

# SDL '93 USING OBJECTS

*Proceedings of the Sixth SDL Forum  
Darmstadt, Germany, 11 – 15 October, 1993*

*edited by*

Ove FÆRGEMAND  
*EURESCOM  
Heidelberg, Germany*

Amardeo SARMA  
*Deutsche Telekom  
Darmstadt, Germany*



1993

NORTH-HOLLAND  
AMSTERDAM • LONDON • NEW YORK • TOKYO

## Software Creation: An SDL-Based Expert System for Automatic Software Design

Behrouz Homayoun Far, Takeshi Takizawa, 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

A goal of this project is reproducing human design process by accumulating knowledge and experience of human designers. Particularly, this paper presents an SDL-based software design tool, the experimental expert system CREATOR2, featuring: 1) integration of SDL-based CASE tools with knowledge-based reasoning techniques; 2) object-oriented (O-O) representation of the design process knowledge, composed of *design rules* for detailing, and *tacit knowledge*; 3) O-O representation of the SDL/GR symbols in the knowledge-base; 4) using multiple strategies in applying the design process knowledge; and 5) O-O implementation of the system. This leads to having a uniform modeling and advanced reasoning environment for software design. Experiments on designing switching software are reported. Presently, the CREATOR2 system together with an SDL CASE tool offers 60-100 times code expansion rate.

### 1. INTRODUCTION

In this paper we report on an expert system for supporting knowledge intensive tasks of software design. The focus is on implementing a system that can support the *design process knowledge*, in order to capture and reason with the sort of knowledge that human experts use in design.

(1) Conventional CASE tools generally facilitate requirement analysis, detailed design and code generation. Their ultimate goal is to improve the productivity of design by assisting the designer. Such CASE tools cannot support the higher level *knowledge-intensive* activities of design unless incorporated with the knowledge representation and sophisticated reasoning methods [14]. Presently, higher level design activities, such as those related to selecting proper pieces of knowledge, decision making and evaluation, are to be handled by human designers. The software design knowledge can be captured and implemented in the CASE tools using Artificial Intelligence (AI) techniques. This paper introduces an approach towards developing a Knowledge Based Software Engineering (KBSE) tool using SDL-based CASE tool and object-oriented (O-O) techniques.

(2) There are two knowledge categories involved in design, addressing *design product* and *design process*. The design product knowledge, consists of the domain-specific concepts and constraints of the task. During intermediate design steps, human experts rely on their

---

\*Authors thank 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.

design process knowledge that is accumulated over years of developing similar software products. The design process knowledge includes certain *design rules* (transformation patterns) for detailing and *tacit knowledge* for applying the rules. The design process knowledge is hardly documented, maintained and reused.

(3) Human software designers decompose the goal in a top-down manner until reaching a point that a portion of the decomposed structure can be converted to the code [7]. However, in designing complex systems, it becomes increasingly difficult to keep track of all sub-goals. Even expert designers show rather poor performance in keeping track of all intermediate pieces of information and subgoals simultaneously. This is called *tunneling* effect of the *cognitive overload* [6]. Efficient encoding the design process knowledge and applying it can remove the tunneling effect.

(4) In *Software Creation* project [1, 9–13, 17] we study automatic software design by simulating human designers. Figure 1 depicts the idea and key concepts. The main idea is to follow design steps of human designers. Human designers' knowledge is extracted from an actual design and is reused by an expert system. Hierarchical nature of design process is assumed. The design rules are extracted by comparing the design documents in successive design phases, as shown in Figure 1. The tacit knowledge is used for selecting and applying design rules. The research starts from the lowest and the most detailed design and goes upward hierarchically to more knowledge intensive areas. Specification and Description Language (SDL) [3] has been applied successfully in this project.

