A Construction Method of Expert Systems in an Integrated Environment

Hui Chen

Center for Information Science, Kokushikan University
4-28-1 Setagaya, Setagaya-ku, Tokyo, 154-8515, Japan
Tel: +81-3-5481-3220, Fax: +81-3-5481-3227, E-mail: chen@kokushikan.ac.jp

Abstract

This paper introduces a method of constructing expert systems in an integrated environment for automatic software design. This integrated environment may be applicable from top-level system architecture design, data flow diagram design down to flow chart and coding. The system is integrated with three CASE tools, FSD (Functional Structure Diagram), DFD (Data Flow Diagram) and structured chart PAD (Problem Analysis Diagram), and respective expert systems with automatic design capability by reusing past design. The construction way of these expert systems is based on systematic acquisition of design knowledge stemmed from a systematic design work process of well-matured developers. The design knowledge is automatically acquired from respective documents and stored in the respective knowledge bases. By reusing it, a similar software system may be designed automatically. In order to develop these expert systems in a short period, these design knowledge is expressed by the unified frame structure, functions of the expert system units are partitioned mono-functions and then standardized components. As a result, the design cost of an expert system can be reduced to standard work procedures. Another feature of this paper is to introduce the integrated environment for automatic software design. This system features an essentially zero start-up cost for automatic design resulting in substantial saving of design man-hours in the design life cycle, and the expected increase in software productivity after enough design experiences are accumulated.

Keywords:

Expert system; design knowledge; Standardization; Automatic software design; Intelligent CASE tool

1. Introduction

The software industry has been facing increasing demand for software, and automatic design has been regarded as the final solution [8]. But the penetration to the industry is very slow. One reason for this might be that only 100% fully automatic designs have been considered and cost effective partial automatic design share not yet been introduced. The second seems to be the fact that the start-cost as well as the development cost for a highly automated design system is quite large. Our policy is to enable a cost effective but not complete automatic design system, in a bottom up manner. The first system, named Intelligent CASE tool [2] automates programming phase by reusing past designs in structured charts. It has evolved to Integrated Intelligent CASE tool [3] that widens the application upward to data flow design prior to programming. We have been developing an Integrated Environment putting emphasis on the architecture that enables standardized design process applicable from earlier phase systems design down to coding. These intelligent tools are consisted of some commercial drawing tools and expert systems, so a main problem of developing them is how to reduce development cost and construct these expert systems rapidly.

This paper introduces a method of constructing the expert systems in an integrated environment for automatic software design. The way is based on systematic acquisition of design knowledge stemmed from a systematic design work process of well-matured software development organizations. The design knowledge is expressed by the unified frame structure, so the design knowledge is easily automatically acquired from respective documents. Functions of the expert system units are partitioned mono-functions and then standardized components. Design is a transformation from the input to the output, and there is a main data flow. Is resembles a distributed control system which corresponds to a hierarchy agents as Minsky pointed out [9]. This type of expert systems should be structured hierarchically. As a result of systematic knowledge expressed by data flow and control flow, the design of expert systems is reduced to the same procedure for ordinary software design. Another feature of this paper is to introduce the integrated environment for automatic software design. It may be applicable from top-level system architecture design, data flow diagram design down to flow chart and coding.

2. Basic Assumption

This automatic design method may be regarded as a reuse of design documents in fragments from the viewpoint of

