

Merging CASE tools with knowledge-based technology for automatic software design

Behrouz H. Far^{*}, Mari Ohmori, Takeshi Baba, Yasukiyo Yamasaki, Zenya Koono

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

Abstract

An approach towards developing a Knowledge Based Software Engineering (KBSE) tool by merging a conventional CASE tool with the expert system technology is introduced. This is found useful in assisting human designers. Experimental expert systems CREATOR2 and CREATOR3 are introduced and applied to the design of switching software. The CREATOR2 has the following features: representing software design knowledge, composed of *design product knowledge* and *design process knowledge*, using frame technology; and integrating knowledge based reasoning techniques with a SDL CASE tool. CREATOR3 is an extension of the CREATOR2 system. It enables one with additional design schemas for splitting a process, adding events, etc., and additional representation power, such as using pictorial elements and designers' comments in the frame representation. This leads to a uniform modeling and advanced reasoning environment for software design. Experiments on designing switching software are reported.

Keywords: Software design; Expert system; Specification description language (SDL); CASE tool

1. Introduction

In *Software Creation* project automatic software design by the following design steps of human designers is studied [11–15]. A family of software CREATOR expert systems is developed [14,6–9]. 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.

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 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 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 while maintaining its rationale.

^{*} Corresponding author. Fax: +81-48-858-3716; E-mail: far@cit.ics.saitama-u.ac.jp.

The CREATOR expert systems are applied to designing switching software. In CREATOR1 the idea of acquisition and application of design rules was elaborated [1]. Then CREATOR2 was built using ES/KERNEL/2 expert system shell [5]. The CREATOR2 system now serves as the platform for design experiments [6]. Knowledge representation and reasoning in CREATOR2 is further elaborated in the CREATOR3 system. Fig. 1 depicts the basic idea and corresponding design phases in these two latter systems. This paper describes the combined CREATOR2/3 system. Basic features and successive design phases are introduced in detail in the following sections.

2. Knowledge representation in CREATOR2/3

Software design involves two knowledge categories, namely, *design product knowledge* and *design process knowledge*. Here we concentrate on representation of software design knowledge in reusable form, in a structure that integrates these two knowledge categories.

2.1. Design product knowledge

The *design product knowledge* relies on the perspective that the design system is viewed. It includes domain-specific concepts and constraints. The Speci-

