

**Proceedings of
Specialists' Meeting on
Application of Artificial Intelligence
and Robotics
to Nuclear Plants**

AIR' 94

May 31, June 1, 1994 Mito City, Japan

Sponsors: Cross-over Research Group on Artificial Intelligence
Advisory Committee for Promoting Research and
Development of Base Technology

Co-sponsors: Science and Technology Agency
Power Reactor and Nuclear Fuel Development Corporation
Japan Atomic Energy Research Institute
Institute of Physical and Chemical Research
Ship Research Institute
Electrotechnical Laboratory

USING SPECIFICATION AND DESCRIPTION LANGUAGE (SDL) FOR CAPTURING AND REUSING HUMAN EXPERTS' KNOWLEDGE

Behrouz HOMAYOUN FAR and Zenya KOONO
Department of Information and Computer Sciences
Saitama University
255 Shimo-okubo, Urawa 338, Saitama, Japan
far@cit.ics.saitama-u.ac.jp

ABSTRACT

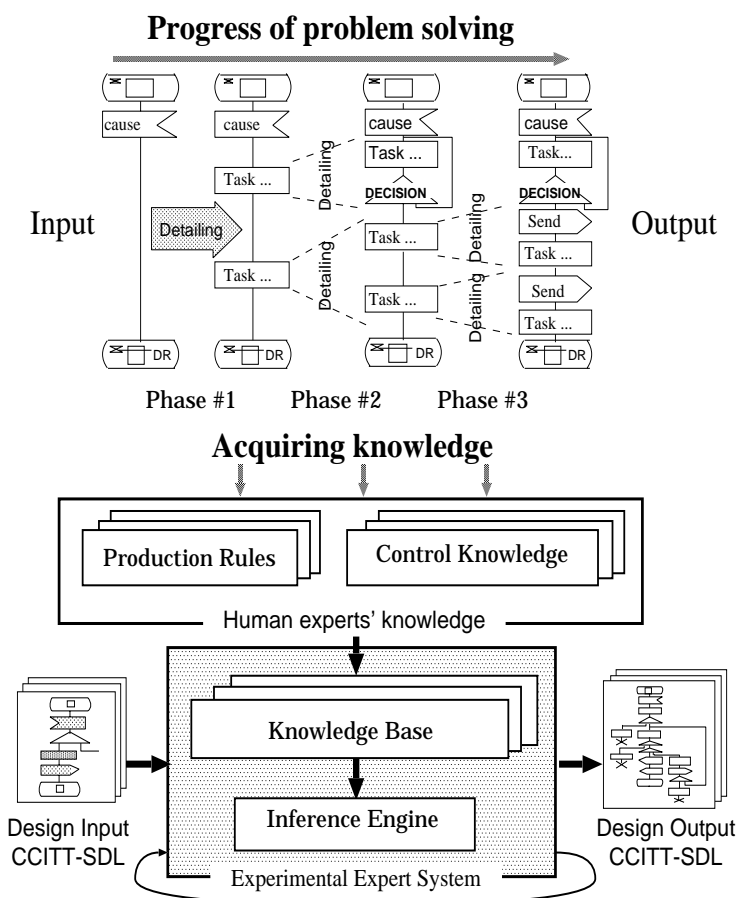
Conventional knowledge engineering techniques for acquiring experts' knowledge can not produce quality knowledge due to improper knowledge documentation and informal knowledge acquisition method. We propose a new method for knowledge documentation and acquisition using Specification and Description Language (SDL). SDL is used to describe both the target system and the reasoning process. The main idea is to follow deterministic problem solving behavior of human experts and document it. Then knowledge can be extracted by comparing documents of the successive steps. This knowledge is recorded and reused in similar or novel cases. We present an implementation of this method in a tool for software design. The implemented system consists of a SDL CASE tool and an expert system for applying the design knowledge. This system serves as an experimental platform for the study of human design by simulating the design at the lowest level. However, we have found that by acquiring enough domain knowledge, this system can simulate general problem solving of human experts.

I. INTRODUCTION

In this project we study knowledge acquisition process of designers and operators of engineering systems, with paying special attention to human designers when designing an artifact or when human operators of man-made systems, such as a nuclear power plant, reasoning about behavior of the system. Figure 1 depicts the idea and key concepts. The main idea is to follow problem solving steps of human experts. This knowledge is extracted from an actual problem solving process, documented using Specification and Description Language (SDL), and reused by an expert system. This is shown in the left part of Figure 1. Hierarchical nature of the problem solving process is assumed.

Opposite to the common knowledge engineering method for knowledge acquisition, in this project, first, we focus on knowledge documentation. This is necessary to produce quality knowledge. Then the knowledge is acquired, in the form of *rules*, by comparing the documents in successive steps of problem solving, as shown in the same figure.

Finally, *control* knowledge is used for selecting and applying such rules. The research starts from the lowest and the most detailed step where the detailed documents are available and goes upward hierarchically to more knowledge intensive areas.