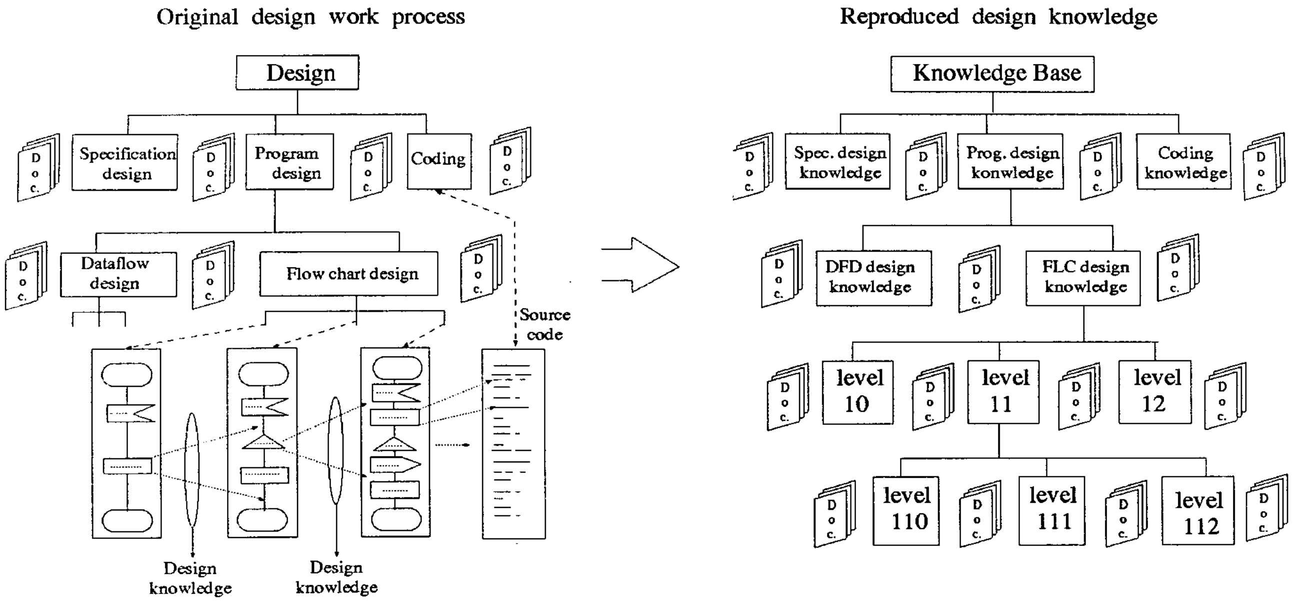


Figure 1 - Hierarchical design work process and the reproduced design knowledge

conventional Software Engineering. As the ***target expert*** for the design, an excellent software development organization with high maturity is taken. Their high maturity has been attained by accumulating experiences in each field and making improvements successively for more than ten years [5]. As a result, they have not only excellent technical knowledge but also a highly, uniform and solid process featuring correct expression.

The development starts with experienced Systems Engineers, who design a product specification considering their standardized functions. Then the system is designed by hierarchical functional teams, following their standardized architecture. Their work is also highly standardized due to specialization. This constitutes the ***expert model***. They take a hierarchical work process as shown on the left of Figure 1[1,7], which is intersected by the hierarchical documents. As the work process intersected by documents is design process knowledge, the hierarchical work process is taken as the ***knowledge model***.

The bottom left of Figure 1 shows the systematic acquisition of the lowest level design knowledge from documents. When design charts are left at each small step of a design, a concept (e.g. symbol) in a preceding document is hierarchically detailed into several pieces (e.g. symbol) of lower level concepts in the output documents. This hierarchical decomposition is the elementary design process knowledge, and is called ***a design rule***. From the adjacent two documents, design rule may be acquired, systematically, reliably, and easily. Similarly to this, other design knowledge may also be systematically acquired from various documents [1,7]. The right of Figure 1 depicts thus reproduced design knowledge. Connecting these design rules from the initial concept up to the source code, they form a huge hierarchically expanding semantic network. An expert system with a knowledge base storing these design rules behaves like an apprentice working quickly at a skill level. For this system, this is taken as the basic principle for automatic design.

3. Construction of an Expert System

3.1 Design knowledge

As has been pointed out in Figure 1, a design consists of various hierarchical detailing. Figure 2 shows major three documents of the system with their intelligent processing, a Function Structure Diagram (FSD), a Data Flow Diagram (DFD), and a Problem Analysis Diagram (PAD is a kind of structured flow chart). In the top center column, a functional module M is hierarchically detailed to lower level functional modules $M1{\sim}3$. Each of M's consists of several hierarchical detailing of functions. In the middle DFD and in the bottom PAD, a hierarchical detailing of function and the control is shown respectively. Each hierarchical detailing shows that a concept (e.g. symbol) in a preceding stage is hierarchically detailed into several pieces (e.g. symbols) of next level concept.

As is well known in Cognitive Science and Psychology, such hierarchical relations are the most fundamental elementary human knowledge. In another word, a design is a process forming a semantic network from a targeted concept to elementary logical or arithmetic operations denoted by a programming language statement. Thus, these hierarchical relationships are chosen to be the elementary design knowledge.

Thus a hierarchically expanding relationship of human concept consisting of a *parent* and the *children* is elementary *design knowledge*, and named a *design rule*. When they are cascaded, a hierarchically expanding network reaches to source code. In Figure 2, those are as follows:

1. *FSD design rule*

   In the top center column diagram, M is a parent and M1~M3 are children, and M to M1~M3 relation is called a FSD design rule.

2. *DFD design rule*

   In the middle center column of Figure 2 shows a pair of DFD, where a lozenge denotes data and a square denotes a function. A unit DFD (*D1-F1-D2*) as a parent is detailed to another DFD (*D11-F11-D12-F12-D13*) as the child. Such a parent-to-children relationship is called a DFD design rule. This case is a design rule which the parent's input-output data and the children's input-output data keep coincidence. For the automatic acquisition from DFD, a dotted arrow line shows the correspondence.

