

Proceedings of the Third
Japanese Knowledge Acquisition for
Knowledge-Based Systems Workshop

JKAW '94

Venue & Date

Advanced Research Laboratory, Hitachi Ltd. (Hatoyama)
November 7 - 9, 1994

Sponsored by

Japanese Society for Artificial Intelligence

Supported by

Advanced Research Laboratory, Hitachi, Ltd.

In Cooperation with

American Association for Artificial Intelligence
The Australian Computer Society
Information Processing Society of Japan
The Institute of Electronics, Information and
Communication Engineers
Japan Society for Software Science and Technology

A Systematic Approach for Design Knowledge Acquisition from Documents

Zenya Koono, Behrouz H. Far, Takashi Sugimoto,
Mari Ohmori and Hui Chen

Department of Information and Computer Sciences, Faculty of Engineering,
Saitama University
255 Shimo-okubo, Urawa, Saitama, 338, Japan

Abstract

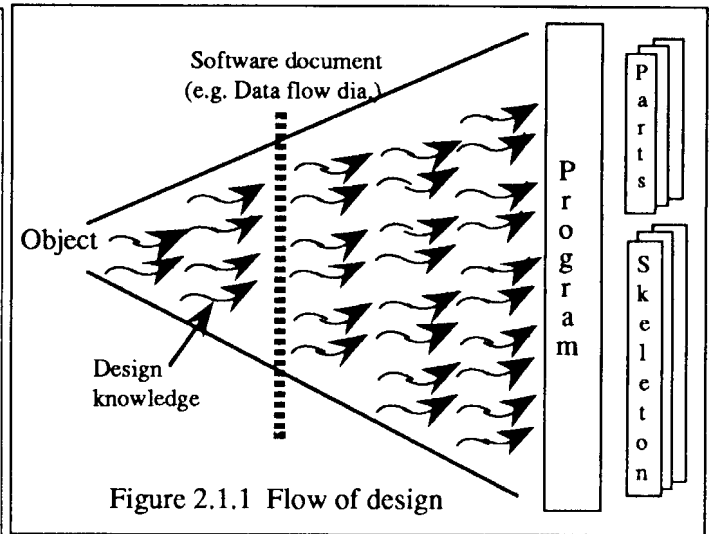
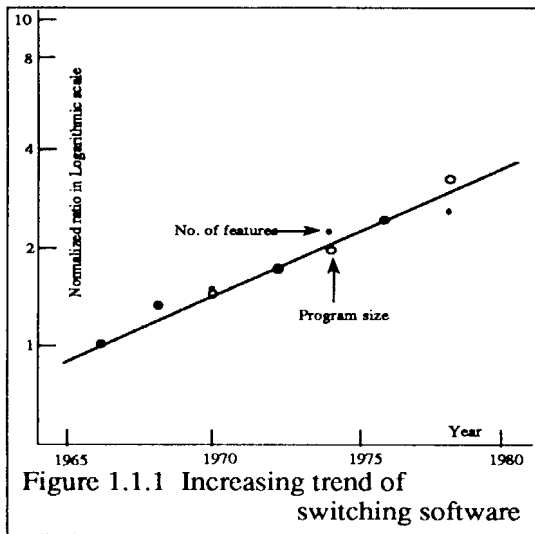
This paper reports on a design knowledge acquisition method by design documents, together with an automatic software design. This automatic software design has been studied to respond to industrial needs. It takes several industrial experiences into account, where some might be unfamiliar. Section 1 describes several of such prerequisites, such as extensive use of design documents. In Section 2, the automatic software design system is overviewed, and techniques and results are explained. The principle of extracting design knowledge is explained, and lowest level flow chart design and higher level data flow design is reported. Other examples of high level design rules are explained, and the structure of design knowledge is discussed. As a result of systematic extraction from a systematic design work process, the acquired design knowledge takes hierarchical structure, and each layer of which consists of both data flow type and control type knowledge. Reproduction of design by experts systems are described. Due to the systematic structure of design knowledge, expert systems have been systematically developed with ease. Besides these working design knowledge for detailing, there is another type of design knowledge for recognizing, deciding and directing design. As a result of these studies, it is pointed out that commonality exists in software design, hardware design, direct works, and also human business process in general.

1 Introduction

This paper reports a new method for extracting design knowledge systematically from documents[Koono. *et al.*, 1994]. It also becomes possible to construct an expert system systematically using this knowledge. Though the method is still immature, it will make a beginning of a new systematic acquisition of design knowledge. These are the results of following premises;

1. responding to strong needs in industry for systematic acquisition of knowledge,
2. hierarchical and systematic work process in human design,
3. abundant design documents for achieving high quality design, and
4. annual growth of systems software.

1. Industry needs a systematic knowledge acquisition method for applying Knowledge Engineering. When a system is put into service, it undergoes two kinds of changes, firstly to improve it for keeping competitive power in ever progressing technology and society, and secondly to adapt it to an always changing environment. Thus, systems must be updated almost continuously for at least five and a maximum of twenty years with so-called maintenance engineers. This method has been devised to meet this kind of needs.



2. The systematic work processes of human works is taken as a model, which enables a systematic approach as a result. In software developments, a matured high quality organization possesses such a process, while an immature one has a random to chaotic work process. In order to acquire expert's knowledge, a matured high quality team's work process has been taken as a model.

3. All such high quality development teams perform their tasks by leaving and utilizing abundant documents. They have a systematic document system, rich documents with accurate expressions, and rigorous checks using them. By them, their product's residual error rate after delivery reaches the order of 0.001error/KL[Kurahara et al., 1990] Abundant documents are prerequisites for high quality.

4. Systems software shows an annual growth, which constitutes a majority of so-called maintenance. It results in a huge software which invites Software Crisis. For one system, it is around 10%/year as shown in Figure 1.1.1[Koono 1979, original data in Joel. 1979].

