

In The Name of Allah
The Beneficent, The Merciful

Proceedings
of the

Fourth Annual International
CSI (Computer Society of Iran)
Computer Conference
(CSICC'98^{*})

January 26-28, 1999

Computer Engineering Department
Sharif University of Technology
Tehran, Iran

and

Iran Telecommunications Research Center

* The conference is held one month later (and hence in 1999) because of the holy month of Ramadan.

Software Creation : Structure of human design knowledge and it's application to automatic software design

Hassan Abolhassani, Hui Chen, Zenya Koono and Behrouz H. Far

Department of Information and Computer Sciences, Saitama University, Japan

E-Mail: ahassan@cit.ics.saitama-u.ac.jp

Abstract

This paper reports researches in Software Creation project aiming at automatic design of software. In the project, 'Knowledge acquisition as modeling' has been taken using a high maturity software organization as the expert model. In it, the hierarchical work process becomes solid and unified. By virtue of the hierarchical work process, the analysis becomes possible to endlessly to reach an elementary human mental operation.

Keywords: Automating software design, CASE tool, Reuse, Design knowledge

1 Introduction

In order to respond strong needs for software development, various automatic designs have been studied. Due to unknown nature of human design, however the penetration of the automatic design is very slow. Software Creation project, started in 1991 [8], have been making researches aiming at to reveal what is human design, then to enable automatic design.

The project intends to establish a scientific basis for automatic design for any kind of software. The strategy is to learn from a human designer who can design any kind of software. The design knowledge must be acquired systematically, and the expert system must be constructed again systematically. This had been regarded as impossible before, but recent *Knowledge Acquisition as Modeling* [4] enables the systematic acquisition, and a project's research found the way to construct it in a systematic way.

In short, a high maturity software development organization has been taken as the model, and their standardized, therefore unified, work process structure enables the systematic acquisition. When the model structure is underlaid, thus gained knowledge may be implemented to the expert system systematically. The research strategy of the project is to make a bottom up approach, starting from the most simple unit and then implementing it to a cost effective design aid.

Intelligent Computer Aided Software Engineering (CASE) tool [1, 2, 3] introduced by the project is a tool for structured programming phase, where the design is the most simple (hierarchical detailing) and the man-hours in a development project is the largest. It assumes a knowledge model underlaying 'The principle of least effort', which is pointed out by Zipf in 1972 [10]. It says that there are many knowledge units for a problem in a human brain, and the person uses one of them starting from the simplest toward a less simple one. Following this view, the simplest unit is just to reuse the past designs (section 2) while the last stage one is to create a design using deep knowledge and complex inference engine (section 3).

2 Design knowledge

Figure 1 shows a hierarchical work process underlaid in the project. The hierarchical work process is a result of accumulating improvements for more than ten years. It is quite the same as that for hardware production. It stems from a hierarchical nature of human knowledge. The hierarchical work process is intersected by respective documents which is also hierarchically organized. In the case of hardware production, they correspond to intermediate products. By checking them, quality of products may be improved. Based on quantitative planning, a rational development schedule may be planned, by checking the progress any delay of a develop-

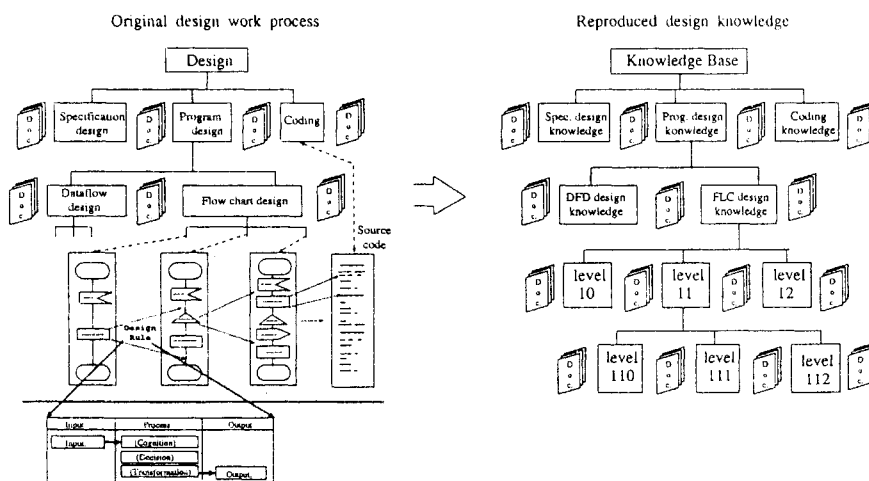


Figure 1: Original hierarchical design work process and the reproduced design knowledge

ment may be detected and appropriate counter-measures may be taken in advance. The process is an important key for developing high quality products.