3. *PAD design rule*

   The bottom center column of Figure 2 shows a PAD. In this figure, a parent concept, *F2* symbol connected to several children symbols (*F21...F24*) located to the right corresponds to a PAD design rule.

One or several symbols in the input design diagram are hierarchically detailed to several symbols to be drawn in the design output diagram. An expert system has to do this type of detailing using design rules stored in it. In order to unify representation of the unit symbols, a super set symbol to represent all symbols has been chosen.

3.2 Standardization of data structure

As the design system transforms the input specification to the designed output, the input specification file requires a different structure for the output file. In a large design system, several different file structures become necessary. In such a case, a master file, a super set file structure, is taken to embody all the information. The standardization of data in hierarchical structure results in hierarchical structures of the internal data, resulting in a hierarchical structure of the design system, and the mono-functional structure enables simple design processes for the design of the design system, which may be easily standardized.

Figure 2 shows super set symbols for three diagrams: FSD, DFD, and PAD. The center column shows a design rule for each diagram, hierarchical expansion from an input to the output. The right side column shows unified representations for each case respectively. Rules for unified representations are as follows:

1. Elementary symbols in diagrams are listed upward.
2. Each design process expands design information hierarchically. The relationship between design process input and output is expressed by the terms of *parent* and *child* for each symbol.
3. In a design document, the symbols are lined to form a string to show the flow of information in a DFD, or PAD. These strings of the symbols in their original diagram are shown by *input connection* and *output connection* for each symbol. In the case where plural inputs or outputs exist, a serial number is added like *input connection 1.*

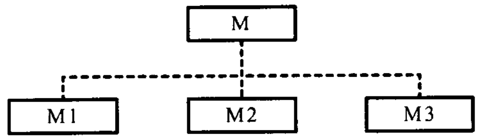
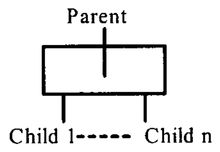
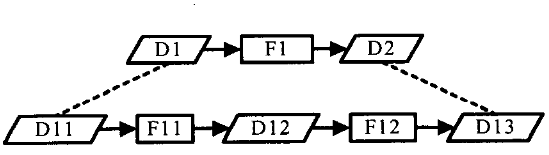
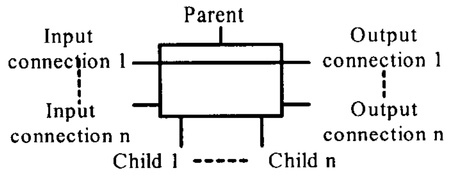
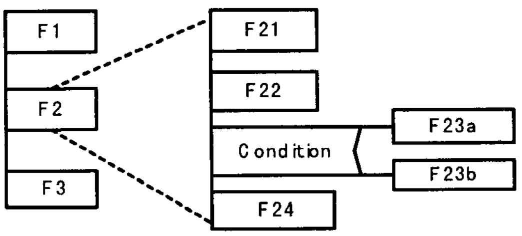
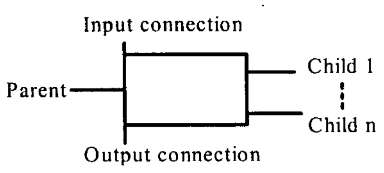
Diagram type	Hierarchical expansion	Representation of unit symbols
Function Structure Diagram		
Data flow Diagram		 Parent Input connection 1 Input connection n Output connection 1 Output connection n Child 1 ----- Child n
Problem Analysis Diagram	 F1 F2 F3 F21 F22 Condition F23a F23b F24	 Input connection Parent Child 1 Child n Output connection