Key concepts and assumptions:

1. Simulating human experts' problem solving;
2. Reuse of process and product knowledge;
3. Hierarchical decomposition of work process;
4. Bottom-up extraction of knowledge;
5. Top-down application of knowledge;

Figure 1 Overview of the knowledge acquisition procedure.

Specification and Description Language (SDL) [1], developed by ITU (formerly, CCITT), has been applied successfully in this project. We have found that SDL is an ideal tool for recording various steps of reasoning procedure. Formal and precise and still ready made concepts in SDL are quite useful to extract correct pieces of information. Furthermore, SDL allows recording *comments* added to the main routine. Those comments are pieces of data not included in the main course of reasoning but useful to reproduce problem solving state in a similar case.

We have used this method to derive experts' design rules and implement them in an expert system for software design. We have recorded the design rules in a particular design case and implemented those rules in an expert system. Those rules are reused when designing similar objects and/or modifying the original one. We have also applied the same technique to design simple devices. We are extending the idea to capture, first, the higher level design knowledge, and second, integrating human designers/operators' knowledge with deep models of the designed artifact.

The structure of this paper is as follows: Section II describes the knowledge components that are acquired. Section III introduces the expert design assistant system and Section IV gives an overview of the implemented system. We give a short discussion on the capabilities of the suggested method in Section V and conclude with some future research problems in Section VI.

II. COMPONENTS OF HUMAN EXPERTS' KNOWLEDGE

There are two inter-related knowledge categories involved in human reasoning, addressing either the *reasoning process* or the *reasoning product*. Here we concentrate on documentation, representation and support of both, in reusable form, particularly by introducing an structure for integrating these two.

During the intermediate reasoning steps, human experts rely on their *reasoning process knowledge* that is accumulated over years of solving similar problems. The reasoning process knowledge includes certain transformation patterns, or *rules* for converting an initial idea and *control knowledge* for applying the patterns. The *reasoning product knowledge*, on the other hand, consists of domain-specific concepts and constraints of the task.

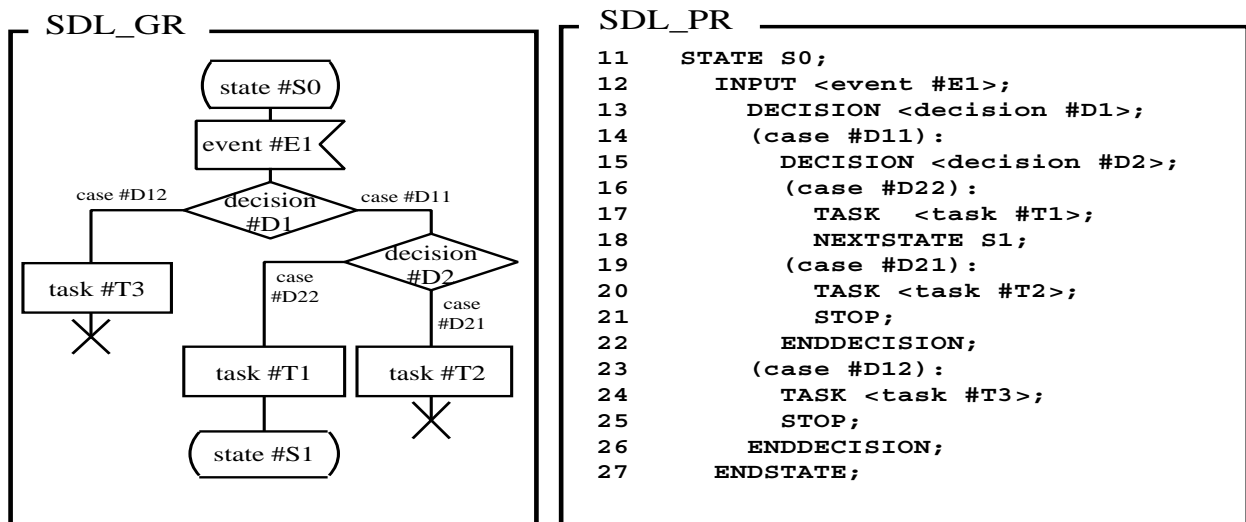


Figure 2 Example of SDL-GR and SDL-PR.

A. Reasoning product knowledge

The product knowledge relies on the perspective that the target system is viewed. It consists of domain-specific concepts, constraints and qualitative or quantitative models of the target system. We have used the Specification and Description Language (SDL) [1] for representing this knowledge.

In SDL a system is described in terms of *blocks* and *processes*. A process is described by a number of simple symbols, such as TASK and DECISION and each symbol may have some data attributes. Furthermore, system STATES are defined and behavior of the system is described in terms of state transitions. Other *data* objects can be saved as *comments*. SDL has both human friendly graphical (SDL/GR) and machine friendly textual (SDL/PR) representation. Fig. 2 shows an example of SDL-GR and SDL-PR. There are some CASE Tools, such as SDT [12], allowing editing, verifying and simulating behavior of the target system and translate between SDL/GR and PR.

