_c(S) = (V, E)$, where V and E are sets of labeled nodes and labeled directed edges, respectively.

Each node of a structured graph represents a concept and each directed edge represents a relation. The structured graph is a visualization of a pattern description which may be described by a visual language. Therefore, the visual language may be defined as follows.

[Definition 2.2] A visual language L_V is defined as

$$L_V = \{x \mid x \text{ is a graph representation which can be generated by a visual grammar } G_V\},$$

and the visual grammar G_V is a four tuple

$$G_V = (V_N, V_T, P, S)$$

where V_N is a set of nonterminals (composite visual objects), V_T is a set of terminals (primitive visual objects), P is a set of production rules, and S is a set of starting symbols.

[Definition 2.3] The language L_V is called a t-based visual language if the representation of visual objects used in the structured graph is t-based.

There are many possible t-based visual languages which have been used in various applications. Among them, form-based[21, 22, 23, 28, 29, 30, 31, 32], diagram-based[1, 2, 8, 11, 34, 38], icon-based[6, 12, 15, 33] visual languages are the most often seen. However, in practice, more than one t-based visual languages may be used simultaneously in the development of an application program[25].

Let S be an application program with program segments $\{S_i\}$ then the structured graph $S_c(S)$ is the visualization of S and $S_c(S_i)$ the visualization of each program segment S_i. Using a visual programming language L_V to describe each $S_c(S_i)$, a programming synthesizer infers a grammar G_i or a structure equivalent to G_i and finally a grammar $G = G_1 \cup G_2 \cup \ldots \cup G_n$ and $G_i \cap G_j = \emptyset$ for $i \neq j$ to complete the realization of the application program S. The result of this realization is a generated program which can be run on a machine, i.e., the task of the programming synthesizer is to perform grammartical inference and interpret inference results into programs[13, 20].

The process that utilizes visual languages to construct an application program such that the construction result is acceptable by a machine is called visual programming. The environment under which the users can freely develop their application programs by the visual programming technique is a visual programming environment. However, in this paper we shall not go to the detail description of the visual programming and its environment concerning each mentioned system, instead, we simply describe them from the above defined visual language theoretical point of view.

III. WEB GRAMMAR IN ICON APPLICATIONS

The web grammar proposed by Pfaltz and Rosenfeld[14] can be used to generate sentences whose visualization are directed graphs with symbols at their nodes. In visual application, it is particular useful for the realization of icon applications.

[Definition 3.1] A web grammar is a four tuple

$$G = (V_N, V_T, P, S)$$

where V_N is a set of nonterminals(composite icons), V_T a set of terminals(primitive icons), S a set of initial webs, and P a set of web productions. A web production is defined as

$$\alpha \rightarrow \beta, \ \gamma$$

where α and β are webs and γ is an embedding of β .

Example 3.1. Each Chinese character is a web whose formation can be generated by a web grammar as shown below (Fig. 3.1).