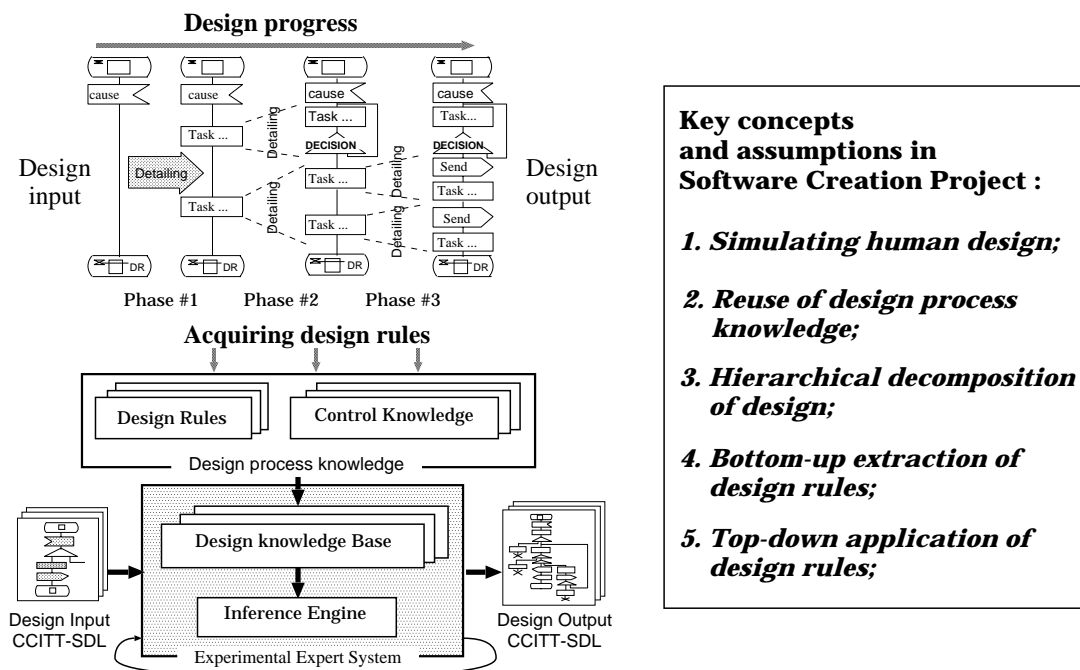


Figure 1. Overview of the Software Creation project.

(5) The experimental expert system, CREATOR2, is built to demonstrate the results and verify the ideas. It is developed based on the idea of integrating different expert units to achieve a common goal. In CREATOR2, object description is central. Each expert

unit is composed of a number of objects. Objects can store information, communicate with the other objects, process information and take part in reasoning using dedicated *methods*. Domain concepts, such as design rules, organization of design input/output, etc., are represented by the frame structure. Frames are special objects that can only store information and communicate with the other objects. Hierarchical representation of domain concepts are described in terms of *class* and *instance* objects. The frame structure in CREATOR2 system corresponds to the hierarchical structure of concepts in SDL.

(6) This paper is structured as follows: Section 2 presents the design knowledge representation in the CREATOR2 system. Section 3 describes the CREATOR2 system in detail and Section 4 gives some experimental results. Finally, Section 5 compares CREATOR2 with the other systems and concludes with a summary of achievements and suggests some future research directions.

## 2. SOFTWARE DESIGN KNOWLEDGE

Here we concentrate on representation and support of software design knowledge, i.e. *design product knowledge* and the *design process knowledge*, in reusable form, particularly by introducing a structure for integrating 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, constraints and models of the system. We have used the Specification and Description Language (SDL) [3] for representing the design product knowledge. SDL has both graphic (SDL/GR) and text-based (SDL/PR) versions. In SDL the system is viewed as a collection of ‘blocks’ embodying concurrent ‘processes’. A process is represented by an *Extended Finite State Machines* (EFSM) [3]. Processes communicate by discrete signals. The capability of SDL to model and design sequential/non-sequential software, and to give a formal description of the design at any level of interest are found very useful in this project. We are also interested in a particular feature of SDL that is the ability to concentrate on the ‘process’ as the basic module of design and exchange messages between processes that corresponds with our O-O view of the system.

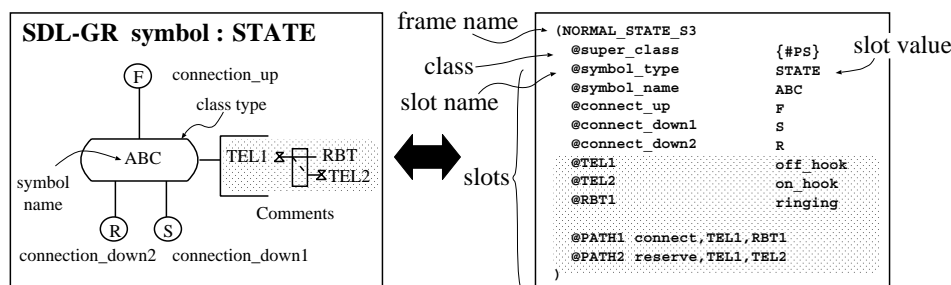


Figure 2. Frame representation of a typical SDL/GR symbol.

In CREATOR2, the design product knowledge is represented by a structure of *class* and *instance* objects (frames). Each SDL/GR symbol is associated with an instance object

that embodies all the information related to that symbol, such as its class, function, name, connections, etc. Figure 2 shows frame representation for the STATE symbol that includes pictorial elements in the comment. A frame has a number of *slots* to record the data. A slot is identified by its name, value and data type. A frame also has a 'class' that corresponds to the SDL/GR symbols, such as STATE, TASK and DECISION. Frames of the same class have common representation templates defined by class objects.

A design input file, in SDL, is represented by a structure of frames in CREATOR2. The frame structure for a simple input file is shown in Figure 3. The upper left part of this figure shows the SDL representation, the lower left part shows the frame structure in CREATOR2 system, and the right part shows the content of each frame corresponding to the SDL/GR symbols.