B. Reasoning process knowledge

Reasoning process is viewed as a progression towards a goal by applying already recorded reasoning patterns. The process knowledge involves 'rules' (patterns) acquired from experts, and 'control knowledge' to make such patterns operational.

Rules: Rules are used for replacing given symbols with a number of other symbols in detailing the function, generating a task from successive states, deciding upon next steps, and adding events, etc. Conventionally, in expert systems such rules are recorded in the form of production rules. Using SDL, we can visualize rules by a number of patterns, such as those shown in Figure 3. In this way we can categorize rules into groups and define templates for encoding rules.

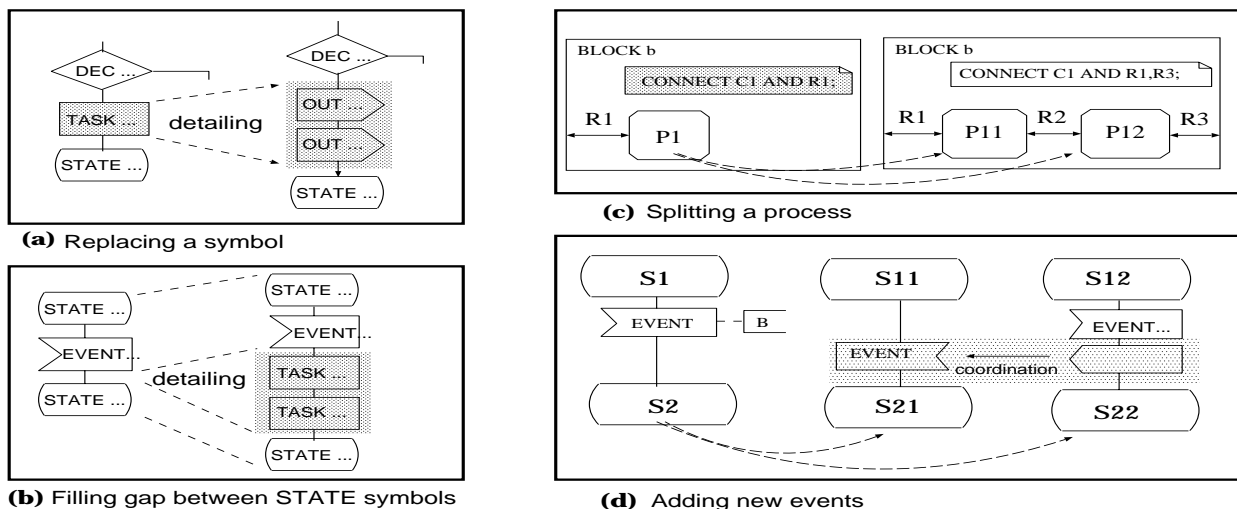


Figure 3 Several rule patterns.

Control knowledge: A main part of human reasoning involves selection and application of rules using control mechanisms. It is believed that control knowledge is neither declarative as in the rules, nor sequential as in the product knowledge. Here each step may be triggered based on a certain symptom. Some recent studies show that this knowledge may be applied in an opportunistic way [13]. The need for a scheme that can integrate the pure top-down approach with the data driven strategy has already been mentioned [2]. This is the meeting point of conventional expert system technology and case based reasoning. Using the method presented in this paper, we can integrate these two.

It should be noted that opposite to the rules, control knowledge does not depend on a particular case and can be implemented independently.

III. APPLICATION: AN EXPERT DESIGN ASSISTANT SYSTEM

A. Overview

In order to give a clear image of the documentation and reasoning techniques used in this research we introduce a research case called *Software Creation* project. In *Software Creation* project automatic software design by following design steps of human designers is studied [7–11]. A family of software CREATOR expert systems are developed [14, 10, 4, 5]. Figure 4 shows their basic features. An application of these systems is assisting a human designer when using a conventional CASE tool. Such CASE tools generally cannot support higher level *knowledge-intensive* activities of design, including knowledge selection, decision making and evaluation. These design activities can be automated using the software CREATOR expert systems. Knowledge base in the CREATOR expert systems is accumulated gradually using the documentation and acquisition

System Name		CREATOR1	CREATOR2	CREATOR3
Language		C, YACC	Knowledge representation language (ES/KERNEL2)	Knowledge representation language (ES/KERNEL2)
Input/output representation	External	Process level based on SDL-PR	CCITT SDL-PR	Frame structure based on SDL-GR
	Internal	Data structure based on SDL-GR	Frame structure based on SDL-GR	Frame structure based on SDL-GR
Type of design rules	Decomposition rules	○	○	×
	Process splitting rules	×	×	○
Input processing		○	○	○
Output processing		○	○	○
Learning		×	○	○

Figure 4 Software CREATOR system family.

technique.