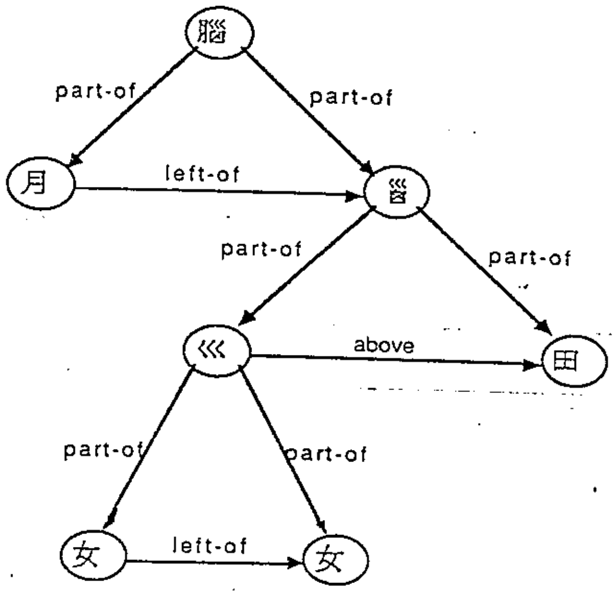


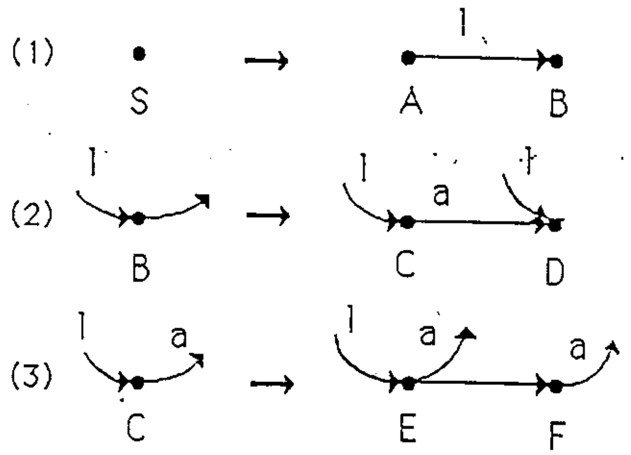
Fig. 3.1(a). The decomposition graph of character 腦 .

VN = { 腦 , 甾 , 巛 }

VT = { 月 , 女 , 田 }

S = { 腦 }

P :



S = 腦

A = 月

B = 甾

C = 巛

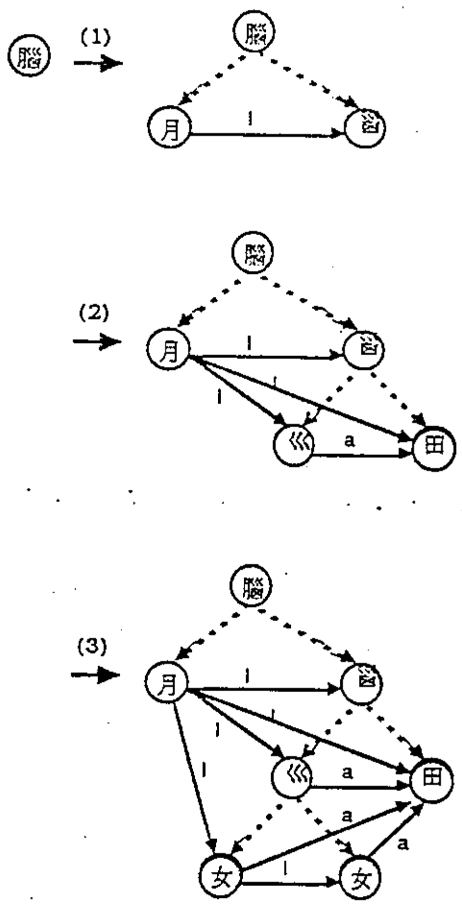
D = 田

E = 女

F = 女

Fig. 3.1(b). The web grammar that generates character 腦 .

Example 3.2. The act of an icon-based Chinese input is an iconic language. In Tsang-Chi method, 腦 is visualized as a sentence 月 女 女 田 (a sequence of primitive icons). As shown in Fig. 3.1, each initial web (a Chinese character) can be expressed in terms of a derivation diagram in the derivation of the sentence according to some meta rules provided in the input method. It can be shown that the derivation can be generated by a context-free attributed grammar[15], i.e., each production rule of the grammar shown in Fig. 3.2 is associated with a semantic rule (defined in the predefined meta rules).



Semantic rules used

Extc(腦) = Extc(月) · Extc(𡿺)

(· denodes a concatenate operator)

Extc(𡿺) = Extc(巛) · Extc(田)

Extc(巛) = Extc(女) · Extc(女)

Fig. 3.2. The derivation steps of character 腦 .

From this observation, we see an immediate visual application is in the design of a Chinese input simulator[24]. Now, the virtual language for the simulator is the visualization of the structured graph for each Chinese character and the evaluation is based on the realization of the visual grammar that generates sentences of the iconic language.

IV. TREE GRAMMAR IN FORM-BASED APPLICATIONS

Tree grammar and the corresponding tree automata have been studied and applied to syntactic pattern recognition[18, 19]. As will be seen in the following, most form-based visual applications are a visualization of trees.