2 An automatic software design extracting design knowledge using documents

2.1 Software Creation Project[Koono et al., 1992, 1993a, 1993b]

In Software Creation project, an automatic software design learning from human design has been studied, in which design knowledge is extracted from documents, and reused to reproduce design. Figure 2.1.1 shows design process, which the project intends to reuse. The left side starting point is the object or the target of a development, a right side square box shows the program obtained, and arrows in between these two show unit design knowledge. Program synthesis uses skeletons and program components, and they are arranged to form the program aimed at. Design knowledge is not a main player in it. Cliche approach[Rich et al., 1988] uses relations between frequently used names of function or knowledge and the corresponding program. It uses part of design knowledge, but gaps between cliches are left.

Learning software history, there is a long trend of bottom up automation of software design. It started from assembler and then compiler in the old days, which used fixed relationships between the input and the output. Presently it is becoming popular to use automatic conversion from structured flow chart to pseudo code or code. It also use fixed relationship between the input and the output. A more advanced method[Kono et al., 1983] starts from data flow diagram, a higher level document than flow chart, various data and function definitions, and generates programs for Electronic Data Processing. Our project also aims at automating software design in a bottom up manner by reusing design knowledge. A dotted intersection line in Figure 2.1.1 is a design document. A piece of design knowledge is acquired from the difference of two adjacent documents. The knowledge is extracted, systematically and elementarily, which may be reconstructed to reproduce the design. The method is universal and flexible.

A work process is decomposed hierarchically to several processes repeatedly. As it goes down, the decomposed work process become clear and simple procedures, and finally they are reduced to human elementary operations. It is called the hierarchical nature of work process in Industrial Engineering. As it stems from basic human nature, it is taken as a basis. Similarly, a hierarchical structure of design objects (functions, data etc.) is chosen. It means that the final object is hierarchically decomposed to several means to attain it, and repeating these hierarchical decompositions, these means become services. The service is hierarchically decomposed to the behavior to perform a service. Repeating these hierarchical decompositions, they are reduced to software functions, which is again hierarchically decomposed. As these decompositions go on, functions are reduced to elementary functions to perform so and so, which are then converted to some code in software. This is a basic principle of all kinds of design. Thus in our project, any specialities of programs are carefully avoided, so as to gain an overall picture of design[Koono et al., 1992].

In order to gain a generalized solution, software with sequential structure is chosen. A system is first decomposed to several Extended Finite State Machine (EFSM), which communicate with each other by discrete messages, and they behave as a much more complex Finite State Machine as a whole. State transitions are performed by state transition programs, in which the sequential nature is lost and conventional program technique is applicable. CCITT (ITU) Specification and Description Language[SDL] (SDL) - a kind of formal expression language developed for communication systems - has been chosen for the working language. It has both textual (SDL-PR) and graphical (SDL-GR) representations, and a SDL CASE tool, SDT[SDT] can edit texts or graphics, can convert them each other, can perform various check, and finally converts SDL to C/CHILL/Ada code.

The switching software was chosen as the problem area. A switching control system is strongly sequential, while a switching administration system is like a EDP program. Then human design was made based on a standard architecture. The design knowledge was extracted by design documents, and an expert system reproduce a design by design knowledge accumulated. The lower part is essentially detailing of flow charts but the upper part includes more complicated design. In the next phase, ordinary simple programs are chosen and the detailing of data flow has been studied. The design knowledge during data flow detailing is systematically extracted by data flow diagrams.

2. 2 Lower level hierarchical design by flow charts

In an actual design, such strict hierarchical decompositions are not made. But an excellent development organization featuring high quality always has a hierarchical product structure as well as standardized hierarchical work process system, in which documents are interfaces between processes. Such an organization is taken as a model.

Figure 2.2.1[Koono et al., 1992a] shows the principle of design knowledge extraction. The upper part of the figure shows the hierarchical structure of design work process. The lowest part of the