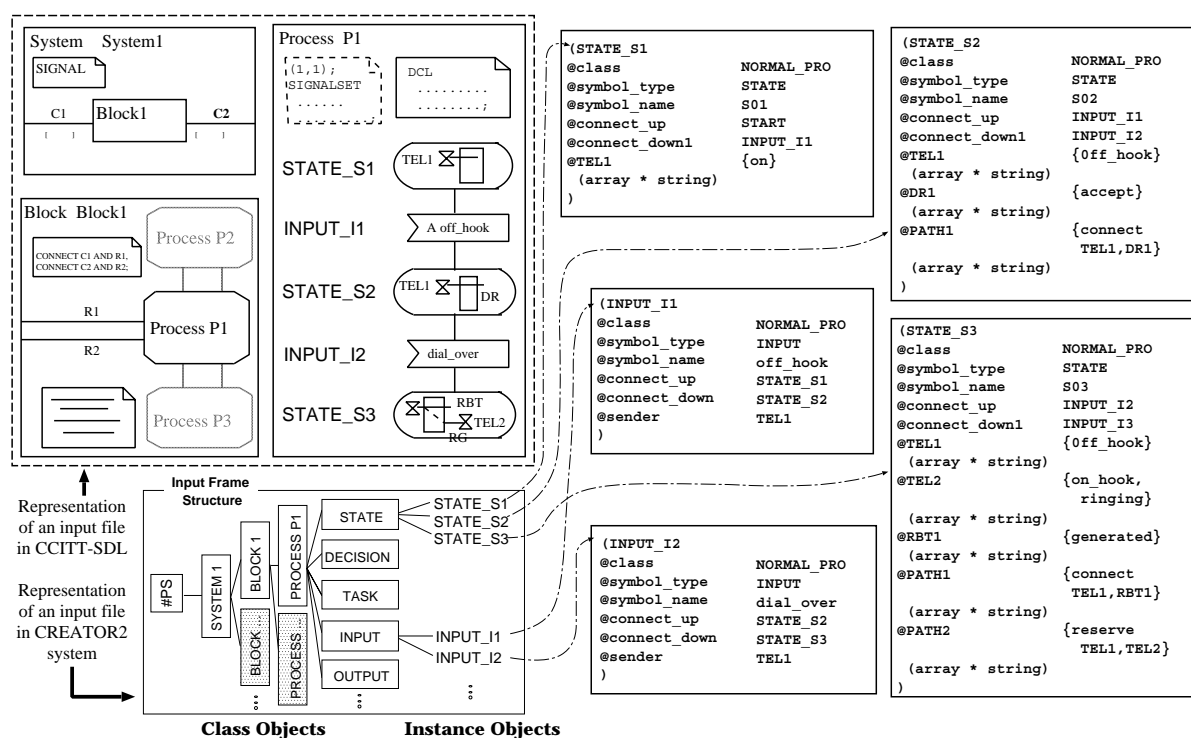


Figure 3. Example of frame representation of an input file.

## 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' (transformation patterns) 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, splitting an SDL process, and adding events, etc. Several patterns for design rules are shown in Figure 4 and a method for deriving design rules has already been introduced [1, 9–11].

In CREATOR2, the detailing design rules are classified in 4 groups defined by TASK-RULE, DECISION-RULE, OUTPUT-RULE and INPUT-RULE. Figure 5 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.

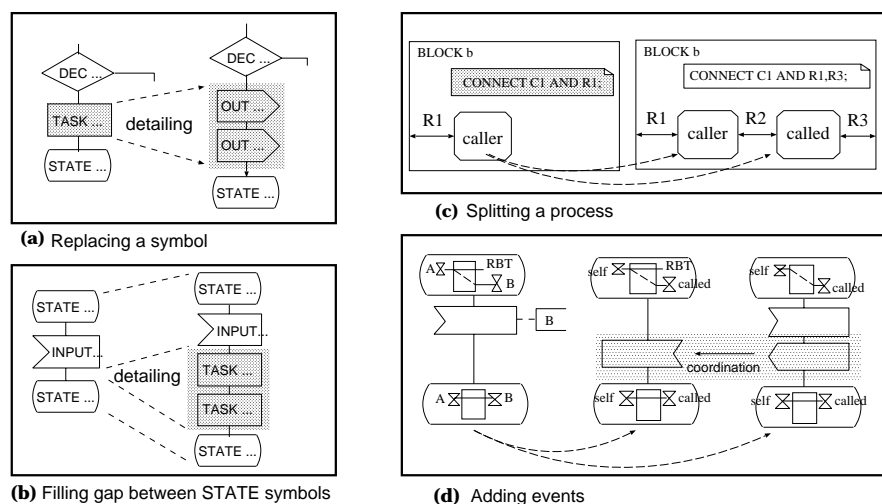


Figure 4. Four patterns for design rules.