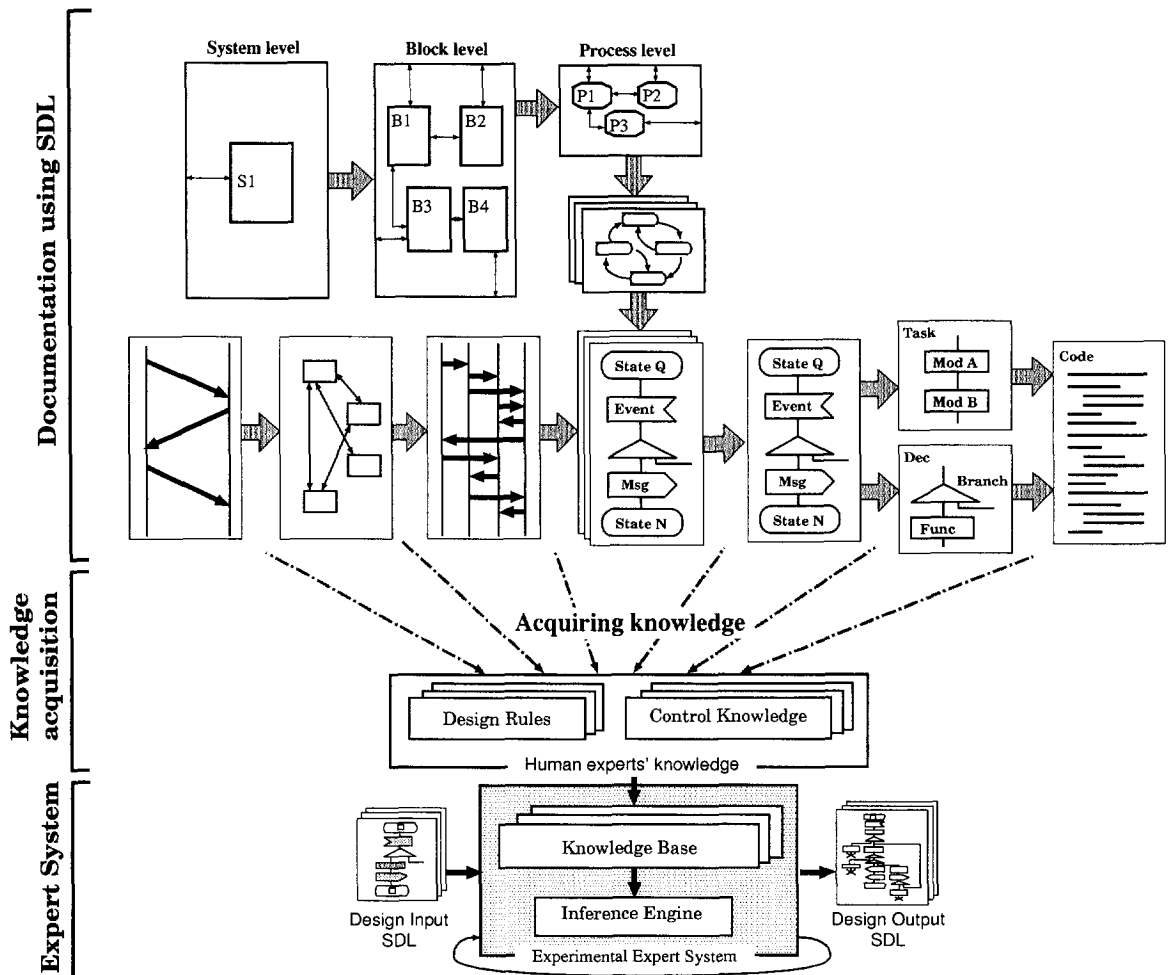


Fig. 1. Successive design phases in CREATOR2/3 systems.

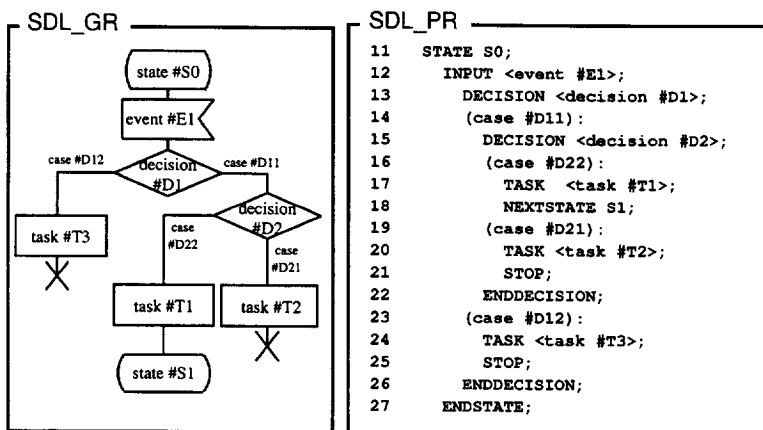
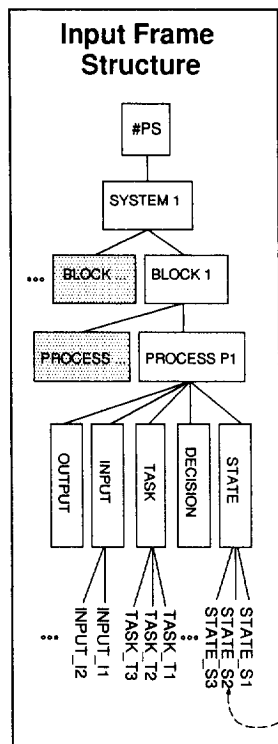


Fig. 2. Example of SDL/GR and SDL/PR.

ification and Description Language (SDL) [3] is used for representing the design product knowledge. In SDL a system is viewed as a collection of “blocks” embodying concurrent “processes”. A process is

represented by an *Extended Finite State Machines (EFSM)* that communicates with other processes through discrete signals. SDL has both graphic (SDL/GR) and text-based (SDL/PR) versions. Fig.