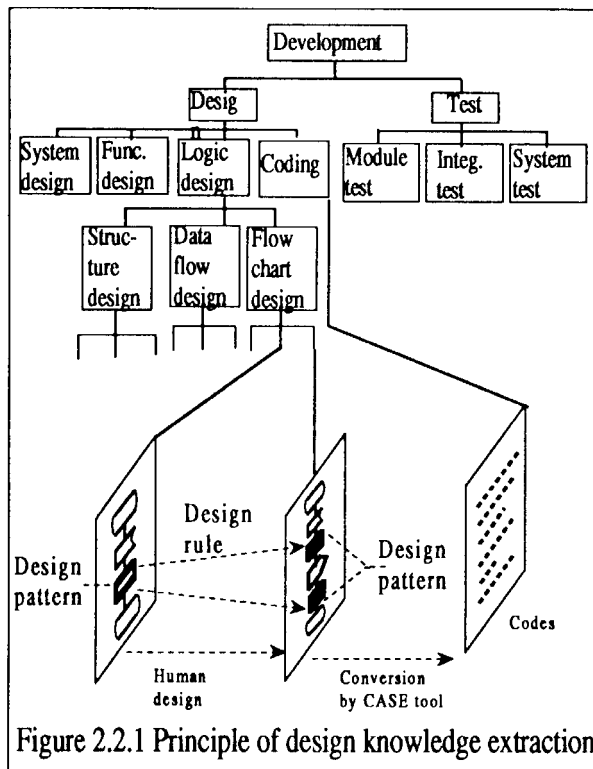


Figure 2.2.1 Principle of design knowledge extraction

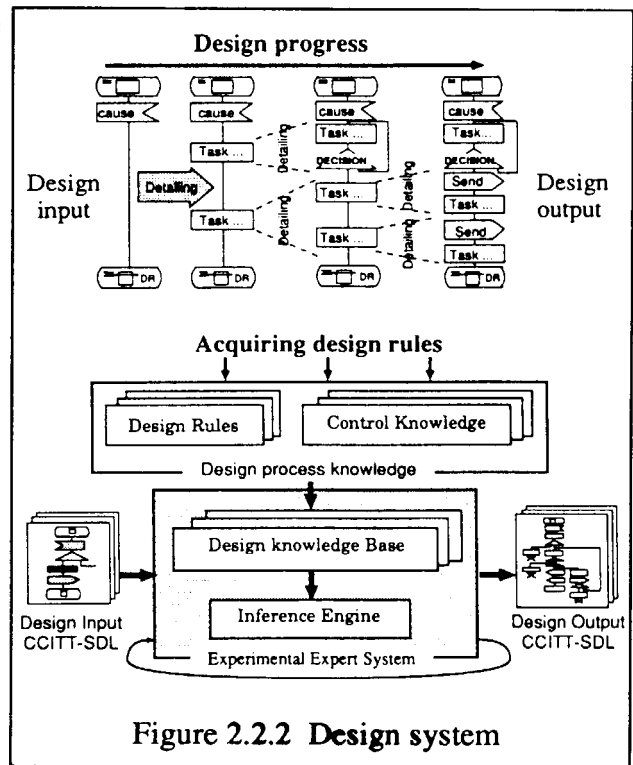


Figure 2.2.2 Design system

work process is to draw documents. The bottom part of Figure 2.2.1 shows the principle of extracting design knowledge. Take two adjacent documents, and the interval between them is taken to be small enough as to make the detailing simple, elementary corresponding to a piece of knowledge. An input and the corresponding output pattern are called design patterns, and the relation from the input to the output is called a design rule. The output design pattern is chosen to consist of several design patterns sent from following stages. When cascading these design patterns by design rules, the entire forms a hierarchical knowledge.

Figure 2.2.2[Koono et al., 1993a] shows the progress and the reproduction of the design. The upper part shows the initial human design leaving documents. Design knowledge is extracted from adjacent documents and stored in an expert system below. If the original input is given, the same output is reproduced by the expert system. If necessary design rules are equipped, it automatically designs. It acts as detailing flow charts from a general flow chart to the most detailed flow charts.

2.3 Higher level design by data flow chart[Koono et al., 1994a, Koono et al., 1994b]

Figure 2.3.1[Koono et al., 1987, Koono et al., 1993c] shows design steps of a clock as an example. As a whole, design starts from data flow design, then advance to flow chart design, and finally reach at coding. In data flow design, a conceptual *clock* is defined by its input and the output, and there is a data flow. The data flow is partitioned by Myers' S-T-S division[Myers, 1978]. The original input *real time clock* is detailed to *1 sec. clock* and *time now*, while the output *time display* is partitioned to *display board* and *hands*. As a result, the original function clock is decomposed to three functions, *obtain time*, *transform to hands*, and *display*.

Hereafter, the same procedures are repeated, and detailing of data flow goes on. Enough small progress of data flow design is taken to give a hierarchical expansion of data and function. A

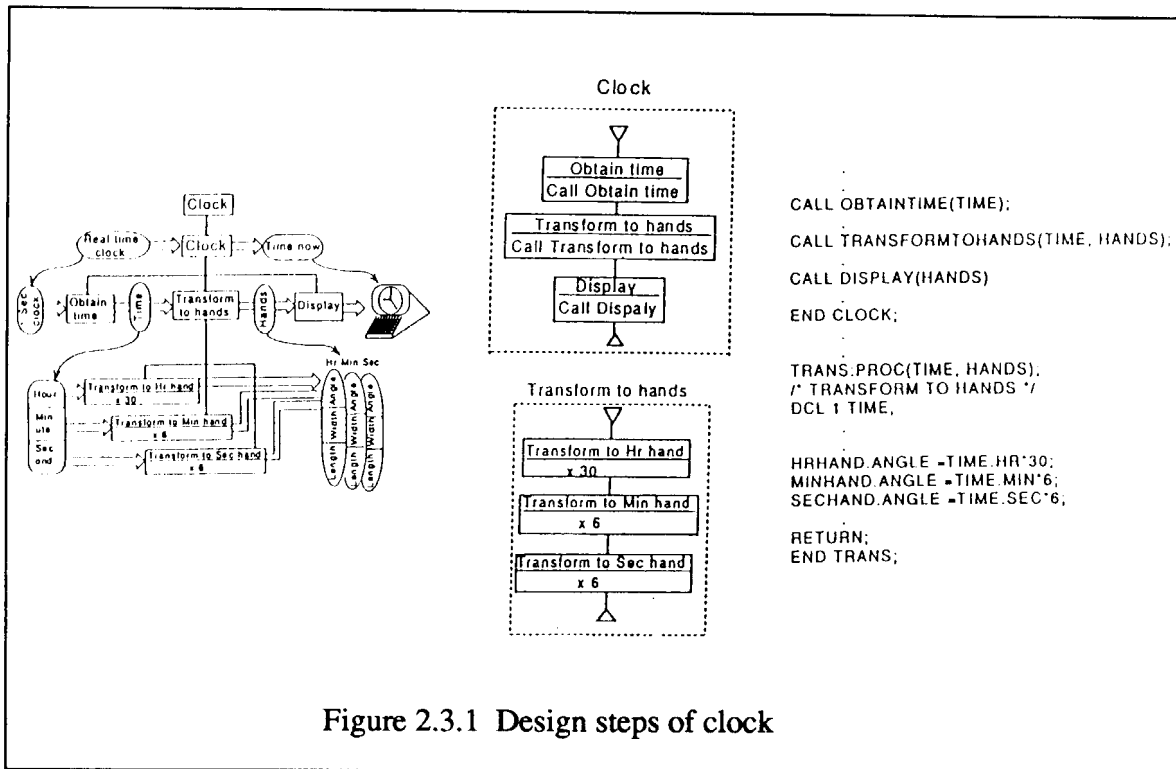


Figure 2.3.1 Design steps of clock

design rule is extracted similarly as before from documents. Design rules of detailing of data and function are extracted. Each of design rules of data and function form a hierarchical net of knowledge as before.

As a data flow is detailed hierarchically, there is a control knowledge that controls the sequential advance of the detailing. From the document, the knowledge may be extracted. Thus essential design knowledge of this level consists of data flow type knowledge for data flow (data and function) detailing, as well as sequential control type knowledge stepping data flow design. In the detailing of flow chart, there are the same two kinds of design knowledge, but data flow type knowledge is dominant and left unnoticed. In Figure 2.3.1, there is the highest level sequential control knowledge, advancing ... data flow design ... flow chart design ... coding. In this case, however, control type knowledge is dominant.