There are other design rules that embody certain procedures, and can be coded using objects' *methods*. Such rules are represented by a single object that has some dedicated *methods*. A *method* is a short program embodying the procedure for creating objects, slots, and assigning value of the slots, etc.

### 2.2.2. Tacit knowledge

A main activity in human design involves selection and application of design rules using the tacit knowledge. It is believed that tacit knowledge is neither declarative as in the design rules, nor sequential as in the design product knowledge. Here each step may be triggered based on a certain symptom. Some recent empirical studies show that this knowledge may be applied in an opportunistic way that deviates from the stepwise refinement of design [21]. The need for a scheme that can integrate the pure top-down approach with the data driven strategy has already been mentioned [4].

Opposite to the design rules, tacit knowledge does not depend on a particular design and can be implemented independently. In the CREATOR2, the CREATION expert unit embodies this knowledge (see Section 3.3 for details).

## 3. EXPERIMENTAL EXPERT SYSTEM CREATOR2

### 3.1. Overview

The CREATOR2 system is composed of 6 experts (plus a dedicated #ROOT program). The system is shown in Figure 6. The human designer prepares an initial design sketch using graphic symbols by the CASE tools (in this project the SDL/GR symbols). This is

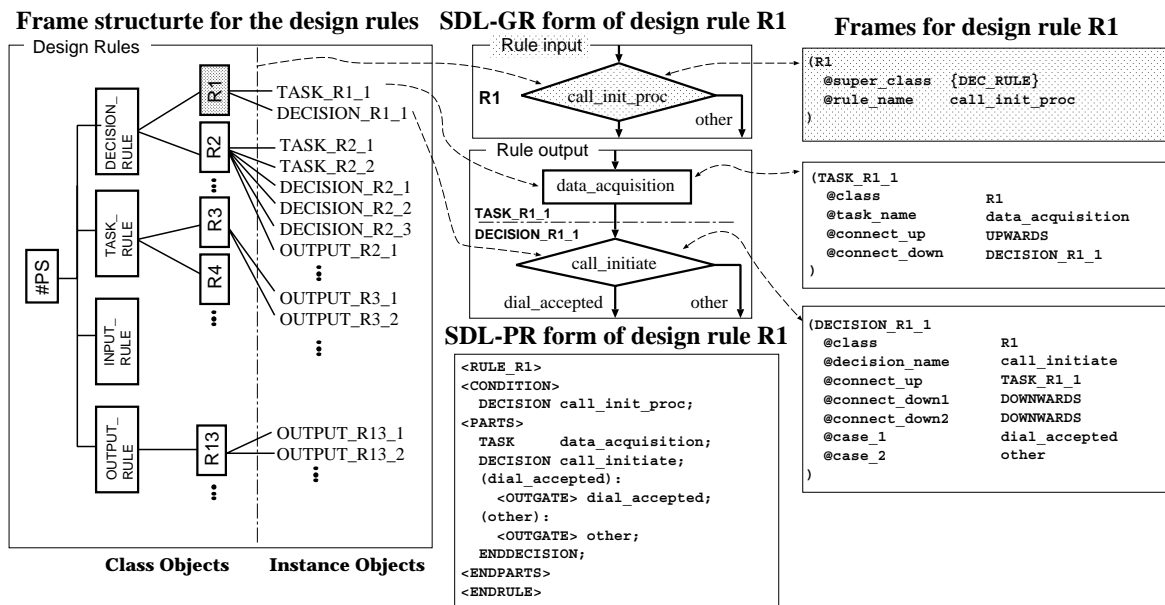


Figure 5. Example of frame representation of a design rule.

converted to SDL/PR by a CASE tool, SDT [19], and fed to the CREATOR2 system. The SDL/PR is converted to frame structure, suitable for processing by the expert system. The CREATOR2 checks if this frame structure can be detailed by already recorded design rules that can be customized to exhibit the required function. The results 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. This ensures high flexibility of the design while maintaining its rationale. This procedure is described in detail in the following subsections.