[Definition 4.1] Let r(a) be the out-degree (or rank) of the labeled node a. Then an expansive tree grammar G_t = (V, r, P, S) has production of the form

$$\begin{tikzcd} X_0 \arrow[r] & x \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & & \text{or} & X_0 \arrow[r] & x \\ X_1 & X_2 \cdots & X_{r(x)} & & & \end{tikzcd}$$

where $x \in V_T$ and $X_0, X_1, \ldots, X_{r(x)} \in V_N$.

A typical form-based visual application is the QBE(Query-By-Example) proposed by Zloof[29]. When a user uses QBE to query, update, define, or control a database, he/she uses some skeleton tables to describe the result of what he wants. The commands in each skeleton table and the table itself can be regarded as an application program. The structured graph of the application program has a tree structure with some leaf nodes to represent fields of skeleton table(each field is a primitive visual object), a root node to represent the name of a skeleton table, as well as directed edges to represent the descendant relation between the root node and leaf nodes.

Example 4.1. A QBE query as shown in Fig. 4.1 can be expressed as follows(Fig. 4.2).

TYPE	ITEM	COLOR	SIZE
		P. <u>WHITE</u>	

Fig. 4.1. A simple query.

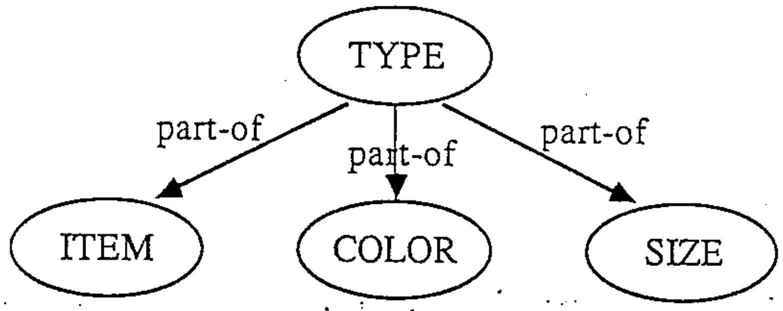


Fig. 4.2. Structured graph of table 'TYPE'.

The visual grammar which generates this structured graph is the following context-free attributed tree grammar[17].

$V_N = \{ S, A, B, C \}$

$V_T = \{ TYPE, ITEM, COLOR, SIZE \}$

$r(TYPE) = 3, \qquad r(ITEM) = r(COLOR) = r(SIZE) = 0$,

and

P :

Syntax rules	Semantic rules
$S \rightarrow \begin{tikzcd}[column sep=small] & TYPE \arrow[dl, no head] \arrow[d, no head] \arrow[dr, no head] & \\ A & B & C \end{tikzcd}$	$v(S) = AND(v(A), v(B), v(C))$
$A \rightarrow ITEM$	$v(A) = (v(ITEM), p(ITEM))$
$B \rightarrow COLOR$	$v(B) = (v(COLOR), p(COLOR))$
$C \rightarrow SIZE$	$v(C) = (v(SIZE), p(SIZE))$

$p(ITEM) = p(SIZE) = FALSE, p(COLOR) = TRUE$

where the function p indicates whether a field is going to be displayed, function v represents the field value of a tuple and AND(v(A), v(B), v(C)) represents a tuple satisfying all constraints of v(A), v(B), and v(C). As can be seen, QBE application programs can be regarded as sentences of a visual language generated by a context-free attributed tree grammar. Also, a QBE system can be regarded as a tree automaton that accepts QBE application programs.

Other form-based visual programming systems like QBE/OBE[4,28], FORMAL[30, 31], and VIPS[22, 23] which have been used to develop application programs in office information, database management, and computer-aided instruction can be similarly regarded as tree automata for the recognition (or acceptance) of user-defined application programs.

V. DIAGRAM-BASED VISUAL LANGUAGES

Diagram-based visual programming languages use the diagrams such as flowchart[2], Nassi-Shneiderman diagram[11, 27], state transition diagram[34], or Petri net[1, 38] as visual objects to develop user-defined application programs. Usually, diagram-based programming is a procedure-wise style which requires users possessed with at least some programming knowledge. For example, Fig. 5.1 shows a flowchart of a binary search program which can be regarded as a structured graph with the flowchart symbols as nodes and the control flow as edges.