As shown in Fig. 2.3.2[Koono et al., 1994a], it is estimated that one level of design consists of data

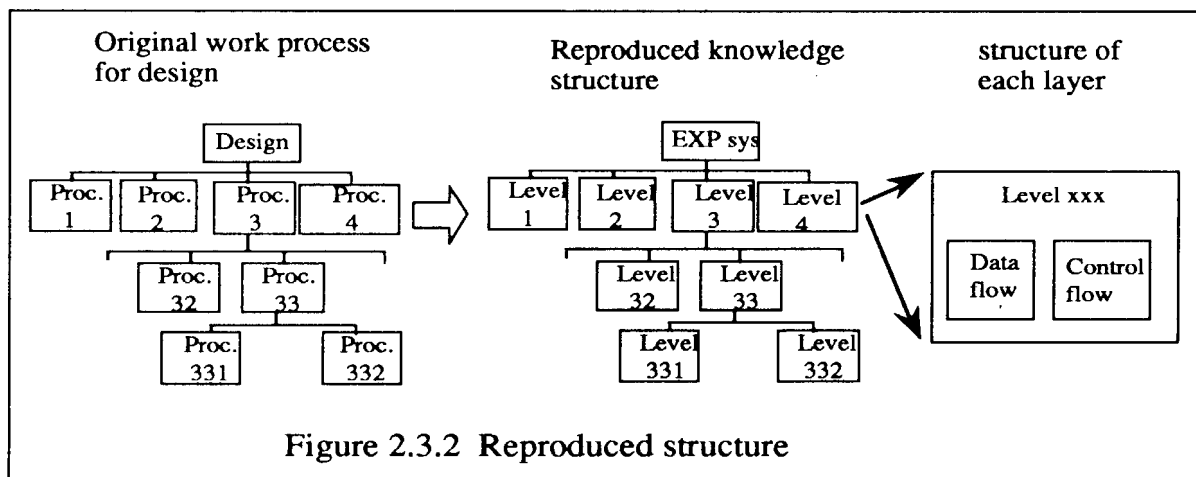
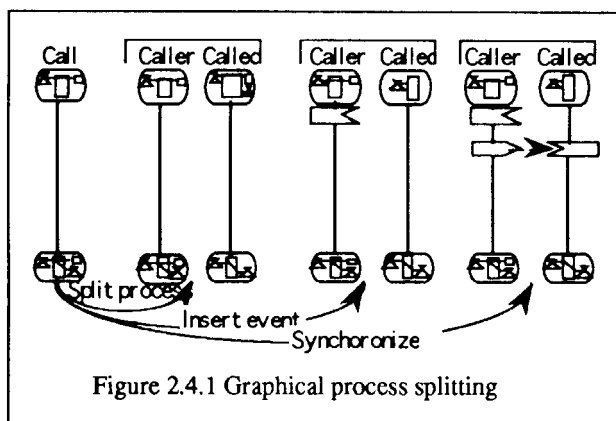


Figure 2.3.2 Reproduced structure

flow type design knowledge and control type knowledge. In the lowest work process, data flow type one is dominant, while in the high level control type one is dominant. These are results of systematic work processes of preceding human design. In actual design, however, such purely hierarchical design is rare and is called a good design. If not pure design were made, one level of design includes other level's design. It invites more complicated design knowledge. As what is needed for industry is to acquire experts' knowledge, it would be suited to assume hierarchical design throughout. This systematic knowledge acquisition method necessitates good designers instead of a good Knowledge Engineer.

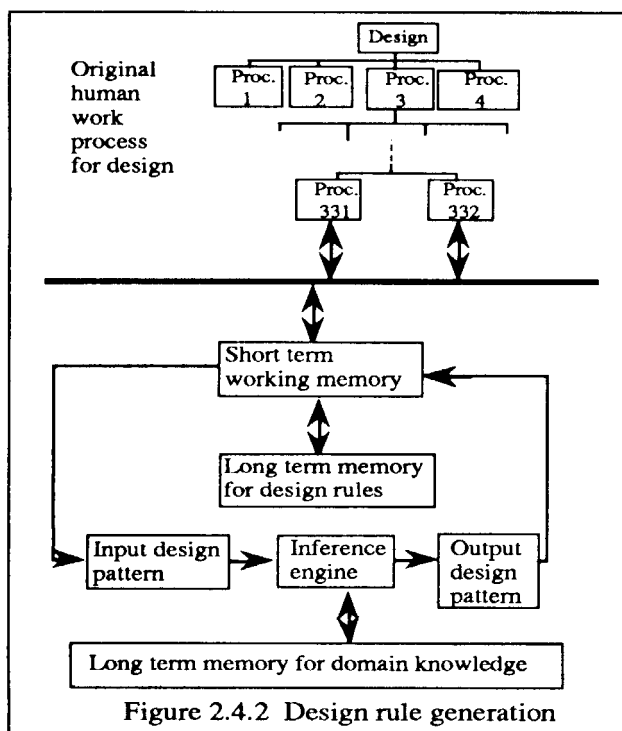
A data flow detailing is performed with data detailing and function detailing. There must be cooperation between them, which is not always clear from documents. There are several ways of detailing both. One example is Myers' S-T-S division which is first data detailing and then function detailing. The other example is Jackson's Program Development (JPD), which assumes a hierarchical data and mainly detailing of function goes on. As these decisions are made on recognition of other level's knowledge, must conform with the direction by higher level and part of such design knowledge exists in each level, they constitutes another class of hierarchical knowledge not yet clear in this method.



2.4 High level design using SDL

In the previous Section 2.2, all the detailing is a simple hierarchical correspondence from the input to the output. In SDL design, more complicated dedetailing is made, in higher levels than flowchart detailing.

A one step earlier phase than detailing of flow chart is 'task insertion'. Each state symbol has its state representation with pictorial elements. From the difference of the input and the output states (the initial and the final), a task for changing to the output state is obtained. The input in this case are two blocks of information, and relations between from the input to the output is conveniently expressed by a table logic.