The CREATOR expert systems assist the designer by efficient encoding and reusing the design knowledge. The limited capacity of the short term memory of human designer is enhanced by system's stack, queue and array that have a larger capacity, in principle. In representing the domain concepts and organization of design input/output, frame representation is found useful. Each graphic symbol, commonly used in CASE tools, is represented by a frame and design refinement is nothing but a proper selection of such frames or their child instances and inserting message paths among them. Yet other frames are used to control data access and selection, and the whole structure can cope with the event-driven nature of the design process, ensuring high flexibility of the design and maintaining its rationale.

The CREATOR expert systems are applied to designing switching software. In CREATOR1 the idea of acquisition and application of design rules was elaborated [14]. Then CREATOR2 was built using ES/KERNEL/2 expert system shell [3]. The CREATOR2 system now serves as the platform for design experiments [4]. The knowledge representation and reasoning in CREATOR2 is further elaborated in the CREATOR3 system.

B. Product knowledge

In CREATOR2/3, the designer prepares an initial design sketch using graphic symbols of the SDL CASE tool. In CREATOR2/3, this design sketch is transformed to a structure of *class* and *instance* frames. Each SDL/GR symbol is associated with a frame that embodies all the information related to that symbol, such as its class, function, name, connections, etc. As shown in Figure 5, a design input file, given in SDL, is represented by a structure of such frames.

Frame representation of SDL/GR symbols in the CREATOR2 and CREATOR3 systems is also shown in Figure 5. Human designers usually use comments to save important pieces of information, and retrieve them in later design steps. Furthermore, there are some pictorial elements that are used by the CASE tools that simplify visual interface. For example, in SDT CASE tool in order to represent a *state* symbol, a number of pictorial elements are used [12]. Some of those elements are shown in Figure 5. The CREATOR3 system supports both comments and pictorial elements. The shaded part of Figure 5 shows additional slots that are used for saving data related to comments and pictorial elements.

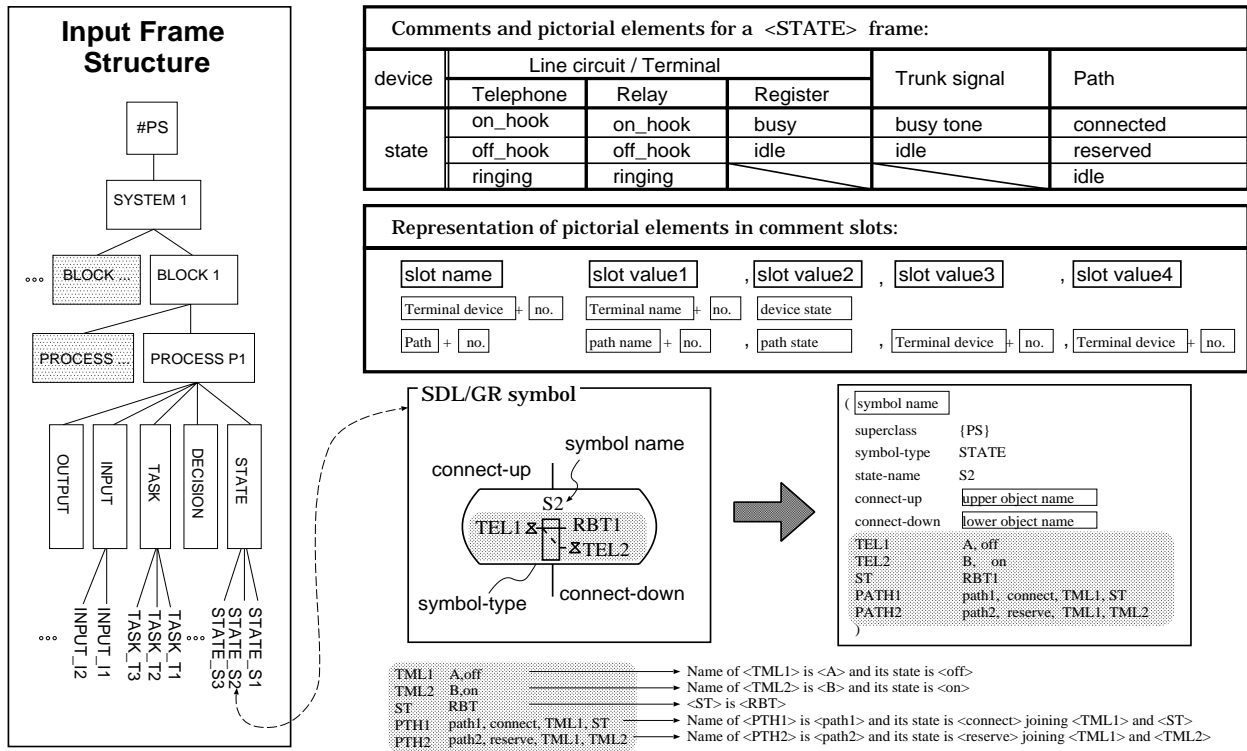


Figure 5 Frame representation of an input file using SDL/GR symbols.