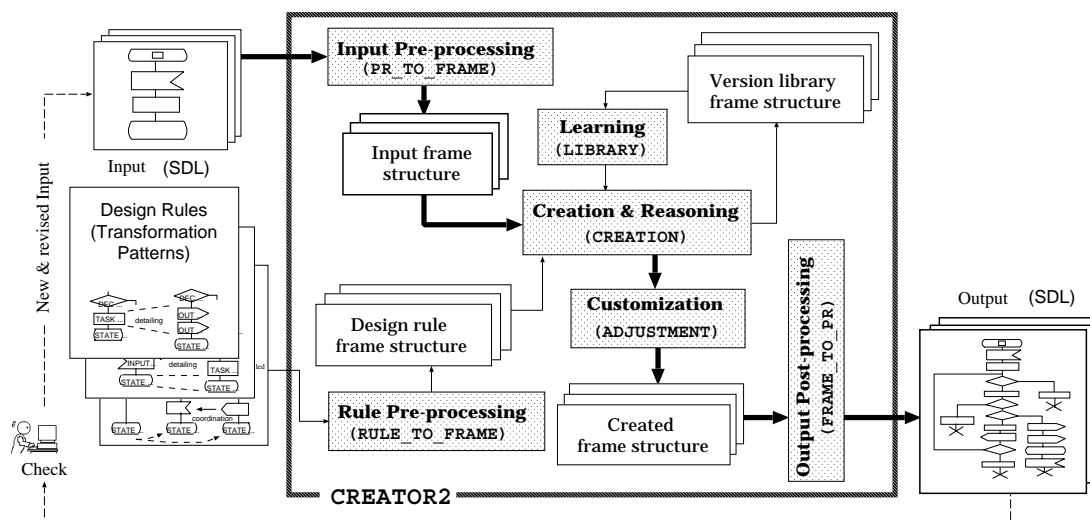


Figure 6. Experimental expert system CREATOR2.

The CREATOR2 is implemented on Hitachi 3050 Workstation using ES/KERNEL/2 expert system shell [5] that works together with the SDT CASE Tool [19]. 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.

The expert units in CREATOR2 fall within three groups: *conversion*, *detailing*, and *other* experts. The number of objects and program size for each unit is given in Table 1. Their function is explained in the following paragraphs.

Table 1. Function and program size of the CREATOR2 expert units.

Expert	Function	LOC	Objects
(#ROOT	Controlling and coordinating other units	450 lines	1)
1 PR_TO_FRAME	Converting SDL/PR to frame structure	770 lines	7
2 RULE_TO_FRAME	Converting design rules to frame structure	400 lines	5
3 CREATION	Selecting and applying design rules	2090 lines	12
4 ADJUSTMENT	Adjusting value of slots of other frames	3100 lines	12
5 LIBRARY	Recording new design rules	1200 lines	5
6 FRAME_TO_PR	Converting frame structure to SDL/PR	840 lines	7

### 3.2. Conversion expert units

(1) The PR\_TO\_FRAME expert is used for preprocessing and converting the text based SDL/PR to the frame structure suitable for processing by the CREATOR2. Figure 7 shows how PR\_TO\_FRAME creates new frames. In this program, each word from the SDL/PR file is read and interpreted and in this case three new frames are created.

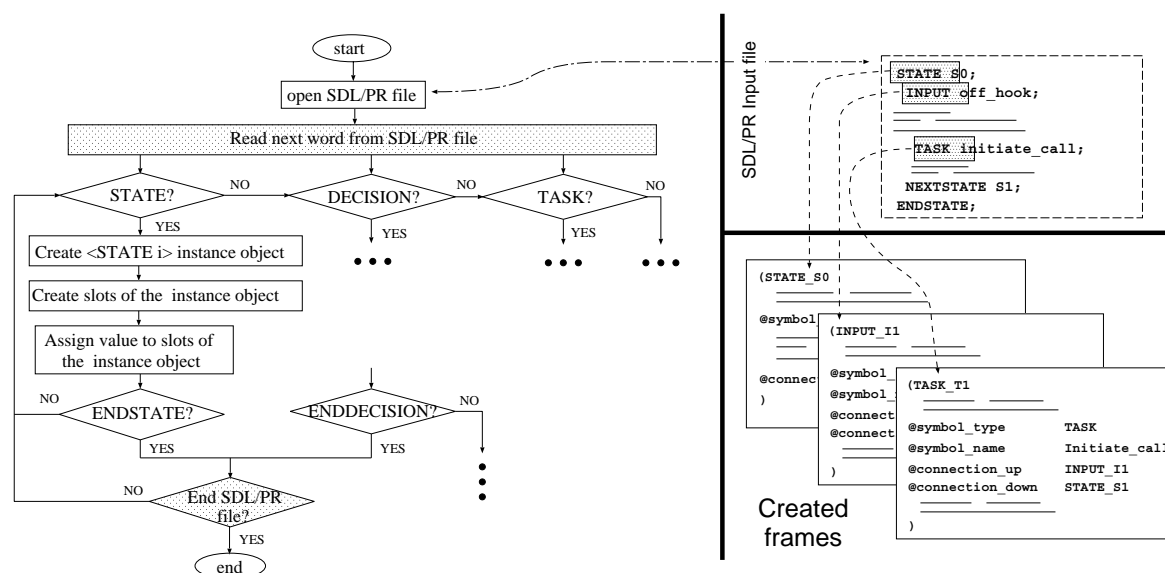


Figure 7. Creation of new frames by PR\_TO\_FRAME program.