Further earlier phase is a 'process splitting'. The initial input is talking states of a call (consisting of originating and terminating speakers' state), it is split into originating and terminating EFSMs in order to make the switching control program simple. In order to split the process with fractional information of design, a graphic method is devised [Ohmori et al., 1993] and shown in Figure 2.4.1. It consists of splitting states, copying events and inserting events following each algorithms. If an expert is assigned to this design, he/she almost immediately writes down the design output diagram for split processes, and can not explain

how and why it was done.

From these, it is estimated that;

*a design rule is a relationship between the input and the output,

*there are various types of the relationship,

#by a conceptual (using natural language) hierarchical expansion by human memory,

#by a conversion table logic,

by algorithmic expansion, and

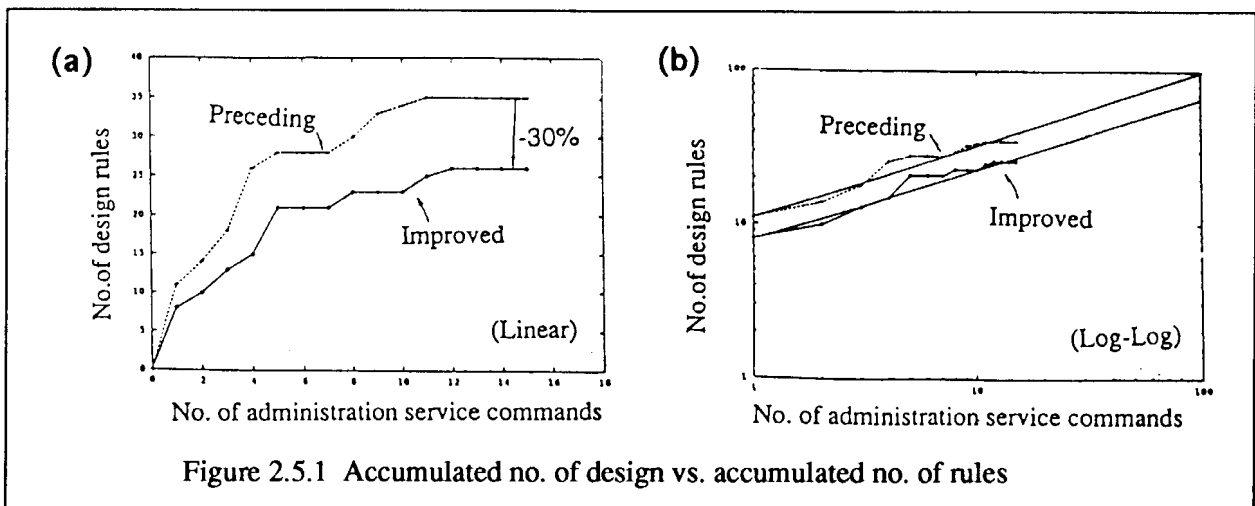
*a design rule may be generated from more fundamental knowledge, as Motoda[Motoda et al., 1989] pointed out.

Figure 2.4.2 is a model which visualizes the relation between design rules and the mechanism for generating a design rule. There must be other many algorithms which expands the input to the detailed output. (It also explains the increase of speed-skill of learning effect, and the reason why an expert can not explain why and how the work result is obtained.) Such expansion seems to be the essential existence for detailing. If all design rules were replaced by more fundamental knowledge, it would require more knowledge and consume more time for processing. Considering this, the difference of our system and academic models, based on mainly fundamental knowledge, is well understood. Our approach may be regarded as collecting least necessary knowledge for re-producing design.

2.5 Number of design rules

In an automatic design system using design knowledge, the number of design rules is the crucial factor. In our system, though the extraction may be done solidly and systematically by any person, the extraction requires a lot of labor. In it, when a design is performed with existing design rules, the system designs automatically. If a design rule is found necessary, it must be added on. It is a kind of learning system.

In Figure 2.5.1.a, curves show the case of a switching administration system[Hatae et al., 1993, Chen et al., 1994]. As it experiences various designs, the number of design rules or knowledge increases. At first basic design rules increase and when most of them are accumulated, a considerable number of features may be designed automatically without adding any design rules. When a new series of features begins, special rules become necessary. As basic rules only are extracted, new rules are all basic increments, thus the total increase is smaller than before. (If complex rules were taken, as a result, the curve increases first rapidly then gradually, but continues to increase without saturation. This kind of curve is frequently observed in human related matters such as the speed skill of human mental operation[Crossman, 1959] in Psychology, direct workers' man hours per operation[Salvendy, 1982] in Industrial Engineering, and productivity[Koono et al., 1990]

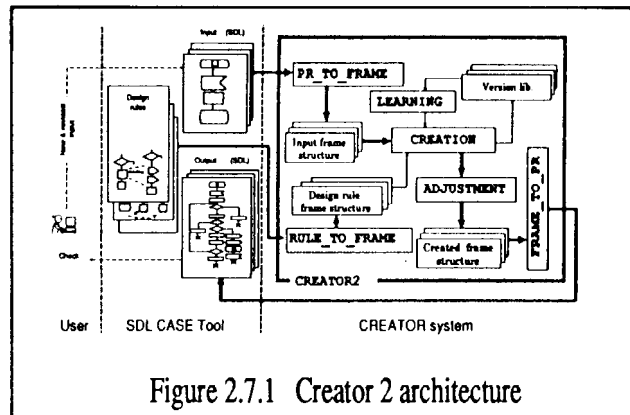
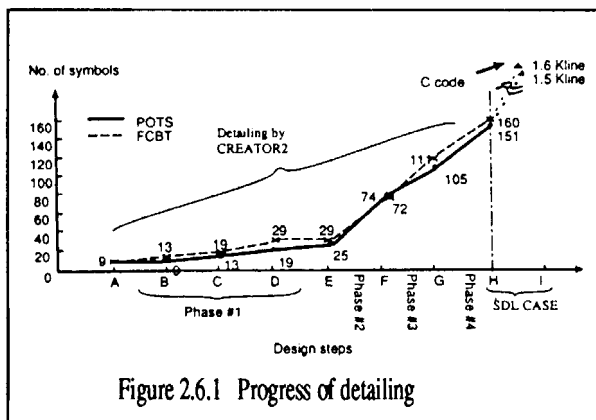


and quality[Koono et al., 1988] of software developments. They are called a learning effect. In Figure 2.5.1.b[Hatae et al., 1993, Chen et al., 1994], the vertical axis shows a performance factor and the horizontal axis shows the accumulated number of experiences, both in Logarithmic scales. When a learning effect occurs, the plot becomes linear. Plots show a linear trend line. This means that as the system experiences designs, the need to add a design rule decreases, which is favorable for actual use.