Comments and pictorial elements for a <STATE> frame:

device	Line circuit / Terminal			Trunk signal	Path
	Telephone	Relay	Register		
state	on_hook	on_hook	busy	busy tone	connected
	off_hook	off_hook	idle	idle	reserved
	ringing	ringing			idle

Representation of pictorial elements in comment slots:

slot name slot value1 slot value2 slot value3 slot value4

Terminal device + no. Terminal name + no. device state

Path + no. path name + no. path state Terminal device + no. Terminal device + no.

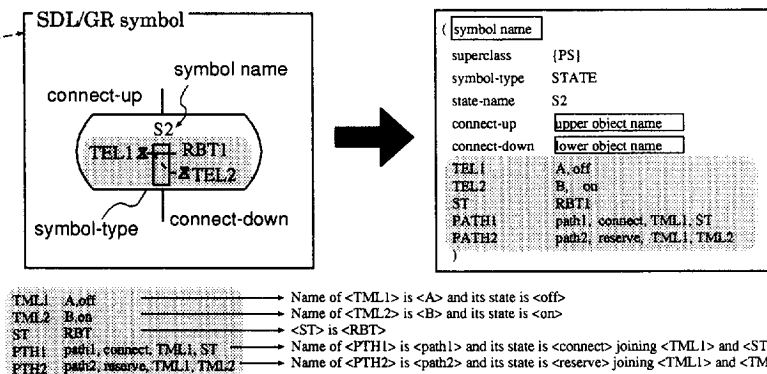


Fig. 3. Frame representation of a SDL/GR symbol.

2 shows an example of SDL/GR and SDL/PR. Some CASE tools using SDL have already been appeared on the market, e.g., SDT [17].

A 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 automatically. 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 Fig. 3, 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 Fig. 3. 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 is used [17]. Some of those elements are shown in Fig. 3. The CREATOR3 system supports both comments and pictorial elements. The shaded part of Fig. 3 shows additional

slots that are used for saving data related to comments and pictorial elements.

2.2. Design process knowledge

Design process is viewed as a progression towards a goal by applying detailing patterns. The design process knowledge involves "design rules" acquired from human design, and "tacit knowledge" to make such patterns operational.

2.2.1. 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 [1,12]. The main idea is to follow design steps of human designers, extract their knowledge in an actual design case and reuse this knowledge in similar cases. This requires proper documentation of the design steps. SDL is used to generate design documents. On the next step, 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