Figure 2 - Unified representation of design knowledge

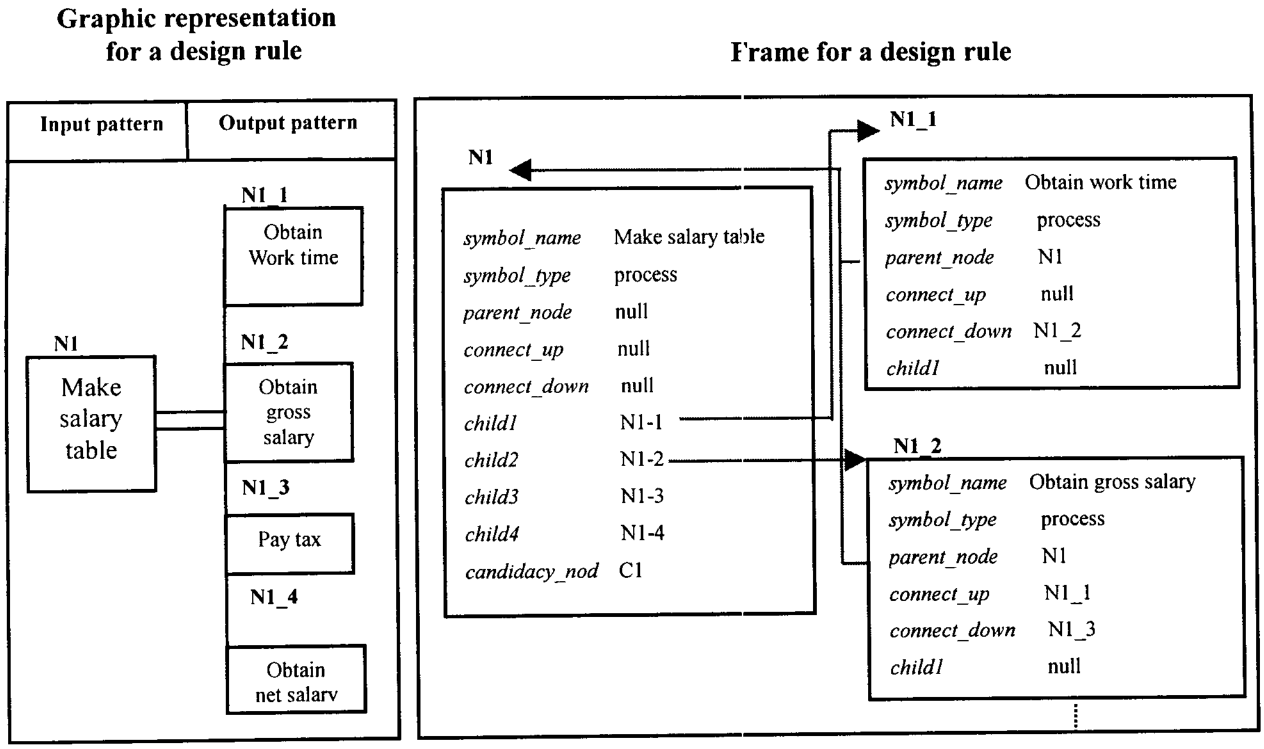


Figure 3 - An example of frame for a design rule

By the above rules, all symbols of several types of diagrams can be expressed uniformly. The design information is expressed by frames for knowledge processing. Figure 3 shows an example of frame for a design rule. The left part shows the graphic representation for a design rule, a hierarchical expansion from an input to the output. The right part shows the structure of such frames for the design rule and unified representations for each symbol. These unified unit symbols can be easily converted to frame representations. Thus all the design information in the system takes the same representation framework and design rules are also expressed by the same frame structure. The unified frame representation results in a standardization of both design information, design rules, and makes each processing easy to be standardized. This standardization over various types of documents enables that a structure of frame, acting as an inference engine, is commonly used for various documents such as PAD, DFD, and FSD.

3.3 Principle of design knowledge acquisition

The automatic acquisition of the design knowledge is attained by extracting a design rule that is a pair of the parent concept and its children. Here, a case of PAD's design knowledge acquisition is explained. In the bottom left of Figure 4, PAD, and a dotted line trace is shown. It is performed by a tree-walk program. It uses three types of symbols: *processing:* a square symbol in concatenation; *selection*: a wedged symbol and *repetition*: a square symbol with one vertical line. In each symbol, a natural language statement, or a part of a source code, must be written. Visualization of the program structure may be easy coding. In the leftmost side, the initial concept is hierarchically detailed to a group of children in the right side. As the initial concept is detailed, the PAD grows to the right as the tree grows.