Improved methods to decrease the necessary number of design rules have been found[Chen et al., 1994]. In Figure 2.5.1.a and Figure 2.5.1.b, the results by improved method are shown by bold lines, which are decreased to 70% of the preceding. The essential part of a switching administration system is operations on various data. The original method repeats detailing of programs only. The improved method starts from the type of a data structure, which controls the operation of switching control programs, and therefore the target like nature of the design. The next phase of design uses PAD (Problem Analysis Diagram), a kind of structured flow chart. The third stage design uses SDL. From this experience, it was learned that the necessary number of design rules may be reduced by the following reasons;

1. Starting from the near target, the standardized final object reduces the initial design patterns,
2. Use of PAD simplifies the structure of the program to simple shape in PAD, and
3. When the design is standardized, the effective design rules may be obtained by much larger progress of design than before.

Thus it becomes possible to use two approaches, unit knowledge following learning effect and practical knowledge enough for the purpose, for reducing labor of extraction and amount of knowledge.



2.6 Reproducing design

Various types of expert systems, which reflect the extracted design knowledge, have been designed. They are named Creator. Before going to the detail, an example of the performance is shown. Figure 2.6.1[Koono et al., 1993a] shows the progress of detailing of switching program as its design progresses. Through splitting, task insertion and two stages of detailing, the number of the initial input symbols are detailed to around 15 times, and the SDL output is converted to 10 times large number of lines of C code, giving 150 times detailing in all.

2.7 Expert system architecture

Creator 1[Koono et al., 1992a] is an experimental model using conventional programming language. Since Creator 2, an Expert System Shell ES/KERNEL 2[ES/K] has been used. Figure

2.7.1[Far et al., 1993] shows the architecture of the initial Creator 2 for the detailing of a switching control system. The designer defines the initial input by graphic form SDL-GR using a SDL CASE tool, SDT[SDT]. It converts SDL-GR to (machine familiar) textual form SDL-PR, and it is fed to the expert system. It then converts SDL-PR input to frames, and after a stage of detailing by design rules, frames are again converted to SDL-PR, and the SDL CASE tool displays the detailed result. After checked and modified by the SDL CASE tool, it then is fed back to the expert system for further detailing.

A salient feature of the expert system architecture is data flow structure as shown before. As the main input and output data is standardized to SDL, design information may be recirculated or supplied to a SDL CASE tool. The internal data structure is a standardized frame structure, which is used throughout the system. The system consists of several mono-functional expert units. As it processes the bottom level data flow detailing, control in this case is very simple.

After Creator 3, various algorithmic detailing have been introduced. Standardization of data/frame structure, mono-functional expert units along the main data flow, and hierarchical structure of the system have been extensively used. They contributed to the rapid development of expert systems.

As a result of systematic and hierarchical structure of the knowledge as shown in Figure 2.3.2, expert systems may be easily planned, designed, and developed easily and quickly. Various standardizations are now being made.

3 Discussions

3.1 Merits and demerits of present method

The method explained in this paper enables an extraction of design knowledge and features;

- *M1. reliabilityextracted from solid documents,
- *M2. easy extractionextracted by any person,
- *M3. adaptabilityapplicable to any design document.

The demerits are as follows;

- *D1. need of excellent designer for the initial design,
- *D2. labor cost (development period) for preparing abundant documents, and
- *D3. need of experienced designer when special (algorithmic) relation appears.

Among the demerits, 'D1. need of excellent designer' corresponds to the need of Knowledge Engineer (KE), but an excellent designer is more easily obtained than a KE, and he also can contribute to the quality of the product whereas KE can not. In a development of systems software, it is very important that the original concept, intention and strategy is kept and followed for many years. A present day headache is the difficulty of keeping and following them due to the lack of documents, and the difficulty in teaching them to people coming later. From documents with small progress of design, various techniques are very easily reviewed and learned and not only the original way of thinking is easily understood but also very flexible adaptation to a new change becomes possible. If a beginner traces the progress of a design in small steps, he/she will understand how and what design should be done. So the demerit is decreased to a large extent. The solution to be taken is how to make the valuable accumulation of technical knowledge available for people beyond the product responsibility. It is, so to say, to share a great designer's brain. 'D1. need of experienced designer' is not necessarily a demerit. It hinders unexperienced people from violating a rough design neglecting the original idea.

'D2. labor cost for document preparation' seems essential. As has been shown in Figure 2.5.1, the additional efforts decrease as the system experiences various designs. As has been described in

Section 2.5, the necessary number of design rules may be reasonably decreased by standardization and good design. As for standardization and the need for good design, the essential improvement is beyond the scope of this paper. But, when preceding example documents follow this principle, standardization becomes much easier by reviewing, and good design may be learned, both by abundant documents. The solution must be a virtual document system using machine aid in connection with sharing the knowledge required.

Beyond them, the merit of reusing design knowledge is very large. The depreciation period for switching systems was formerly forty to twenty years, and it has been shortened to much less in order to receive the merit of technical innovations. During such a long period, in addition to the annual increase, change and addition requests special to each user have been added on. An annual increase of 10 %/year is 2.6 times for 10 years, 6.7 times for 20 years and 17.4 times for 30 years. It has passed around thirty years since the first computer controlled telephone switching system started its service. Most typical computer controlled telephone switching system's software has grown up to the order of millions of lines now. During the period, some systems were thrown away as the software became dirty. Some developers avoided to get into new market in order not to add annual maintenance costs by providing a nother series of software. Some developers gave up the business due to the increasing so-called maintenance cost. But if enough initial investment is paid to install a new system, the sharing of great designer's skill become possible, and running of a super computer for some hours can provide any addition or change of software, The investment is enough paid off.