C. Process knowledge

Design rules: Design rules are used for replacing given symbols with a number of other symbols in detailing the function, generating a task from successive states, etc. A method for deriving design rules has been introduced in [8]. The main idea is to follow design steps of human designers, extract their knowledge in an actual design and reuse this knowledge in detailing functions. The design rules are extracted by comparing the design documents in successive design steps. This starts from the most detailed design and goes upward hierarchically to derive all useful detailing patterns [8].

In CREATOR2, the detailing design rules are classified in 4 groups defined by TASK-RULE, DECISION-RULE, OUTPUT-RULE and INPUT-RULE. Figure 6 shows an example of such design rules. Each design rule is composed of a parent frame for the pre-transformation symbol and a number of child frames for post-transformation ones.

Control knowledge: We have developed a collection of 'design schemas' that embody the control knowledge. Each design schema is a coded form of a human design activity. Those schemas are collected and applied in three successive phases.

IV. Experimental Expert System CREATOR2/3

A. System overview

An overview of the CREATOR2/3 system is shown in Figure 7. The human designer prepares an initial design sketch using graphic symbols by a SDL CASE tool. This is converted to SDL/PR and is fed to the CREATOR2/3 system. The SDL/PR is converted to frame structure, suitable for processing by the expert system. The CREATOR2/3 checks if this frame structure can be

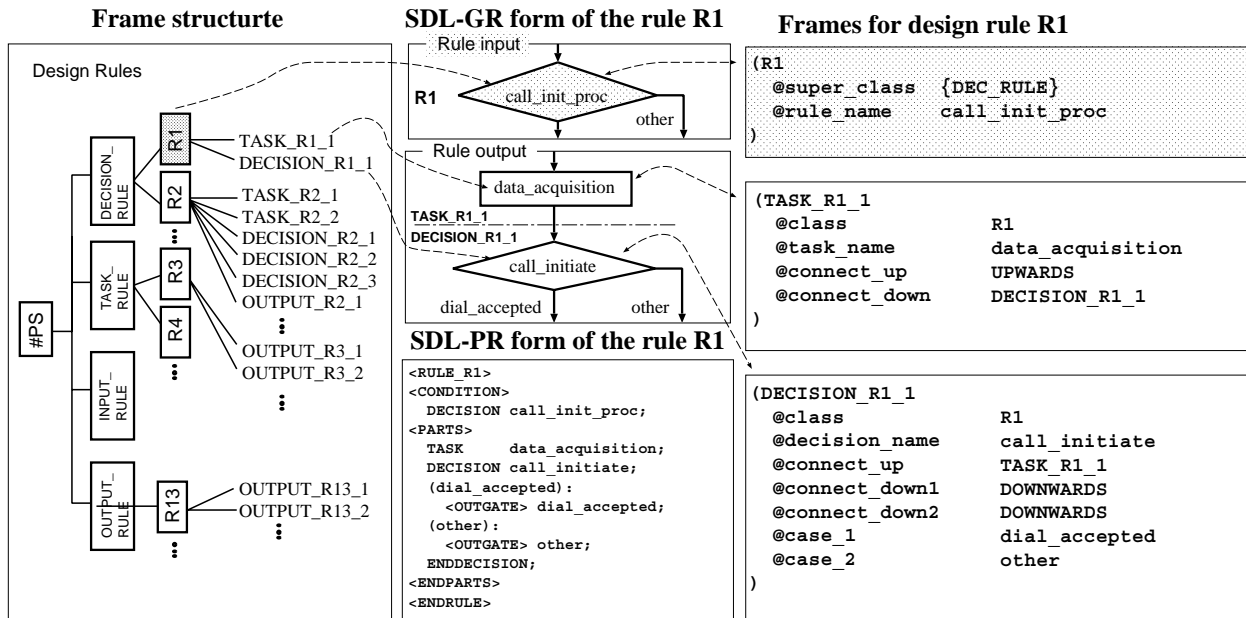


Figure 6 Example of frame representation of a design rule.

detailed by either a design scheme or an already recorded design rules that can be customized to exhibit the required function. The results of detailing and customization are recorded in the created frame structure which is finally converted to SDL/PR. This can be converted to C code by the SDT CASE tool. The designer can check and modify the results, if it is necessary.

The system is implemented on Hitachi 3050 Workstation using ES/KERNEL/2 expert system shell [3] that works together with the SDT CASE Tool [12]. SDT is used for graphical input/output and editing, translation between SDL/PR and SDL/GR, and final conversion to the C code. The other design tasks, conversion, knowledge based reasoning and detailing are performed by CREATOR2/3. The function of expert units in CREATOR2/3 is explained below.

B. Input pre-processing unit

The PR_TO_FRAME expert is used for pre-processing and converting the text based SDL/PR to the frame structure suitable for processing by the CREATOR2/3. Each word of the SDL/PR file is read and interpreted and added to the created frames.