Each of the process in a hierarchy, has a corresponding control structure (design knowledge), which performs a sequential control of lower processes being modeled by a state transition diagram. Another design knowledge checks reports from lower processes. There is another for managing the next work based on the forecast of the future. As many developments are done following the hierarchical work process in Japan, these kind of design knowledge may be acquired from managers and leaders of processes.

Another category of design knowledge structure common to every design is hierarchical decomposition of design information. The bottom part of Figure 1 shows an example (above dotted line). A box in a flowchart (SDL chart here) is hierarchically detailed. As the decomposition goes on, the operation becomes more and more clear and minute, and finally they are reduced to simple operations expressed by programming language statements.

These hierarchical decompositions may be acquired by a difference on successive design documents as shown in the bottom of Figure 1 (labeled as design rule in the figure). Thus, design knowledge may be systematically acquired throughout the entire work process. The right side figure depicts thus reproduced design knowledge.

As mentioned earlier, the most simple but the most man-hours consuming structured programming work process has been automated by

Intelligent CASE tool. A designer designs software using editor of CASE tool, or writes structure charts.

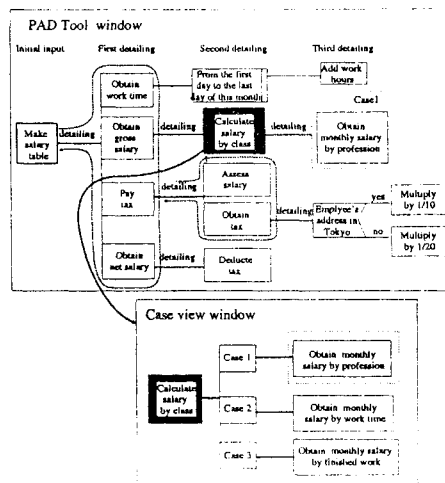


Figure 2: Display and selection of design rules

Figure 2 shows a sample of the structured chart. It is called Problem Analysis Diagram (PAD) [5]. A natural language initial concept is shown in the most left (function) box in the figure, it is then decomposed to several boxes to the right, and finally ends at boxes expressed by source code (omitted). A wedged box is a selection, and a box with one vertical line in it shows repetition (not shown in the figure). By detailing a function on the chart using concatenation, selection and repetition, a structured program may be designed by writing the chart. As the case of flowchart, a hierarchical decomposition (named a design rule) is acquired and is reused

for future designs.

The acquisition may be made automatically by a tree walk program as shown by dotted arrow line in the Figure 2. When a designer writes a starting function on the display of the CASE tool, it is again acquired automatically, and is sent to the Knowledge Base. If a design rule consisting of the same (parent) box is found, the children boxes are read out and the designer sees a hierarchical detailing has been made automatically.

During the automatic detailing, as PAD chart grows to the right, the designer has to check if it is detailed as intended, because this system just mimics design. There is a case where a multiplicity of child concept exists. In such a case, the most frequently used design rule is chosen. When needed, the designer can see others, and select another one as shown in the bottom of Figure 2.

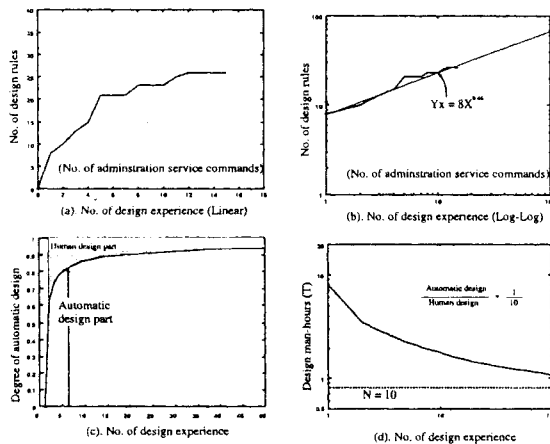


Figure 3: Evaluations

The effectiveness of the tool has been theoretically calculated by applying Industrial Engineering (IE) in hardware production. Figure 3 shows the results [1, 2, 3]. As the procedure has been reported before, only the outline is described here.