3.2 Inside of design

Figure 3.2.1[Koono et al., 1987, Koono et al., 1990] is a structural model of design developed by authors. It is essentially the same as that in Industrial Engineering(IE) and that of Human Reliability Engineering(HRE). The beginning of design is a consecutive decisions such as what and how to make it, while the end of design is conversions such as detailed flow to code. In between these two is a mixture of both. Recognition precedes them.

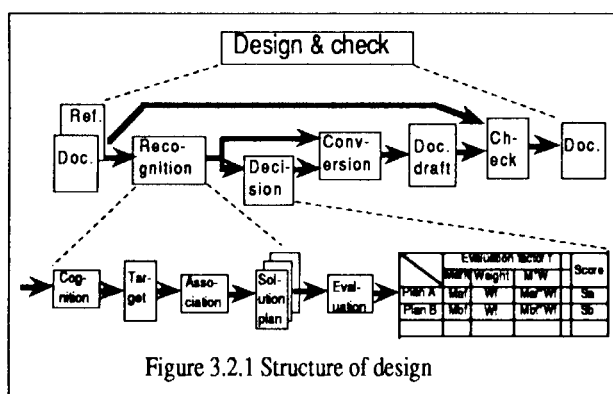


Figure 3.2.1 Structure of design

Almost all intersections of design may be explained by this model. A design starts from the recognition of the problem from documents, without which the information seems to be unreliable. The recognition of the problem, the target to be achieved, sometimes differs almost from people to people. Based on it, ideas for the solution is decided. The decision reflects designers technical as well as management and other knowledge. The result is how to convert the input, which requires the correctness of works to be done.

Recognition, decision and conversion may be further decomposed as shown. That of decision is taken from Management Science[Kepner et al., 1964]. The decision is regarded as a selection from candidates, and the selection is made on a simple comparison table, weighted comparison table or hierarchically abstracting weighted tables according to recent studies.

In HRE, it is usual practice to classify cause of errors such as error of recognition, decision, and operation[Hayashi, 1984]. A model for a human operator in IE[Salvendy ed.1982] is essentially similar to this, and takes other functions into account. As many factors effects on correctness and efficiency of people's work, information from various sensors in a body is mixed together with main information. Another feaure is detailing of an operation starting after a decision, as IE

people are concerned with operations performed. These impress us commonality of designer's activities and direct worker's ones.

3.2 Extended ways to estimate other knowledge

As a result of these studies, another concept to extract design knowledge has been found. An excellent development organization has been chosen as a model in this study. It usually has a hierarchical structure corresponding to hierarchical design work process, and people are specialized. In such an organization, specialized design knowledge may be extracted from a specialized people.

An example is a project leader, who controls the progress of a development, like function designdata flow design.....flow char design....coding.....etc.

It corresponds to the highest level design knowledge. As this example shows, the excellent organization's work specialization structure corresponds to the knowledge structure sought in this paper. In such an organization, a person is specialized to some task. In other words, the organization may be regarded as a hierarchical agent's group as Minsky pointed out'[Minsky,1985. Each mono-functional people corresponds to an agent with knowledge needed for the function. Therefore, the knowledge may be obtained from observations of the input, the output (again documents are the best). As for the transformation function during a task, it is estimated from these observations, it may be answered when a person is not so skilful to respond unconsciously.

Besides the knowledge system discussed before, there is another huge hierarchical knowledge system which surveille the actual design, abstract them to get the evaluation in a hierarchical way, make a decisions the way to design. In actual group works of development, raw information is gathered, abstracted and reported to the higher level as well as to be used in the same level. In a higher level, the report is checked and these reports are further abstracted and sent to the higher. During this, a careful person compares each report with what gained from other sources such as from a subordinate's subordinate and adjust the processing paying attention to what the boss is considering. The overall structure is a hierarchical complex with multilevel input. The simplest form would be robot vision model with neural networks. Thus fixed decisions and directions are dispatched from the top to downward, decompoed to details and expanded. Besides top's decision, various level's people begin to work according to their own decisions. The highest level decision is to some extent fuzzy and transmission of the organization as well as individual disions by people in between make the decision more fuzzy. They, then, affect the direction of design. All these would be design knowledge, which is pushed in a brain.

4 Conclusion

In this paper, a new approach for acquiring design knowledge from documents has been reported. In order to respond to industrial needs, and based on industrial practices and experience, a bottom up approach is taken for the study, and an excellent development organization with hierarchical structure with specialization is taken for a model.

Based on detailed and gradually progressing documents, it is possible to extract elementary knowledge reliably and easily. Thus repeating the study upward in the design, design knowledge with a hierarchical structure like hierarchical software may be obtained. The knowledge structure of an orninary level organization is a modified version of this.

The systematic knowledge acquisition method by documents can extract the results only, and can not reveal why and how it is done. But by making the difference between two documents smaller, more detail may be extracted.

The acquired knowledge is a replica of hierarchical work partitioning and specialization of the original design work process. The systematic and hierarchical nature of the acquired knowledge structure enables the easy and systematic design of an expert system with hierarchical structure for reproducing the design.

The automatic design system using this principle, though still immature, is expected to result in a new type of design system, by which the best designers' knowledge may be shared. It will lead to a new type of knowledge base to be shared by many people, and inherited for long time.

Such a system reproduces existing design correctly. It is like a hierarchically organized production department in manufacturing. Besides this type of working knowledge, another kind of huge hierarchical knowledge exists. It would be more important knowledge as it makes a person an intelligent and creative person apart from a machine. Using agents' analogy like Minsky[Minsky,1985], the knowledge may be extracted externally from human behavior.