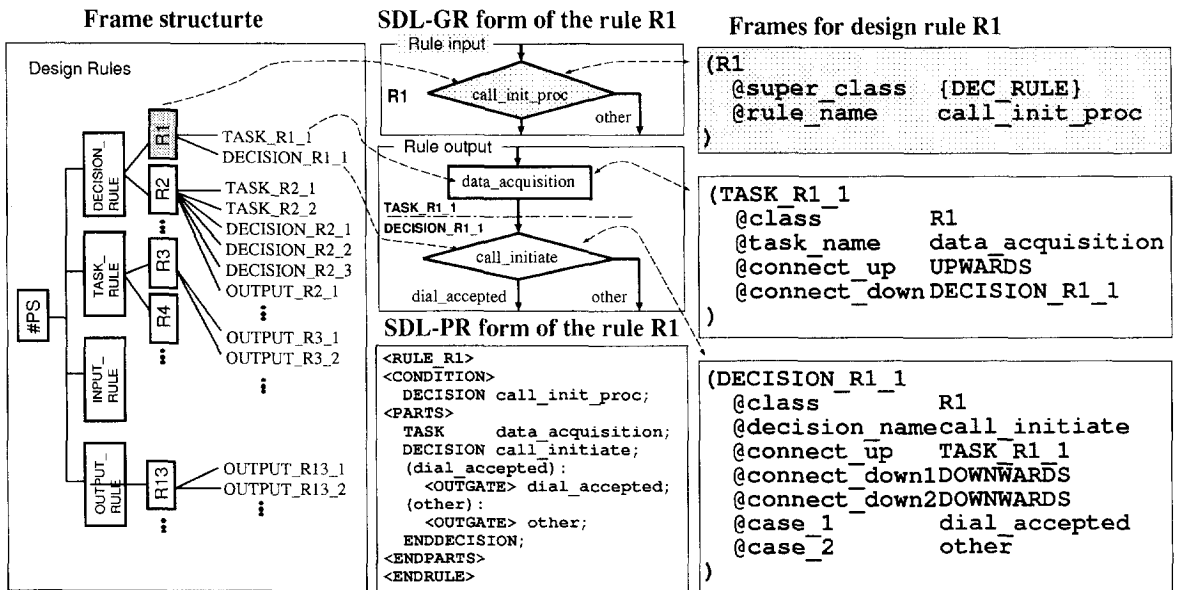


Fig. 4. Example of frame representation of a design rule.

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 SDL CASE tool. The designer can check and modify the results, if it is found necessary.

The system is implemented on Hitachi 3050 Workstation using ES/KERNEL/2 expert system shell [5] that works together with the SDT CASE Tool [17]. 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.

3.2. Input pre-processing unit

The PR_TO_FRAME unit is used for pre-processing and converting the text based SDL/PR to the frame structure suitable for processing by the CREATOR2/3. This is essentially a simple parser for SDL/PR. Each word of the SDL/PR file is analyzed and interpreted and added to the created frames.

3.3. Rule pre-processing unit

The design rules are converted to the frame structure using the RULE_TO_FRAME unit. This is also a parser for design rules. The RULE_TO_FRAME unit receives as its input a design rule in the SDL/PR format and converts it to the frame structure. Fig. 4 shows a design rule in SDL/PR form and the generated frames.

3.4. Creation and reasoning unit

This is the main part of the experimental expert system. There are already two sets of frames for the input file and design rules. The CREATION and Phase#1–#3 units are 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, etc.

We have devised two methods for applying design rules, i.e. *inter-process detailing* and *intra-process detailing*. In inter-process detailing, an instance of a new SDL process is created and detailing is performed based on the information obtained from the initial process. In intra-process detailing, design