As it is understood easily, this system becomes more effective as it accumulates design rules. Figure 3.a shows the accumulation of new design rules as it experiences designs. It grows fast at first then the gradient gradually decreases. It is like a Learning Effect. Figure 3.b shows the curve plotted on both-Logarithmic scales as Learning Effect Engineering [6] in IE teaches. As the trend line is linear, it is judged to be a Logarithmic Learning Effect. By applying Learning Effect Engineering, the degree of automatic detailing has been calculated as

shown in Figure 3.c. At the initial design, it is zero, but it rapidly increases to more than 0.8. Figure 3.d shows the average man-hours decrease as the experiences are accumulated. At 10th experience, it decrease to 1/4 of human design. This proves that this tool is effective for improving productivity.

3 Detail of human design

As described, current implementation of Intelligent CASE tool uses a reuse unit to automate designs. As a result there are situations where designer has to create a new design rule, or guide design toward correct direction. For upgrading the automation level, therefore, it is necessary to consider this cases. For this purpose, in the current research, studies on *Design Rule Inventor*, for automating a simple hierarchical detailing, and *Design Direction Finder* for guiding a design toward correct direction have been studied. This section discusses about the study.

3.1 Approach

To reveal design knowledge related to *Design Rule Inventor* and *Design Direction Finder* a number of actual designs following hierarchical detailing, as describe in section 1, have been examined.

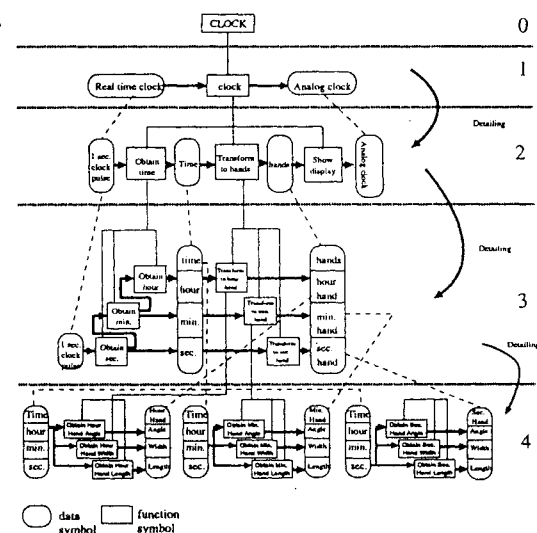


Figure 4: design of a clock program

Figure 4 shows hierarchical detailing for design of a clock program selected for simplifying

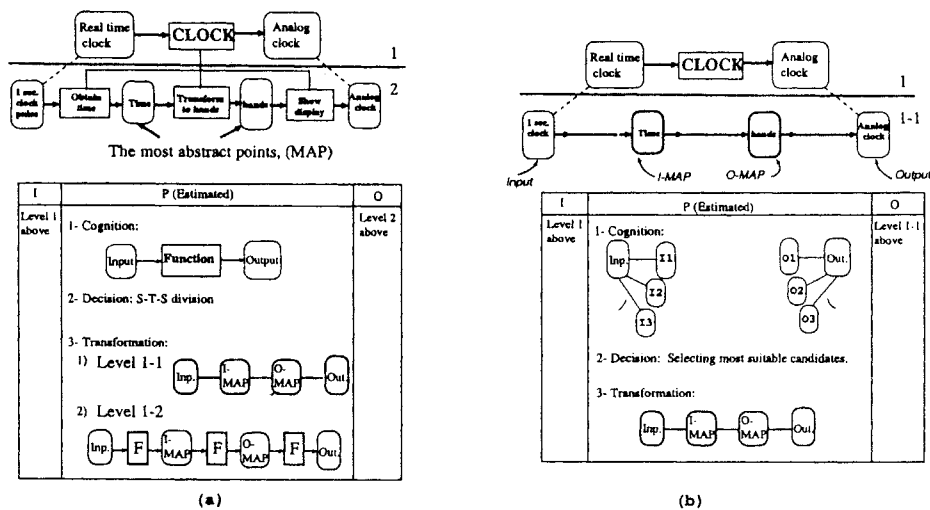


Figure 5: (a) First and (b) second hierarchical decompositions

following discussions. As shown, the initial concept *Clock* is converted to a dataflow by adding the input and the output, and the dataflow is divided to three functions, *Obtain time*, *Transform to hands* and *Show Display*. This division is made by two intersecting data, *Time* and *Hands*. This is an application of Myers' Source-Transform-Sink (S-T-S) division¹ [9] of a dataflow. By repeating dataflow divisions in this way, an initial function may be reduced to a dataflow chains of S-T-T...T-S. This is a typical example of hierarchical decompositions, and is named S-T-S case.