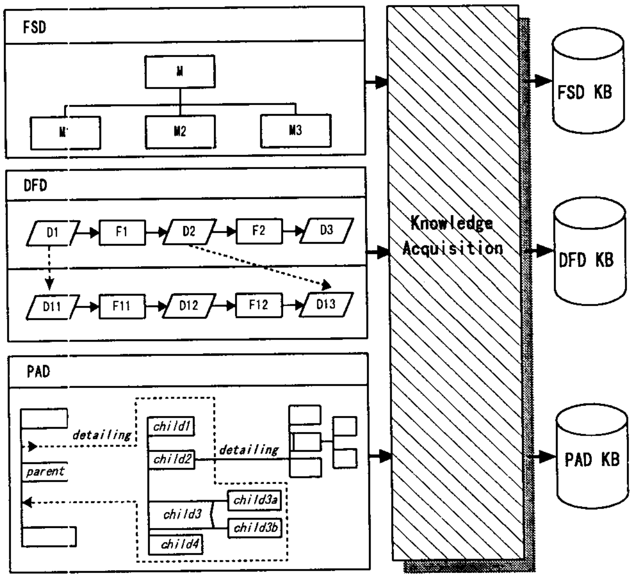


Figure 4 - Acquisition of design knowledge

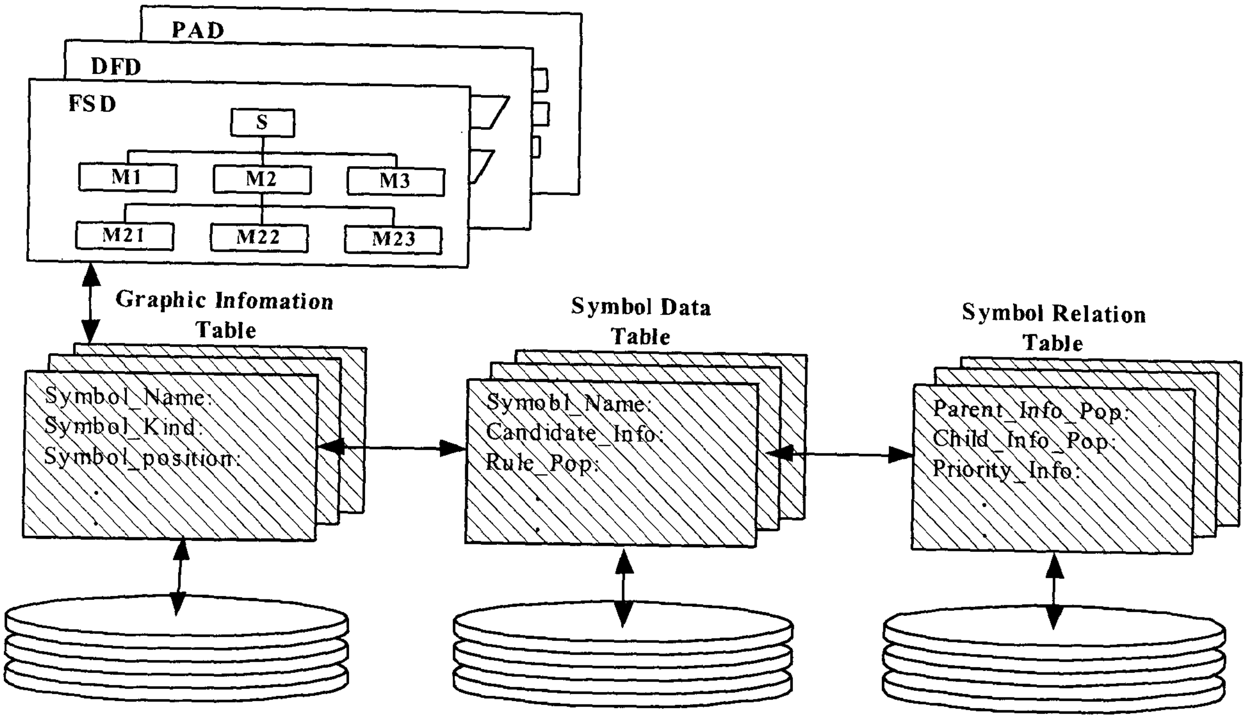


Figure 5 – A structure of knowledge bases

The program first acquires the type of symbol (*processing*) of the first symbol, the statement in it (*parent concept*), and the symbol position, and then repeats this for others until it returns to the starting point. From the acquired information, a design rule (*parent = child1... child4)* may be derived, and stored in PAD Knowledge Base. A major tree walk program, together with this local tree walk program, enables the automatic acquisition of design rules on the whole tree of PAD.

3.4 Structure of knowledge base

The Knowledge Base consists of a group of domain knowledge for program modules where commonality of natural language expressions of the design rules and integrity of resources and major variables are kept. The Figure 5 shows a structure of the domain knowledge bases. Each domain knowledge base consists of three tables, which are called *Graphic Information Table*, *Symbol Data Table* and *Symbol Relation Table*. Each symbol of various documents includes graphic information and design information. Graphic information of each symbol (e.g. symbol name, symbol kind, symbol position, etc.) is recorded in the *Symbol Table*. The graphic information of a symbol acquired by the local tree walk program is analyzed, and a design rule (e.g. a relation of parent and child(ren)) is created and recorded in *Symbol Relation Table*. *Symbol Relation Table* includes each tree structure of the design knowledge. To recover a design rule quickly, each symbol related information (e.g. position of design rule, position of graphic information) is recorded in a *Symbol Data Table*. By this structure of the knowledge base, the design knowledge may be quickly accessed, easily added, deleted and modified. When these tables are defined and made, constitution or expansion of a new domain knowledge base becomes very easily.