C. Rule pre-processing unit

The design rules are converted to the frame structure using the RULE_TO_FRAME expert. The RULE_TO_FRAME expert receives as its input a design rule in the SDL/PR format and converts it to the frame structure. Figure 6 shows a design rule in SDL/PR form and the generated frames.

D. Creation and reasoning unit

This is the main part of the experimental expert system. There are already two set of frames for the input file and design rules. The CREATION expert is responsible for checking the input frame structure, splitting it and adding events, fetching design rules and inserting the child frames of the matched rules in the input frame structure. All design steps are recorded according to their order of appearance and the system can explain each step if asked to.

Figure 8 shows an example of reasoning by the CREATION unit. Here a frame of the input frame structure is fetched. First, the CREATION unit decides to which system, block or process

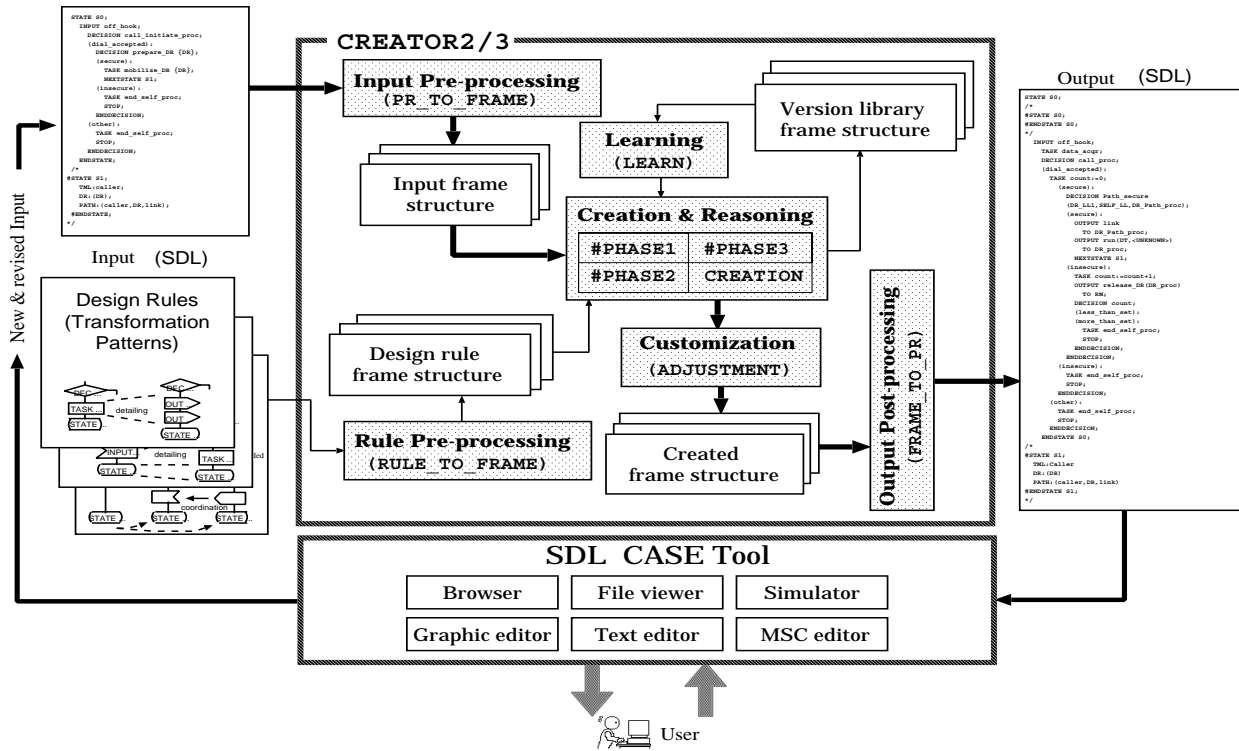


Figure 7 Experimental expert system CREATOR2/3.

this particular frame is belonged. This is done through using a number of routines coded in the methods of this unit. Next, it is examined against the recorded rules. If a match is found, it may be considered for replacing this frame with the other ones indicated in the matched rule. Finally, the frame is copied into the output frame structure while adjusting its links to other frames.

E. Customization unit

The results of creation and reasoning are delivered to the ADJUSTMENT expert which is responsible for customizing the candidate frames and adjusting the links. This is the most time consuming task of automatic design because every single slot of a candidate frame must be checked and all the newly created frames should be accounted for.

F. Learning unit

The LEARN unit keeps record of the design rules that are already used and customized. This is necessary for saving time in similar design cases and when a design rule is applied repetitively.

G. Output post-processing unit

At the end of detailing, the FRAME_TO_PR expert converts the final frame structure to text based SDL/PR that can be used by the SDL CASE tool.