In the next lower line, the function *Transform to Hands* is decomposed to three parallel functions, *Transform to Hour Hand*, *Transform to Minute hand* and *Transform to Second hand*. This decomposition is determined by hierarchical structures of the data, *Time (Hour, Minute and Second)* and *Hands (Hour hand, Minute hand and Second hand)*. This shows that it is the base principle of Jackson's Structured Programming. This is another typical example of hierarchical decompositions, and is named JSP case² [7].

¹S-T-S division is a method for structured design proposed by Myers in the context of Data Flow Design to decompose a function. The major steps of this decomposition method is: 1) Identifying major data flow, 2) Finding two points in data flow where input and output are existed in most abstract forms, 3) Inserting three functions between: a) Input and it's Most abstract form, b) Most abstract input and Most abstract output and c) Most abstract output and Output. Myers combined this with module design and named it composite design. S-T-S division, however, is a universal principle applicable to any functional decomposition.

²J.S.P. division is another method for structured de-

3.2 The method of analysis

An estimation approach on inside of designing a hierarchical decomposition is taken. Since the design steps under study are small and standardized, the variation of estimations may be regarded as small enough. The guidance for the estimation is hierarchical decomposition.

Bottom of Figure 1 shows the estimation method. The problem is to estimate hierarchical decomposition (micro design steps) by the human mental operations during designing a design rule. As all human designs consist of *Cognition*, *Decision* and *Transformation* steps, micro design steps are recorded in an IPO chart in three steps of Cognition, Decision and Transformation, respectively. The cognition shows that what kind of algorithm for transformation is used, and suggests the Design Direction Finders' job. Transformations specify the embedded knowledge of micro operations during the rule, and so formulate Design Rule Inventor's job.

3.3 S-T-S case

Figure 5.a shows the detailing of level 1 to level 2 of clock example and the first estimation is projected to an IPO chart. As shown at the top of the figure, a function is detailed to three functions, or the data flow is intersected by

sign proposed by Jackson in the context of design by data structures. The idea is that the correspondence of input and output hierarchical structures determines the decomposition of a function to more detailed sub functions, resulting in a hierarchical decomposition of a function.

two data nodes named *Time* and *Hands*, which is called the most abstract input and output points (MAPs) respectively. As shown in the part of IPO, the estimated human design process can be stated as follow:

1. Cognition is a function with assigned input and output,
2. Decision is to take S-T-S division, and
3. Transformation is a two step process shown here as Level 1-1 and Level 1-2. The first one determines MAPs and the second one inserts functions between two successive data.

The next step is further decomposition of one step, as shown by Figure 5.b which shows estimation for level 1-1. Thus repeating hierarchical detailing, a mechanism as shown in Figure 6 seems to be probable for this case (*def* stands for defines and specifies that destination term, pointed by arrow, is defined by source term, i.e. using a mathematical expression. *parts* specifies a whole-part relationship). It shows estimated

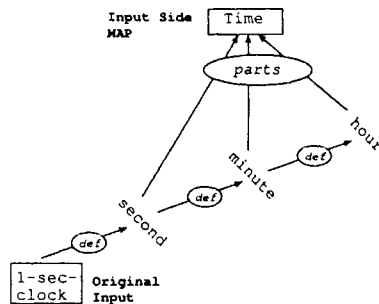


Figure 6: dictionary definitions

situation for selection of input side MAP and can be stated as:

1. Forming a data chain starting from the original input toward the original output,

2. Choosing an appropriate abstracted data.

Referring to the figure a possible mechanism may be implemented by introducing:

- Natural language dictionary defining a concept with lower level concepts.
- An inference engine for providing the data chains.
- Another one for preparing and defining abstraction levels.

Selection of output side MAP is implemented in similar way.

After MAP's selected, the next step is to insert functions between each pairs of data

symbols (for lack of space, estimations is not shown). They are named Sourc, Transform and Sink in a S-T-S division. Two cases can be considered for insertion of each of the functions. The simple case is where already there exist a function in knowledge base matching required function. Another case is where such function is not in knowledge base and so a new one must be created. In this case name of function can be selected following software engineering practices and its detailing can be created applying inventor for it. As a result a mechanism can be implemented by:

- A Knowledge base (extension of what exist in Intelligent CASE tool)
- An inference engine for selecting a function from knowledge base.
- Another one for creating a new function symbol, as discussed above.