3.5 Principle of function design

In any kinds of software design, there appears a hierarchical structure. In the top managerial level, it is called 'hierarchical decomposition of object'. The design of software system comes next, starting with the specification. It is a hierarchical decomposition such as subsystems, modules, functions and so on, and during these the design information is detailed hierarchically and then standardized. Procedures for the standardization of function are as follows:

1. As a principle of Myers' STS division [9], a data flow is partitioned to mono-functional elements along the data flow. This is detailing in the horizontal direction.
2. As a principle of Jackson' programming [6], the gained mono-functional elements are decomposed hierarchically by small step in the vertical direction.
3. These hierarchical detailing of functional elements must be performed along with the hierarchical detailing of data.
4. The decomposition of the algorithm is shown by DFD. After the decomposition, the algorithm is

converted to a processing sequence, and the results may be shown by structured chart PAD. And so each design is done using by a pair of DFD and PAD.

When this procedure is repeatedly applied, a hierarchical function structure may be obtained. This process may be applicable from initial systems design phase down to programming phase. By this standardized work process, the same automatic design may be applicable from the high level design down to coding.

Based on the above-mentioned principle of software design, a standardized way of design process becomes possible. Figure 6 shows the idea of standardizing a design process. It decomposes an input function (Func.) to several lower level functions (Func1., 2, 3), which is a universal principle from system design phase to the detailed design phase. The middle level boxes in the Figure 6 shows the inside of the decomposition process. The decomposition of the algorithm is shown by DFD clearly. After the decomposition, the algorithm is converted to a processing sequence, and the results may be shown best by structured chart PAD.

By repeating a paired design process of functional decomposition by DFD then converting the algorithm to a control sequence by structured chart to the right side middle in Figure 6, the decomposition forward to the source code. This process may be applicable from initial systems design phase down to programming phase. When thus standardized, the same set of documents (DFD and PAD) may be used throughout all the design. This may be applied from top system level, then each functional unit, and finally down to each program module. By this standardized work process, the same automatic design may be applicable from the top-level design down to coding.

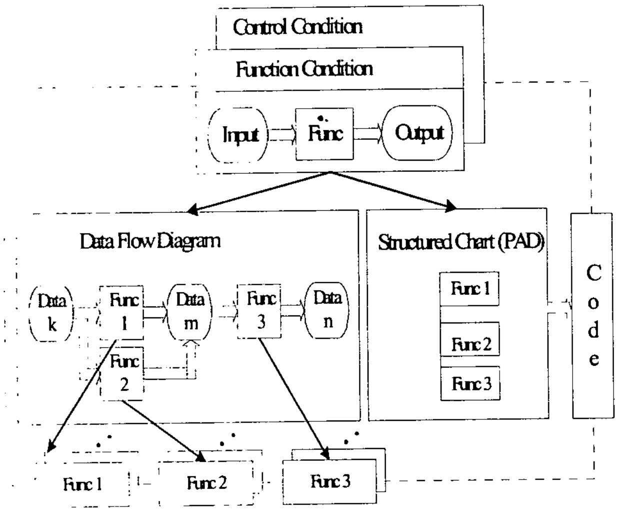


Figure 6 - An example of standard design documents

It should be noted that only one set of design knowledge for hierarchical decomposition and some control is enough for handling two types of design documents, DFD and structured chart. This minimization of required design knowledge is a vital key for enabling automatic design. At present, Integrated Intelligent CASE tool bases on reusing original human design. For a human designer, two step procedure of functional decomposition using DFD showing data first then adding control on structured chart is easier than doing the same using only structured chart. The effectiveness may be proved by a fact that some structured chart is used in combination with data notation.

When thus standardized, the same set of documents may be used. This may be applied from top system level, then each expert unit, and finally down to each frame. Thus design work process is reduced to a set of simple elementary procedures. This shows that a design of an expert system has been reduced to standard work procedures similar to design of software.

4. Overview of an Integrated Environment System

Integrated Environment system has been implemented in Windows. Visio [10], a commercial dedicated drawing system, is used for implementing FSD, DFD and PAD CASE tools. Graphical symbols in Visio are called shapes and they have a wide range of the property. Visio enables to construct CASE tools easily by the shapes following to ISO standard as well as the drawing environment. Displays may be achieved by various operating shapes and their properties. For knowledge acquisition and automatic design operations these graphics are hidden and programmers can design without thinking only FSD or DFD or PAD. The other intelligent tasks, conversion of graphical information, knowledge acquisition, automatic design, integration and control, are performed by Integrated Intelligent subsystem. A view of the system is shown in Figure 7. The typical combined design operation is as follows:

1. Each function is detailed to several lower level DFD's using the DFD CASE tool. All the function and data symbols bear a natural language concept. After the design completes, the corresponding PAD may be drawn automatically from the DFD, and some necessary control function, such as decision and repetition, must be added if necessary. Integration unit performs to generate these pieces of information for PAD.
2. A FSD CASE tool for Function Structure Diagram is provided. It shows the functional structure of a system in a hierarchical functional structure diagram. Two paired diagrams of DFD and PAD for a function is registered on a function symbol in the FSD. When a function symbol is specified on FSD display, a pair of DFD and PAD is displayed on the respective displays. Using these diagrams and tools, a standardized way of design from early phase down to coding becomes possible. But the unique way is to acquire automatically reusable design knowledge from documents, then detail automatically by reusing the design knowledge.
3. As the present system reuses design knowledge in

previous designs, a designer has to design a matched pair of sketches of DFD and PAD for each small step of design. In order to eliminate these designers' labor, reduce mistakes and keep coincidence with DFD and PAD, the integration unit is provided.

4. For knowledge acquisition, human designed paired diagrams are sent to the expert system and the integration unit checks the consistency of DFD and PAD for each small step design. The design knowledge is automatically acquired from respective documents and stored in respective knowledge bases.
5. For automatic design of DFD and PAD, a designer draws an initial design sketch of a DFD using DFD CASE tool, then the corresponding PAD may be generated and displayed on the PAD display. When automatic design is commanded, a DFD pattern on DFD display is sent to the expert system. Also the corresponding functions on PAD display are sent to the expert system. It checks the consistency of both information, and then reads out fragments of DFD and PAD which are hierarchically detailed information of the previous function. In continuous mode, the system repeats these hierarchical detailing.
6. By monitoring the progress of the automatic design on displays, the designer checks the design and intervenes when necessary. In cases when the automatic detailing is not made, the person has to do the design (equal to adding a new design rules). In cases where a correct detailing is not selected, the person selects an appropriate design.
7. A human designer has to trace and check the whole design. If the detailing is found to be going toward an incorrect way, the design must diverge from the previous design at some point by correcting the design rule to advance toward the new target, resulting in another 'children to parent' relationship. Thereafter, such a symbol bears special hatching for easy recognition.
8. During automatic design, in cases when a multiplicity of design rule exits, the most frequently used design rule (the first candidate) fitted to the situation is automatically selected and the hatching appears. When a hatched symbol appears, the human designer operating this system must select an appropriate one. All the possible design rules are displayed on the candidate view window when clicking "display candidates" button. The original parent concept and the design rules headed by the candidate's numbers are shown. In order to choose another design rule, the designer clicks the number, the selected design rule is sent to the original design graphic and the new one replaces previously designed part.