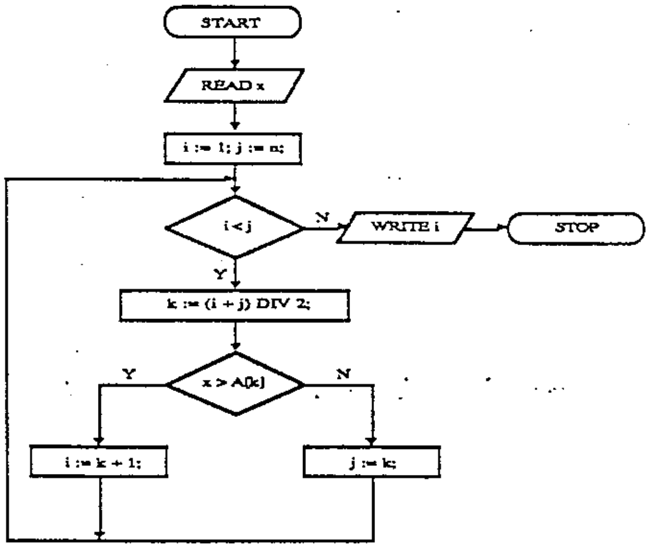


Figure 5.1. A flowchart.

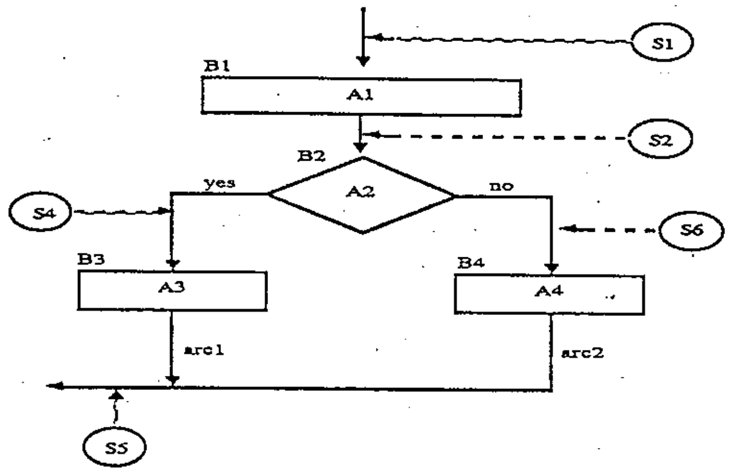


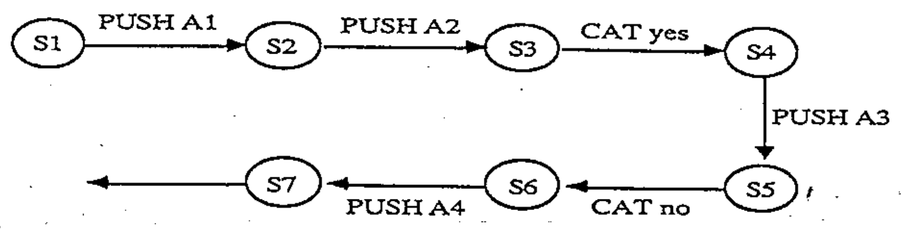
Figure 5.2. Part of Fig. 5.1.

The realization of the structured graph shown in Fig. 5.2(part of Fig. 5.1) can be expressed as the following context-sensitive grammar.

$$
\begin{array}{lcl}
 & \vdots & \\
\alpha B1 & \rightarrow & \alpha x1\ B2 \\
\beta B2 & \rightarrow & \beta x2\ \text{yes}B3\ \text{no}B4 \\
\text{yes}B3 & \rightarrow & \text{yes}x3\ \text{arc1} \\
\text{no}B4 & \rightarrow & \text{no}x4\ \text{arc2} \\
 & \vdots &
\end{array}
$$

where yes, no, arc1, and arc2 $\in V_T$; B1, B2, B3, and B4 $\in V_N$; α, β, *x1* , *x2* , *x3* , and *x4* $\in V_T^*$.

The context-sensitive language generated by the above grammar can be accepted by a transition network shown as follws



where Ai (i = 1, 2, 3, and 4) represent the automata which accept program statements.