3.4 JSP case

Detailing of *Transfer to hands* function from level 2 to level 3 of Figure 4 is a typical JSP case. It is possible to distinguish two cases that referred as *case A*, when hierarchical detailing of input and output data is given in advance and as *case B* where such detailing is not given. For *case A*, the first estimation of this rule can be stated as:

1. Cognition is that the input and the output has hierarchical structures and each component of the input corresponds to that of the output.
2. Decision is to take JSP division, and
3. Transformation is to insert three functions in between three positions bridging respective input and output data.

For *case B*, it is necessary to add another step before steps 1,2 and 3 above to create hierarchical detailing. It is represented as step 0 here and can be detailed further as:

- 0.1. Cognition is to consider all various hierarchical detailing for input and output.
- 0.2. Decision is to select the best hierarchical detailing for each as appropriate for JSP division, and
- 0.3. Transformation is to record such hierarchical detailing.

Either cases can be implemented by:

- Natural language dictionary defining a concept with other concepts.
- Knowledge base of past experienced designs.

- An inference engine for finding hierarchical detailing in *case B*.
- An inference engine for constructing correspondence of data structures.
- Inference engine for inserting functions between data items using natural language dictionary and/or knowledge base (as described for S-T-S case).

3.5 Discussion

In this section, method of analysis for finding detailing of design knowledge is presented and result of applying it for S-T-S and JSP cases is discussed. The next step is to develop and evaluate them in the framework of Design Rule Inventor and Design Direction Finder for flowchart level.

While current focus of study is on flowchart level, but the same mechanism as described in this section can be applied for upper levels. It means that the next phase of the project is to introduce design expert units (Reuse unit, Inventor and Direction Finder) for the dataflow level. Starting from actual human made designs, decomposing them to small steps until reaching elementary operations, and finally finding estimation of them is the main strategy as have been down for flowchart level.

4 Conclusion

This paper reported studies on human design knowledge. A hierarchical work process of high maturity organization has been taken as the knowledge model. In the process, all the work processes have an identical structure.

For hierarchical decomposition of the design input to the output, human design knowledge consists of a design knowledge unit creating a hierarchical decomposition, and a reuse unit for reproducing experienced designs:

- The later is the most frequently used unit, which is implemented in Intelligent CASE tool.
- For the former, various kinds of design knowledge units are possible. In this paper, S-T-S and JSP cases have been explained. They may be implemented by a natural language dictionary and relatively simple inference engines.

As these types of design knowledge may be used in any of the hierarchical work process, authors believes that accumulation of these studies and evolutions will lead to a more intelligent CASE tool than that at present.

References

- [1] H. Chen, N. Tsutsumi and Z. Koono, "Software Creation : An Intelligent CASE Tool for Reusing Design Knowledge Acquired Automatically," in Proc. of the Fourth World Congress on Expert Systems, pp. 218-226, Mexico, March 1998.
- [2] H. Chen, N. Tsutsumi, H. Takano and Z. Koono, "Software Creation : An Intelligent CASE Tool Featuring Automatic Design," in Proc. of the Tenth International Conference on Software Engineering and Knowledge Engineering, pp. 52-61, U.S.A, June, 1998.
- [3] H. Chen, N. Tsutsumi, H. Takano and Z. Koono, "Software Creation : An Intelligent CASE Tool Featuring Automatic Design for Structured Programming," in IEICE Transaction, to be published (1998.12).
- [4] K. M. Ford and J. M. Bradshaw, "Knowledge acquisition as modeling part 1, International Journal of Intelligent Systems," Vol. 8, No. 1, 1993.
- [5] Y. Futamura, "Programming Techniques: Structured Programming using PAD," 1984, Ohm sya. (in Japanese)
- [6] W. M. Hancock, and F. H. Bayer, "Ch2. Learning curve, (Gavriel Salvendy ed.) Handbook of Industrial Engineering," John Wiley & Sons, 1982.
- [7] M. Jackson, "Principles of program design", Academic Press, 1975.
- [8] Z. Koono, B. H. Far, T. Baba, "Software Creation: Towards Automatic Software Design by Simulating Human Designers", 5th International Conference on Software Engineering and Knowledge Engineering, pp. 327-331, U.S.A., June, 1993.
- [9] G.J. Myers, "Composite/Structured design", Van Nostrand Reinhold, 1978.
- [10] G.K. Zipf, "Human Behavior and the Principle of Least Effort," Hafner Publishing, 1972.