V. DISCUSSION

An interesting outcome of using SDL in documentation and acquisition of knowledge is the ability to revise the knowledge base dynamically while maintaining its rationale. The syntax of SDL and semantics are rich enough to detect inconsistencies and redundancies.

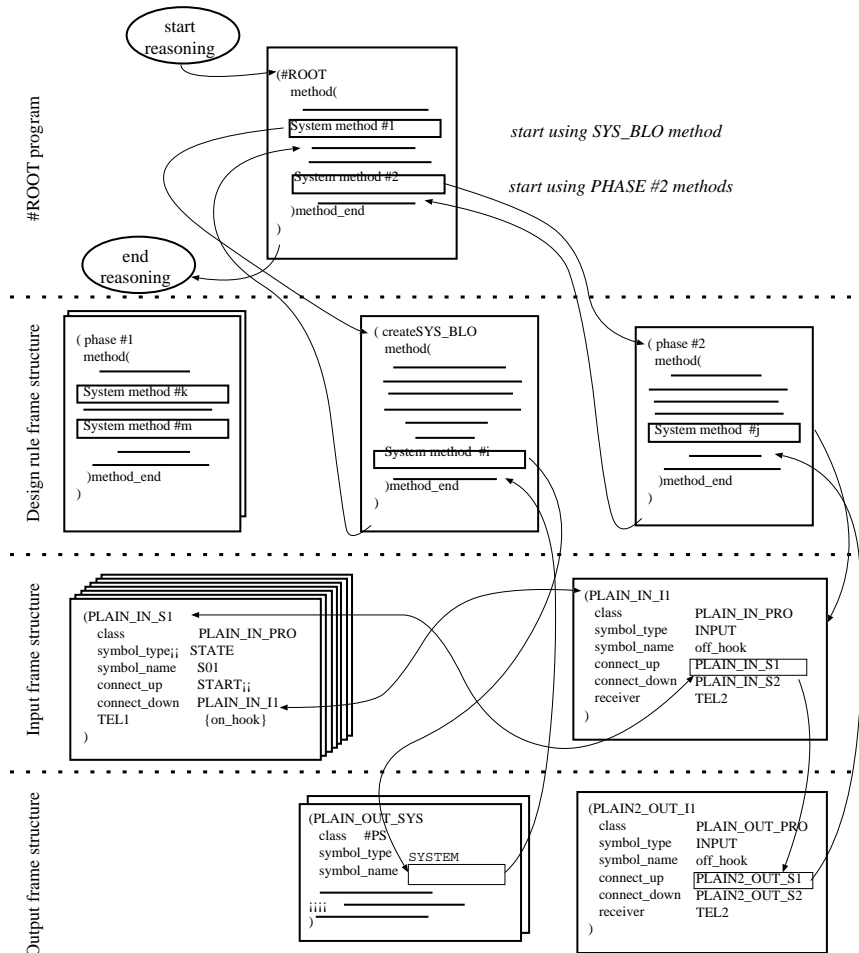


Figure 8 Detailing procedure by the CREATION expert.

Another interesting point is the *learning effect*. We have found that knowledge acquisition through documentation shows learning effect, therefore after accumulating enough rules, the implemented system can perform the job almost automatically. For instance, Figure 9 shows the accumulated number of design rules vs. administration service commands in designing switching administrative program [6, 9]. As is seen the accumulation curve shows rapid increase at first, then the rate decreases gradually though the increase continues. This resembles a learning curve appearing in various kinds of human behavior. At the flat area of the curve, full automatic design is possible by using already extracted design rules. This is repeated when novel group of commands with new features are added. Using this curve, the extra resources required to solve a similar problem can be easily estimated.

VI. CONCLUSION

This paper presents a new approach towards acquisition and implementation of human experts' knowledge. The proposed approach efficiently encodes human knowledge. The problem solving steps of experts is documented and knowledge is extracted by comparing documents in successive phases. In this way, one can encode both the product and process knowledge within a unified framework and reuse them in novel cases. In the knowledge base, we have distinguished between the rules and control knowledge. The former is domain oriented and derived from actual

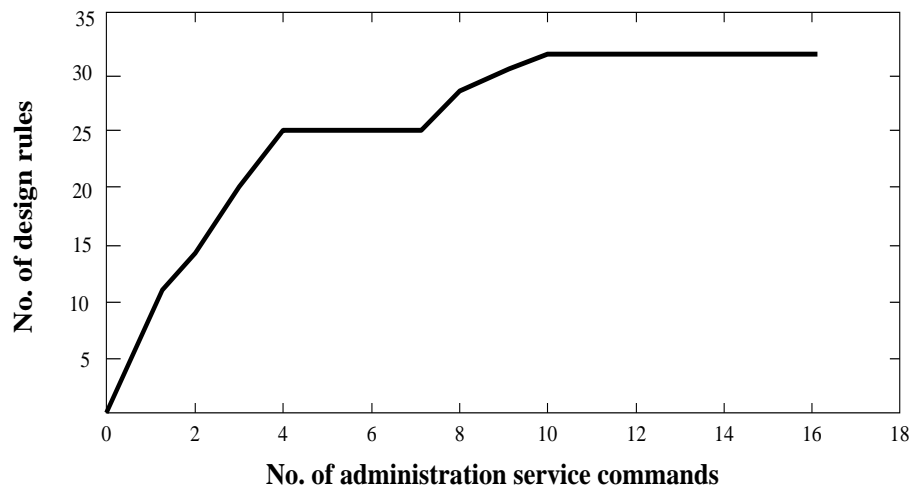


Figure 9 Learning effect of rules vs. service commands.