During the study, it has been intended not to specialize to programming, but to stand upon the basic human activities common to all kind of design. It is based on the industrial fact, that a good component designer can be a good hardware designer, and a good hardware designer can be converted to an excellent software designer. Essentially what is described here may be applied to business processes in general. These shows that commonality exists in various systematic human activity. The revealing of both type of above knowledge makes it possible to explain all human conducts, and the way to success might be gained.

The method reported here is a tiny challenge and beginning toward a huge human knowledge.

Acknowledgements

The authors express their deep thanks to many students contributed to this study, without their naive views, this study would not be advanced. They also appreciate many pioneers in the industry who contributed to establish industrial ways.

References

- [Chen et al., 1994] H. Chen, B. H. Far, and Z. Koono. Software Creation: Reuse of Design Knowledge of Switching Software. Intl. Conf. Commun. Tech. '94, 1994.
- [Crossman, 1959] E. R. F. Crossman. A theory of the acquisition of speed-skill. Ergonomics, 1959, 2, p. 153-166, 1959.
- [ES/K] ES/KERNEL/2W-BS Reference Manual, Hitachi, 1991.
- [Far et al., 1993] B.H. Far, T. Takizawa and Z. Koono. Software Creation: An SDL-Based Expert System for Automatic Software Design. SDL Forum '93: SDL'93: Using Objects , pp. 399-410, Elsevier Publishing Co., North-Holland, 1993.
- [Hatae et al., 1993] K. Hatae, B.H. Far and Z. Koono. Software Creation: Design Rules of Switching Administration Program. Proc. IEICE Spring Conf. , 1993. (in Japanese)
- [Hayashi, 1984] Y. Hayashi. Human Reliability Engineering. Kaibunndo, 1984.
- [Joel, 1979] A. E. Joel. Bell system features and services. International Switching Symposium 1979 , Paris, 1979.
- [Kepner et al., 1964] C. H. Kepner and B. B. Tregoe. The rational Managers: A systematic approach to problem solving and decision making. Mc Graw Hill, 1964.
- [Kono et al., 1983] F. Kono, J. Ugai, M. Nakaoda, and K. Kishi, Development of the application program generator based on the data oriented specification language λ DSL \ddagger , Hitachi Hyoron, Vol. 65, No. 7, p p. 53 - 57, 1983. (in Japanese)

- [Koono, 1979] Z. Koono. Central processing systems in high integration age, Joint Conference of Four Electrical Institutes , 1979. (in Japanese)
- [Koono et al., 1987] Z. Koono and M. Soga. Structural way of thinking as applied to systems design. IEEE Global Telecommunications Conference '90 , 1990.
- [Koono et al., 1988] Z. Koono and M. Soga. Structural way of thinking as applied to improvement process. IEEE Global Telecommunications Conference '88, 1988.
- [Koono et al., 1990] Z. Koono, H. Tsuji, and M. Soga. Structural way of thinking as applied to productivity. IEEE ICC '90, 1990.
- [Koono et al., 1992a] Z. Koono, T. Baba and T. Yabuuchi. Software Creation: A Trial for Switching software. Proc. 5th JC-CNSS, 1992 Joint Conf. on Communications, Networks, Switching Systems and Satellite Communications, 1992, pp. 261-265.
- [Koono et al., 1992b] Z. Koono, T. Baba, Y. Yabuuchi, Y. Naito, Y. Shigemori, and K. Miya, Software Creation: Systematic Approach. Technical Report of IEICE , KBSE 92-28, pp. 33-40, 1992. (in Japanese)
- [Koono et al., 1993a] Z. Koono, B.H. Far, T. Baba, Y. Yamasaki, M. Ohmori, and K. Hatae. Software Creation: Towards Automatic Software Design by Simulating Human Designers. Proc. 5th Int. Conf. on Software Eng. and Knowledge Eng., SEKE' 93 , pp. 327-331, 1993.
- [Koono et al., 1993b] Z. Koono, B.H. Far, T. Takizawa, M. Ohmori, K. Hatae and T. Baba. Software Creation: Implementation and Application of Design Process Knowledge in Automatic Software Design. Proc. 5th Int. Conf. on Software Eng. and Knowledge Eng., SEKE' 93, pp. 332-336, 1993.
- [Koono et al., 1993c] Z. Koono and T. Ohtsubo. Evaluation of build-in and check-out of software errors. Technical report of IPSJ , vol. 95-5, 1993. (in Japanese)
- [Koono et al., 1993d] Z. Koono, T. Baba, Y. Yamasaki, and M. Ohmori. Software creation: Automatic software design derived from human design. Technical Report of IPSJ, 93-3, 1993. (in Japanese)
- [Koono, et al., 1994a] Z. Koono, B. H. Far and T. Sugimoto. A method for acquiring systematic design knowledge. 8th Natl. Conf. Japan Soc. Art. Int, 1994, (in Japanese)
- [Koono et al., 1994b] Z. Koono, T. Sugimoto, T. Tanaka, and B. H. Far. A systematic acquisition of human design knowledge. Proc. JCKBSE'94, 1994.
- [Kurahara et al., 1990] B. Kurahara, K. Uchimaru, and S. Okamoto. A Total Quality Control of a technical group, JUSE Press, 1990. (in Japanese)
- [Minsky, 1985] M. Minsky. The society of mind. 1985.
- [Motoda et al., 1989] H. Motoda and K. Yoshida. Qualitative reasoning as a basis of deep reasoning. J. of JSAI, Vol. 4, No. 5, pp. 538-546, 1989.
- [Ohmori et al., 1993] M. Ohmori, B.H. Far and Z. Koono. Software Creation: Design Rules of Switching Service Program. Proc. IEICE Spring Conf. , 1993.(in Japanese)
- [Rich et al., 1988] C. Rich and R. C. Waters. A research overview : The programmer's apprentice, IEEE Computer, pp. 11-25, Nov. 1988.
- [Salvendy, 1982] G. Salvendy ed. Handbook of Industrial Engineering. John Wiley & Sons, 1982.
- [SDL 92] CCITT Recommendation Z.100, Specification and Description Language (SDL) , ITU, Geneva, 1992.
- [SDT] SDT CASE Tool ver. 2.2 Reference Manual, Telelogic, 1992.