Other diagrams such as Nassi-Shneiderman diagram, state transition diagram, and Petri-net can be viewed as flow diagram with slight defferent representations. Hence, the visualization of the diagram is the directed graph and the realization of the directed graph is the context-sensitive graph grammar.

VI. CONCLUSION

In this paper, a formalization on visual programming languages is presented. We show that an icon-based visualization may be realized by a context-free attributed web grammar, a form-based by an attributed tree grammar, and diagram-based by a context-sensitive graph grammar.

REFERENCE

[1] Alfs T. Berztiss, "Specification of Visual Representations of Petri Nets," *Proceedings of IEEE Computer Society Workshop on Visual Languages*, pp. 225-233 , Aug. 1987.

[2] B. E. J. Clark and S. K. Robinson, "A Graphically Interacting Program Monitor," *The Computer Journal*, Vol. 26, No. 3, pp. 235-238, Mar. 1983.

[3] C. F. Herot, "Spatial Management of Data," *ACM Trans. on Database Systems*, Vol. 5, No. 4, pp. 493-514, 1980.

[4] C. J. Date, "An Introduction to Database Systems," third edition, *Addison-Wesley Publishing company*, 1981.

[5] Dawei Luo and S. B. Yao, "Form Operation by Example - A Language for Office Information Processing," *Proceedings of SIGMOD Conference*, pp. 212-223, June 1981.

[6] D. H. Lu, M. S. Hwu and K. Y. CHeng, "Ideographical Formation of Icon," *Proceedings of National Computer Symposium*, pp. 138-144 , Dec. 1987.

[7] D. H. Lu, "Notes on Visual and Iconic languages," M. S. Thesis, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, R.O.C., June 1988.

[8] E. P. Glinert and S. L. Tanimoto, "Pict: an Interactive Graphical Programming Environment," *IEEE Computer Magazine*, pp. 7-25, Nov. 1984.

[9] F. S. Montalvo, "Diagram Understanding: Associating Symbolic Descriptions with Images," *Proceedings of IEEE Computer Society Workshop on Visual Languages*, pp. 4-11, June 1986.

[10] G. F. McCleary, Jr., "An Effective Graphic "Vocabulary" ," *IEEE Computer Graphics and Applications*, pp. 46-53, March/April 1983.

[11] I. Nassi and B. Shneiderman, "Flow Chart Techniques for Structured Programming," *SIGPLAN Notices*, Vol. 8, No. 8, pp. 12-26, Aug. 1973.

[12] I. Yoshimoto, N. Monden, M. Hirakawa, M. Tanaka, and T. Ichikawa, "Interactive Iconic Programming Facility in HI-VISUAL," *Proceedings of IEEE Computer Society Workshop on Visual Languages*, pp. 34-41, June 1986.

[13] J. M. Brayer and K. S. Fu, "Some multidimensional grammar inference methods," *in Pattern Recognition and Artifical Intelligence*, (C. C. Chen, ed.) Academic Press, New York, 1976.

[14] J. L. Pfaltz and A. Rosenfeld, "Web grammars," Proc. First Int. Joint Conf. Artif. Intell. May 1969, Washington, D. C., pp.606-619.

[15] K. C. Cheng, "Automatic Evaluation on Radical-based Chinese Character Input Methods," M. S. Thesis, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan, R.O.C., June 1988.

[16] K. Hoehne, "The ISQL Language - A unified Tool for Managing Images and Non-Images Data Management System," *in Visual Languages*, edited by S. K. Chang etc., Plenum Pub. Co. 1986.

[17] K. S. Fu, "Syntactic Pattern Recognition and Applications," *Prentice-Hall, Inc. Englewood Cliffs*, N. J., 1982.

[18] K. S. Fu, "Tree Languages and syntactic pattern recognition," in *Pattern Recognition and Artificial Intelligence*, (C. H. Chen, ed.). Academic Press, New York, 1976.

[19] K. S. Fu and B. L. BHARGAVA, "Tree systems for syntactic pattern recognition," *IEEE*

Trans. Comput. C-25, 262-74 (mar. 1976).

[20] K. S. Fu and T. L. Booth, "Grammatical Inference: Introduction and Survey -- Part I," *IEEE Trans. on System, Man and Cybernetics*, Vol. SMC-5, No. 1, pp. 95-111, 1975.