(2) The design rules are converted to the frame structure using the RULE\_TO\_FRAME expert. Each design rule is composed of a parent frame representing the pre-transformation

symbol and a number of child frames for post-transformation ones. 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 5 shows a design rule in SDL/PR form and the generated frames.

(3) After detailing is over, the `FRAME_TO_PR` expert converts the final frame structure to text based SDL/PR that can be used by the SDT CASE tool.

### 3.3. Detailing expert unit

The `CREATION` expert stays as the core of the `CREATOR2`. 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, fetching design rules and inserting the child frames of the matched rules in the input frame structure.

In implementing tacit knowledge in the `CREATOR2`, we have distinguished between two strategies for applying design rules, i.e. *intra-process detailing* and *inter-process detailing*. In intra-process detailing, the tacit knowledge is used for applying design rules within a given SDL process. This is shown in the right side of Figure 8. 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 SDL process with some elaboration or fine tuning to fit it into the specific case.

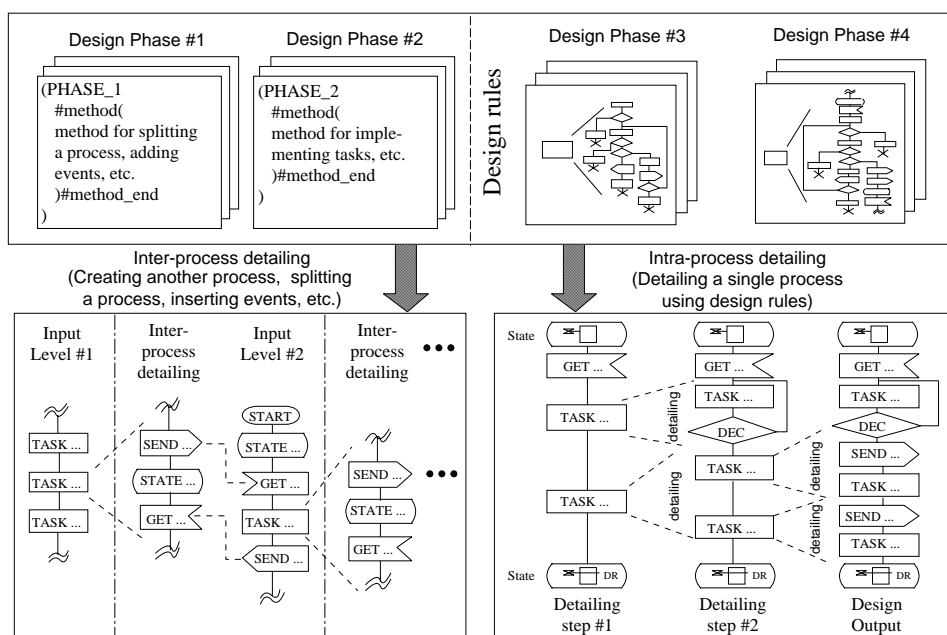


Figure 8. Intra- and inter-process detailing.

On the other hand, in the inter-process detailing, an instance of a new process is created and the detailing is performed based on the information obtained from the initial process. This is shown in the left side of Figure 8.

In intra-process strategy, a local and limited search is sufficient to find the proper design rule. However, in inter-process strategy a look-ahead (e.g., finding the succeeding state and finding match for the events) or look-back (e.g., finding the preceding state, matching the events, add to the block, system, etc.) search strategy are applied.

Various steps of the creation procedure are coded by the *methods* of the objects belonging to the CREATION expert. Figure 9 shows an example of a method for intra-process detailing by the CREATION expert. Here the CREATION expert fetches a frame of the input frame structure, matches its name and other attributes with a design rule, and replaces the input frame with those child frames of the matched rules. All design steps are recorded according to their order of appearance and the system can explain each step if asked to.

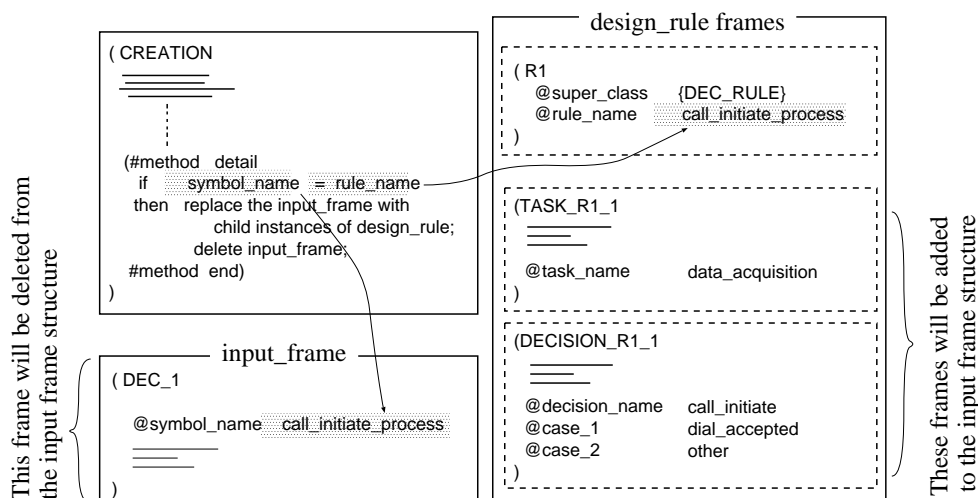


Figure 9. Detailing procedure by the CREATION expert.