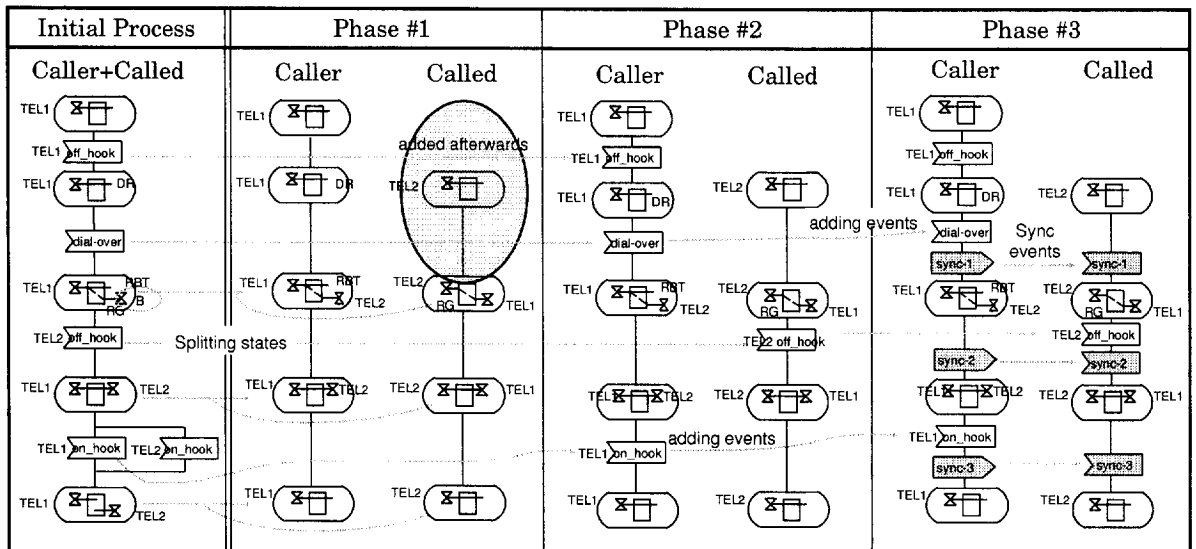


Fig. 6. Process splitting procedure.

rules are applied to a given SDL process. For instance, a graphical SDL/GR symbol is replaced by a collection of other symbols that exhibit the same function in more detail within the same SDL process with some elaboration or fine tuning to fit it into the specific case.

Inter-process detailing is realized by a collection of design schemas. Fig. 6 shows a SDL view of splitting a process, creating new processes and adding events. This is programmed in three successive phases. The idea is moving from an existing process to a more flexible one composed of mono functional processes. As it is shown, PHASE#1 splits an input process, i.e., “Caller” + “Called”, into two new processes, i.e., “Caller” and “Called”. At the first step, the STATE symbols are copied. As every state must be followed by an event, the events of the initial process are inserted in the newly created processes. This is done in PHASE#2. Finally, in PHASE#3 synchronizing events are added to make the new processes compatible with the original one. In this case, the “sync-i” message is sent by the “Caller” and acknowledged by the “Called” process.

Other steps of fetching and applying design rules are coded in the *methods* of the CREATION unit. There are *methods* for fetching a frame, fetching a design rule, comparing, deciding upon accepting or rejecting a rule, writing to the output frame structure and optimizing the search. For example, a SDL symbol, represented by a frame belonging to the input frame structure, is examined against the already recorded design rules. If the matching attributes are found, that frame is deleted from the input frame structure and those child frames of the matched rule are copied to the output frame structure. All design steps are recorded at the background according to their order of appearance and the system can explain each step, by popping up the detailed reasoning record and log files, if asked to.

3.5. 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.

3.6. 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.

3.7. 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.

4. Experimental results

Switching software is considered as the problem domain and studied in three areas: switching control, operational (administration) and protocol subsystems, to cover the major spectrum of switching software. The central part of switching control program is chosen as a typical example. Experiments for designing the switching control program for the Plain Ordinary Telephone Service (POTS), Full-Call-Back-Transfer (FCBT), etc., have already been reported [16,14,6,7]. During design by the CREATOR2/3 system, the input file is detailed around 6–10 times. Then the SDT converts it to C-codes of 10 times number of lines, resulting in 60–100 times code expansion. Other experiments for designing a switching administration program are reported in [10] and [19].

5. Discussion

Opposite to the common knowledge engineering method for knowledge acquisition, in this project, we focus on knowledge documentation. This is necessary to produce quality knowledge. Then, knowledge is acquired, in the form of *rules*, by comparing the documents in successive steps of design. Later, *tacit* knowledge is used for selecting and applying such

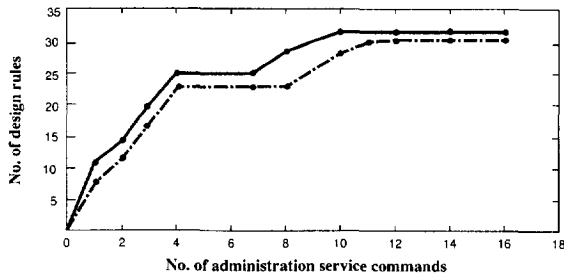


Fig. 7. Learning effect of rules vs. service commands.

rules. In some other research on automatic software design the need for distinguishing between the design product knowledge and design process knowledge is mentioned [2,18]. We have proposed a unified framework for representing both the design process and the design product knowledge. We have distinguished between the design rules and the tacit knowledge of the design process. The former is domain oriented and derived from actual design. The latter is general and can be used in design of software other than switching software.