[21] K. Sugihara, T. Kikuno, N. Yoshida and M. Takayama, "An Approach to the Design of a Form Language," *Proceedings of IEEE Computer Society Workshop on Visual Languages*, pp. 171-176, Dec. 1984.

[22] K. Y. Cheng, C. C. Hsu, I. P. Lin, M. C. Lu, and M. S. Hwu, "VIPS: A Visual Programming Synthesizer," *Proceedings of IEEE Computer Society Workshop on Visual Languages*, pp. 92-98, June 1986.

[23] K. Y. Cheng, M. S. Hwu, and C. C. Hsu, "An Extended Visual Programming Synthesizer for Computer Aided Instruction Applications," *Proceedings of IEEE Computer Society Workshop on Visual Languages*, pp. 147-160, Aug. 1987.

[24] K. Y. Cheng, M. S. Hwu, and S. Y. Hsu, "Evaluation of Reduced-Radical Chinese Input Methods from Parsing Visual Routines," Techinique Report, Inst. of Information Science, Academia Sinica, 1988.

[25] K. Y. Cheng, M. S. Hwu, and S. Y. Hsu, "Some Remarks on Developing Application Programs in Visual," *Proceedings of IEEE Computer Society Workshop on Visual Languages*, to be published.

[26] M. Beretta, P. Mussio, and M. Protti, "Icons: Interpretation and Use," *Proceedings of IEEE Computer Society Workshop on Visual Languages*, pp. 149-158, June 1986.

[27] M. C. Pong and N. Ng, "PIGS -- A System for Programming with Interactive Graphics Support," *Software-Practice and Experience*, Vol. 13, pp. 847-855, 1983.

[28] M. M. Zloof, "QBE/OBE: A Language for Office and Business Automation," *IEEE Computer*, Vol. 14, No. 5, pp. 13-22, 1981.

[29] M. M. Zloof, "Query-by-Example: A Data Base Language," *IBM System Journal*, Vol. 16, No. 4, pp. 324-343, 1977.

[30] N. C. Shu, "A Forms-oriented and Visual-directed application development system for non-programmers," *Proceedings of IEEE Computer Society Workshop on Visual Languages*, pp. 162-170, Dec. 1984.

[31] N. C. Shu, "FORMAL: A Form-Oriented, Visual-Directed Application Development System," *IEEE Computer*, Vol. 18, No. 8, pp. 38-49, 1985.

[32] N. C. Shu, V. Y. Lum, F. C. Tung, and C. L. Chang, "Specification of Forms Processing and Business Procedures for Office Automation," *IEEE Transactions on Software Engineering*, Vol. SE-8, No. 5, pp. 499-512, Sep. 1982.

[33] N. Monden, I. Yoshimoto, M. Hirakawa, M. Tanaka, and T. Ichikawa, "HI-VISUAL: A Language Supporting Visual Interaction in Programming," *Proceedings of IEEE Computer Society Workshop on Visual Languages*, pp. 199-205, Dec. 1984.

[34] R. J. K. Jacob, "A State Transition Diagram Language of Visual Programming," *IEEE Computer*, Vol. 18, No. 8, pp. 51-59, Aug. 1985.

[35] S. K. Chang, "Icon Semantics -- A Formal Approach to Icon System Design," *Proceedings of International Conference on Chinese Computing*, Singapore, Aug. 1986.

[36] S. K. Chang, E. Jungert, S. Levialdi, G. Tortora, and T. Ichikawa, "An Image Processing Language with Icon-Assisted Navigation," *IEEE Transactions on Software Engineering*, pp. 811-819, Aug. 1985.

[37] S. K. Chang, Q. Y. Shi, and C. W. Yan, "Iconic Indexing by 2D Strings," *Proceedings of IEEE Computer Society Workshop on Visual Languages*, pp. 12-21, June 1986.

[38] Tadashi Ae and Reiji Aibara, "A Rapid Prototyping of Real-Time Software Using Petri Nets," *Proceedings of IEEE Computer Workshop on Visual Languages*, pp. 234-241, Aug. 1987.

[39] W. Finzer and L. Gould, "Programming By Rehearsal," *Byte*, Vol. 9, No. 6, pp. 187-210, June 1984.