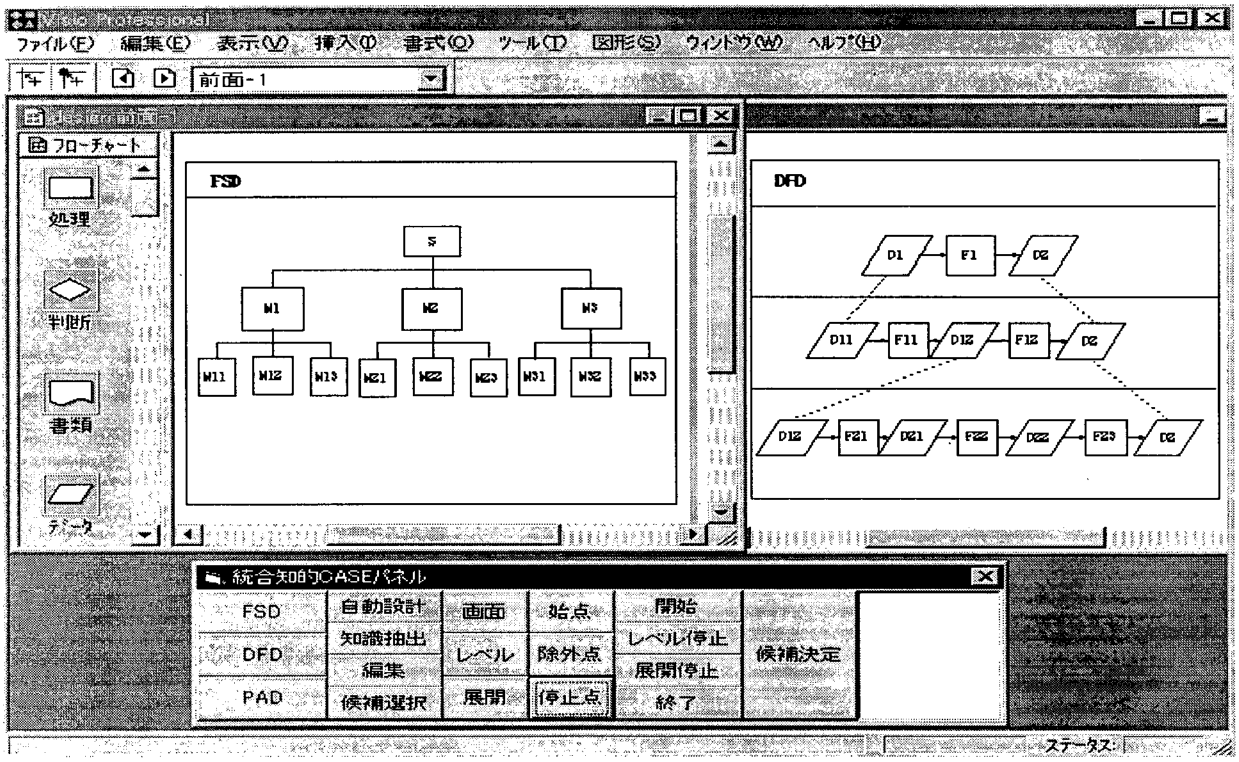


Figure 7 - Overview of the integrated environment system

9. This system can also convert a natural language defined symbol to source code automatically. The reuse of the relation of a natural language to source code eliminates the complex knowledge required for coding, and makes the system independent of the programming language used.

The system has many other functions. For a normal operation, the system works with both DFD and PAD CASE tools. At a high level design (such as for job flow), however only DFD may be used, and at the last end of detailed design only PAD may be used. For human intervention during automatic design, human operations must be made independently of other CASE tool. For these purposes, the system can select a CASE tool to be on-line or off-line.

5. Conclusion

This paper has reported a construction way of expert systems for automatic design. The systematic way is based on Software Engineering and systematic acquisition of design knowledge stemmed from a systematic design work process of well-matured developers. Design information has been expressed by unified frame representation. Based on the standardization of data as well as function, the design has been reduced to standard work procedures. As a result, the construction of expert systems becomes easy. This paper also introduced the integrated environment that may be applicable from top-level system architecture design, data flow diagram design down to flow chart and coding. The design knowledge is automatically acquired from respective documents and stored in the respective knowledge bases. By reusing it, a similar software system may be designed automatically. This system features an essentially zero start-up cost for automatic design resulting in substantial saving of design man-hours in the design life cycle, and the expected increase in software productivity after enough design experiences are accumulated.

Acknowledgements

This system is based on studies in the Software Creation Project. The author thanks many students for their contributions. The author special thank to Dr. Zenya Koono (Professor, University of East Asia) for his valuable advice on this study. The authors wish to express their gratitude to Hitachi, Ltd. and Visio Japan Company for supporting this study.

References

[1] Chen, H., Far, B. H., and Koono, Z. 1997. A Systematic Construction Method of an Expert System Used for Automatic Software Design --Acquisition and Reproduction of Design Knowledge from Design Process. *Journal of Japan Society for Artificial Intelligence*, Vol. 12, No. 4, 616-626.

[2] Chen, H., Tsutsumi, N., Takano, H., and Koono, Z. 1998. Software Creation: An Intelligent CASE Tool Featuring Automatic Design for Structured Programming. *The Journal of Institute of Electronics, Information and Communication Engineers*, Vol. E81-D, No.12, 1439-1449.

[3] Chen, H., Takano, H., Abolhassani, H., Koono, Z., and Far, B. H. 1999. An Approach to an Integrated Intelligent CASE Tool for Automatic Software Design. In Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering, 310-314.

[4] Futamura, Y. 1984. *Programming Techniques: Structured Programming using PAD*. Ohmsya.

[5] Hayashi, T. and Owaki, T. 1994. Japanese Software Development Methodology. V. Quality Centered Methodology and Tools for Developing Computer Control Software. *The Journal of the Institute of Electrical Engineers of Japan (C)*, Vol. 114, No. 6, 645-653.

[6] Jackson M A. 1975. *Principles of Program Design*. Mass: Academic Press.

[7] Koono, Z., Far, B. H., Sugimoto, T., Ohmori, M. and Chen, H. 1994. A Systematic Approach for Design Knowledge Acquisition from Documents. In Proceedings of 3rd Japanese Knowledge Acquisition for Knowledge-Based System Workshop, 253-265.

[8] Martin, J. 1989. *Information Engineering*. Prentice-Hall.

[9] Minsky M. 1986. *The Society of Mind*. Simon & Schuster.

[10] Mayers G J. 1978. *Composite/Structured Design*. New York: Van Nostrand Reinhold.

[11] 1997. Visio Manual. Visio Corporation.