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 its semantics is rich enough to detect inconsistencies and redundancies.

Another interesting point is the *learning effect*. We have found that knowledge acquisition through documentation shows a learning effect, therefore after accumulating enough rules, the implemented system can perform the job almost automatically. For instance, Fig. 7 shows the accumulated number of design rules vs. administration service commands in designing a switching administrative program [4]. As is seen the accumulation curve shows a rapid increase at first, then the rate decreases gradually though the increase continues. This resembles a learning curve that appears in various studies 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 a new group of commands with new features is added. Using such a curve, extra resources required to solve a similar problem can be estimated.

6. Conclusion

This paper presents an implementation of a software design expert system that assists human designers. The proposed system provides the user with methods for acquiring design knowledge, representing both design comments and pictorial elements, and reusing this knowledge in novel cases. Experiments on developing switching software are 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. We are currently adding a user-friendly interface to the system, developing a multi-agent design system based on thus mentioned architecture, and building similar systems for other areas of engineering design.

Acknowledgements

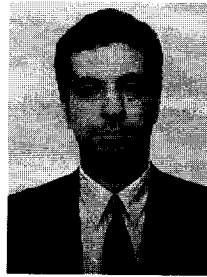
Authors thank the Information and Telecommunications Division, Hitachi, Ltd., for supporting the switching administration area. The Telecommunications Advancement Foundation for supporting the switching control area, and Information Systems Division, Hitachi, Ltd., for supporting the expert system area. The program described in this paper is the result of research at the CIT Lab of the Department of Information and Computer Sciences, Saitama University, since 1991. Many of our students have contributed to this project. We are grateful to them collectively.

References

- [1] T. Baba, K. Miya, T. Yabuuchi, Y. Shigemori, Y. Naito and Z. Koono, Software Creation: The First Results, in: Proceedings of the IEICE Fall Conference, Tokyo (1992) pp. 6420–421.
- [2] S. Bhansali and H.P. Nii, KASE: An Integrated Environment for Software Design, in: J.S. Gero, Ed., Artificial Intelligence in Design '92 (Kluwer Academic Publishers, 1992) pp. 371–389.
- [3] CCITT Recommendation Z.100, Specification and Description Language (SDL), ITU, Geneva (1992).

- [4] H. Chen, B.H. Far and Z. Koono, Software Creation: Reuse of Design Knowledge of Switching Software, in: Proceedings of the International Conference on Communication Technology (ICCT '94), Shanghai, China (June 1994) pp. 63–66.
- [5] ES/KERNEL//2W-BS Reference Manual, Hitachi (1991).
- [6] B.H. Far, T. Takizawa and Z. Koono, Software Creation: An SDL-Based Expert System for Automatic Software Design, in: O. Faergemand and A. Sarma, Eds., *SDL '93: Using Objects* (Elsevier Publishing Co., North-Holland, Amsterdam, 1993) pp. 399–410.
- [7] B.H. Far, T. Takizawa and Z. Koono, Software Creation: An Expert System for Reproducing Human Cognitive Processes in Automatic Software Design, in: Proceedings of the World Congress on Expert Systems '94, Estoril, Lisbon, Portugal (Jan 1994).
- [8] B.H. Far and Z. Koono, CREATOR2 Reference Manual – Version 1.1 (July 1993).
- [9] B.H. Far and Z. Koono, CREATOR3 Reference Manual-Version 1.0 (September 1993).
- [10] K. Hatae, B.H. Far and Z. Koono, Software Creation – Design Rules of Switching Administration Program (in Japanese), in: Proceedings of the IEICE Spring Conference (1993).
- [11] Z. Koono, T. Baba, T. Yabuuchi, Y. Naito, Y. Shigemori and K. Miya, Software Creation: Systematic Approach, Technical Report of IEICE, KBSE92-28 (1992) pp. 33–40.
- [12] Z. Koono, T. Baba and T. Yabuuchi, Software Creation: A Trial for Switching software, in: Proceedings of the 5th JC-CNSS, 1992 Joint Conference on Communications, Networks, Switching Systems and Satellite Communications, Kyungju, Korea (1992) pp. 261–265.
- [13] 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: Proceedings of the 5th International Conference on Software Engineering and Knowledge Engineering, SEKE '93, CA, USA (June 1993) pp. 327–331.
- [14] 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, in: Proceedings of the 5th International Conference on Software Engineering and Knowledge Engineering, SEKE '93, CA (June 1993) pp. 332–336.
- [15] Z. Koono, B.H. Far, T. Baba, Y. Yamasaki and M. Ohmori, Software Creation: A Software Engineering Aspect, in: Proceedings of the Joint Conference on Software Engineering, JCSE '93, Fukuoka, Japan (Nov 1993) pp. 289–296.
- [16] M. Ohmori, B.H. Far and Z. Koono, Software Creation – Design Rules of Switching Service Program (in Japanese), in: Proceedings of the IEICE Spring Conference (1993).
- [17] SDT CASE Tool version 3.0 Reference Manual, Telelogic, Sweden (1995).
- [18] J. Treur and P.J. Veerkamp, Explicit Representation of Design Process Knowledge, in: J.S. Gero, Ed., *Artificial Intelligence in Design '92* (Kluwer Academic Publishers, 1992) pp. 677–696.