### 3.4. Other expert units

(1) The results of creation 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.

(2) The LIBRARY expert 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. The ADJUSTMENT and LIBRARY experts together realize the learning function of CREATOR2.

## 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.

In the experiment for designing the switching control program for the Plain Ordinary Telephone Service (POTS), we start with a single SDL process, describing 'call' (caller and called) behavior. The system then splits it to two 'caller' and 'called' processes [17]. These two processes are further detailed to elemental tasks and decisions. Figure 10/a and 10/b show a portion of the POTS file in SDL/GR and SDL/PR, respectively. The

file shown in Figure 10/b is fed to the CREATOR2 system and the file in Figure 10/c is produced by CREATOR2 as a result of detailing. This latter is further converted to SDL/GR by the SDT CASE tool, as shown in Figure 10/d. It takes about 12 minutes to design simplified POTS composed of about 50 SDL/GR symbols.

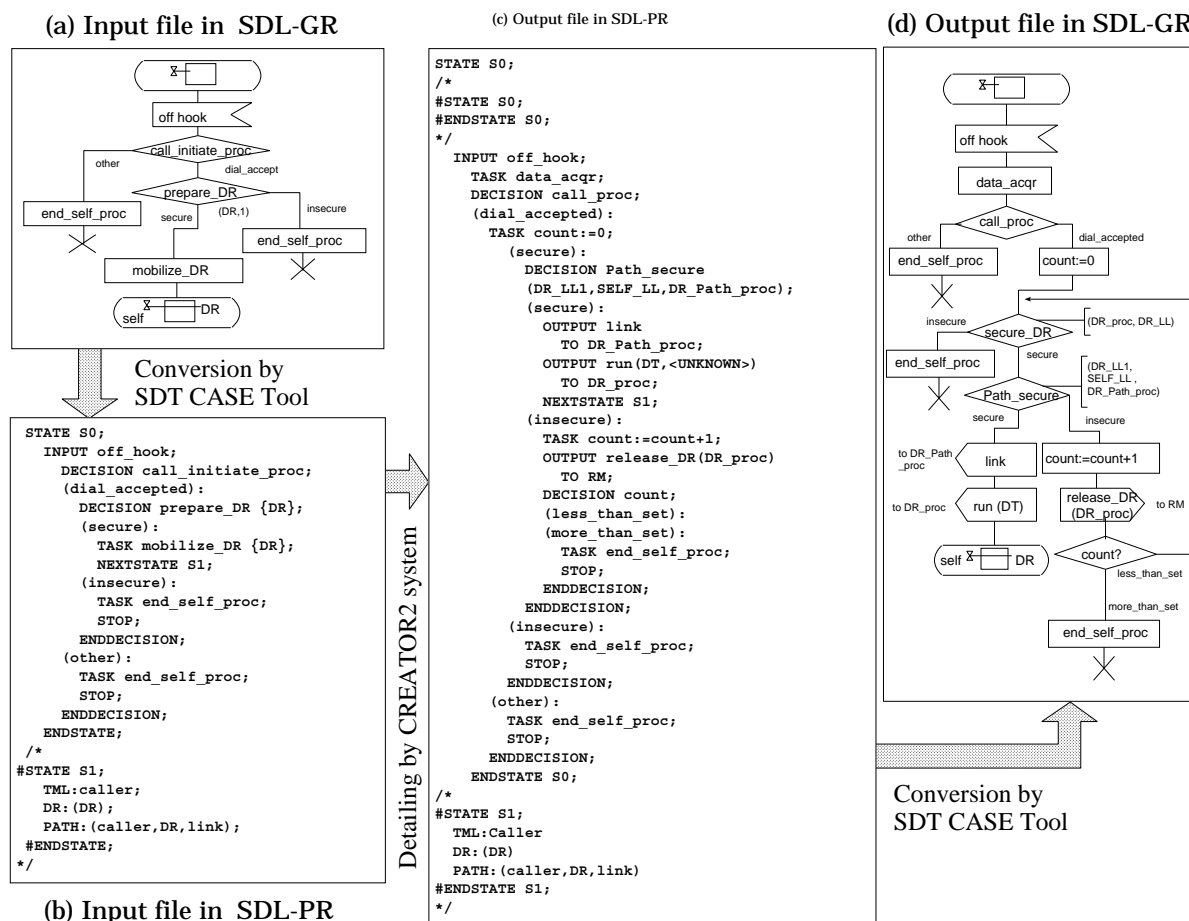


Figure 10. Detailing an input file by the CREATOR2 system.