cases. The latter is general and can be used when dealing with other case. Performance curve of the system indicates extra resources required for solving similar or new problems.

Experiment on implementing an expert system for software design is reported. The CREATOR2/3 system is currently a domain-specific program synthesis system. It serves as an experimental platform for the study of human design.

Presently we are working on knowledge acquisition for a team of human experts. Using SDL makes it easier to bring the basic concepts closer and the documents serve as a common pool of knowledge. However, the need to encode the *viewpoints* and a common ground for resolving conflicts when different viewpoints are present is a problems left to be solved.

Acknowledgements

We wish to express our gratitude to those who contributed to this project, specially former students of the CIT Lab of Department of Information and Computer Sciences, Saitama University, who developed parts of the experimental system.

REFERENCES

- [1] *CCITT Recommendation Z.100, Specification and Description Language (SDL)*, ITU, Geneva, 1992.
- [2] S.P. Davies and F. Simplicio-Filho, "Opportunistic and Goal Oriented Behavior in Software Design," in *Artificial Intelligence in Design' 92*, J.S. Gero, ed., Kluwer Academic Publishers, 1992, pp. 839-860.
- [3] *ES/KERNEL/2W-BS Reference Manual*, Hitachi, 1991.
- [4] B.H. Far, T. Takizawa and Z. Koono, "Software Creation: An SDL-Based Expert System for Automatic Software Design," *SDL '93: Using Objects*, O. Færgemand and A. Sarma, eds., pp. 399-410, Elsevier Publishing Co., North-Holland, 1993.
- [5] B.H. Far, T. Takizawa and Z. Koono, "Software Creation: An Expert System for Reproducing Human Cognitive Processes in Automatic Software Design," in *Proc. World Congress on*

Expert Systems' 94, Estoril, Lisbon, Portugal, January 1994.

- [6] K. Hatae, B.H. Far and Z. Koono, “ソフトウェアクリエーション-交換運用プログラムの設計ルール,” (Software Creation - Design Rules of Switching Administration Program), (in Japanese), in *Proc. IEICE Spring Conf.*, 1993.
- [7] Z. Koono, T. Baba, T. Yabuuchi, Y. Naito, Y. Shigemori and K. Miya, “Software Creation: Systematic Approach,” in Technical Report of IEICE, KBSE92-28, pp. 33-40, 1992.
- [8] Z. Koono, T. Baba and T. Yabuuchi, “Software Creation: A Trial for Switching software,” in *Proc. 5th JC-CNSS, 1992 Joint Conf. on Communications, Networks, Switching Systems and Satellite Communications*, Kyungju, Korea, 1992, pp. 261-265.
- [9] Z. Koono, B.H. Far, T. Baba, Y. Yamasaki, M. Ohmori, and K. Hatae, “Software Creation: Towards Automatic Software Design by Simulating Human Designers,” in *Proc. 5th Int. Conf. on Software Eng. and Knowledge Eng., SEKE' 93*, CA, USA, June 1993, pp. 327-331.
- [10] Z. Koono, B.H. Far, T. Takizawa, M. Ohmori, K. Hatae and T. Baba, “Software Creation: Implementation and Application of of Design Process Knowledge in Automatic Software Design,” in *Proc. 5th Int. Conf. on Software Eng. and Knowledge Eng., SEKE' 93*, CA, 1993, pp. 332-336.
- [11] Z. Koono, B. H. Far, T. Baba, Y. Yamasaki and M. Ohmori, “Software Creation: A Software Engineering Aspect,” in *Proc. Joint Conference on Software Engineering JCSE' 93*, Fukuoka, Japan, November 1993, pp. 289-296.
- [12] SDT CASE Tool ver. 2.2 Reference Manual, Telelogic, Sweden, 1992.
- [13] W. Visser, “More or Less Following A Plan During Design: Opportunistic Deviations in Specification,” *Int. J. of Man-Machine studies*, vol. 33, no. 3, pp. 247-278, 1990.
- [14] T. Yabuuchi, “ソフトウェアクリエーション: 接続制御プログラム自動生成系の基礎研究,” (Software Creation - A Fundamental Research on Automatic Design of the Switching Control Program), (in Japanese), Technical Report ICS-92B-031, Dept. of Info. and Comp. Sci., Saitama University, 1991.