- [19] Y. Yamasaki, B.H. Far and Z. Koono, Software Creation – A Fundamental Study on Data Access Program (in Japanese), in: Proceedings of the IEICE Spring Conference (1993).



Behrouz Homayoun Far received his BSc. and MSc. degree in Electronic Engineering in 1983 and 1986, respectively, from Teheran University, Iran. He has received his Ph.D. degree from Chiba University, Japan, in 1990. He was awarded a post doctoral fellowship from the Japanese Science and Technology Agency and joined the Japanese Atomic Energy Research Institute (JAERI) as a research fellow. He is currently a Associate Professor at the

Department of Information and Computer Sciences, Saitama University. The research fields of his interest are qualitative and temporal reasoning, Knowledge acquisition and distributed AI. Dr. Far is a member of the Association for Computing Machinery, IEEE Computer society, Japanese Society for Artificial Intelligence, and Information Processing Society of Japan.



Mari Ohmori received her B.E. and M.E. degree in Information Engineering in 1993 from Saitama University, Japan. She is currently working at Systems and Software Engineering Laboratory, Toshiba Corporation. Her research fields of interest are testing techniques and object-oriented software. Ms. Ohmori is a member of the Institute of Electronics, Information and Communication Engineers of Japan.



Takeshi Baba received his B.E. and M.E. degree in Information Engineering from Saitama University, Japan, in 1992 and 1994. In 1994 he joined Hitachi Ltd., Telecommunication Division. He has been engaged in designing and programming ATM switching software. He takes interests in CASE tools and architecture for switching software.

Yasukiyo Yamasaki received his B.E. and M.E. degree in Information Engineering from Saitama University, Japan, in 1992 and 1994. He is Currently with the Fujitsu, Ltd.



Zenya Koono received his Bachelor of Engineering, Master of Engineering, and Doctor of Engineering, all from the University of Tokyo, in 1959, 1961 and 1964, respectively. In 1964 he joined Hitachi, Ltd., and had been engaged in research and development as well as design of electronic telephone switching systems ranging from switching peripheral hardware, central processors, CPU architecture, design automation of automatic printed circuit board, switching

software, switching systems and Total Quality Control. Since 1991 he is a Professor at the Department of Information and Computer Sciences, Saitama University. His present research fields are software engineering for industrial applications and intelligent systems using human knowledge. In 1963, he received the Inada Memorial Award and had been an elected member of the editorial board of the Institute of Electronics, Information and Communication Engineers. Professor Koono is a member of the JARL, Information Processing Society of Japan and Japanese Society for Artificial Intelligence.