Similar experiments are performed for other switching services such as Full-Call-Back-Transfer (FCBT) [17]. Figure 11 shows the overall progress of detailing for POTS and FCBT. During the design by the CREATOR2 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.

In CREATOR2 a set of frequently used SDL/GR symbols are accounted for. This is sufficient for most of the design cases and can be extended if required.

The most important limitation is that as the number of features increase the control program of the ADJUSTMENT expert becomes increasingly complicated. This also dominates the total run time of the system. We are presently investigating the guided search method and the LIBRARY expert to select the best option and minimize the run time.

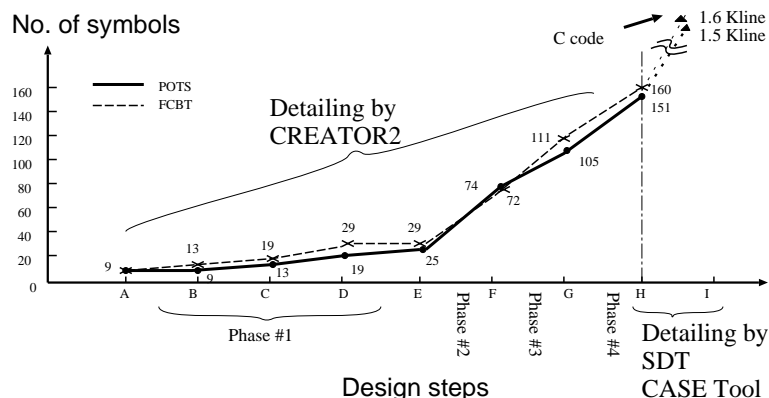


Figure 11. Detailing results for POTS and FCBT.

## 5. RELATED WORKS AND CONCLUSION

This paper presents an implementation of an SDL-based software design tool in the experimental expert system, CREATOR2 that follows the design steps of human designers by extracting and reusing the design process knowledge. Experiments on developing switching software is reported.

Research in automatic software design is inspired by top-down approach of the generic task-oriented methodologies, such as [16, 15, 18, 8], in which software design is viewed as incremental editing and refinement of a text-based generic object. In many systems the design can only proceed if the user is familiar with the given generic object and the available library. In this project we have applied graphical symbol based detailing rather than text-based one. This allows integration of the presently available CASE tools with the knowledge based reasoning techniques.

In some works the need for distinguishing between the design product knowledge and design process knowledge is mentioned [2, 20]. We have proposed a unified framework for representing both the design process and the design product knowledge using O-O techniques. 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.

The CREATOR2 system is still in the state of flux. It is currently a domain-specific program synthesis system. It serves as an experimental platform for the study of human design and lessons learned from its configuration and implementation can contribute to building a more sophisticated Knowledge Based Software Engineering (KBSE) system.

## REFERENCES

1. T. Baba, K. Miya, T. Yabuuchi, Y. Shigemori, Y. Naito and Z. Koono, "Software Creation: The First Results," in *Proc. IEICE Fall Conf.*, Tokyo, pp. 6.420-421.
2. S. Bhansali and H.P. Nii, "KASE: An Integrated Environment for Software Design," in *Artificial Intelligence in Design' 92*, J.S. Gero, ed., Kluwer Academic Publishers, 1992, pp. 371-389.
3. *CCITT Recommendation Z.100, Specification and Description Language (SDL)*, ITU,

- Geneva, 1992.
4. 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.
  5. ES/KERNEL/2-W Reference Manual, Hitachi, 1991.
  6. B.H. Far and M. Nakamichi, "Qualitative Supervisory Control Featuring Cognitive Capabilities of The Domain Practitioners," in *Proc. 2nd Makuhari Int. Conf. on High Technology, MICHT' 91*, Chiba, Japan, Feb. 1991, pp. 135-138.
  7. R. Jeffries, A. A. Turner, P.G. Polson and M.E. Atwood, "The Processes Involved in Designing Software," in *Cognitive Skills and Their acquisition*, J.R. Anderson, ed., Erlbaum, NJ, pp. 255-283.
  8. M. Klein, "DRCS: An Integrated System for Capture of Designs and Their Rationale," in *Artificial Intelligence in Design' 92*, J.S. Gero, ed., Kluwer Academic Publishers, 1992, pp. 393-412.
  9. Z. Koono, T. Baba and T. Yabuuchi, "Software Creation: A Trial for Switching software," in *Proceedings 5th JC-CNSS, 1992 Joint Conf. on Communications, Networks, Switching Systems and Satellite Communications*, Kyungju, Korea, 1992, pp. 261-265.
  10. Z. Koono, T. Baba, K. Miya, T. Yabuuchi, Y. Shigemori and Y. Naito, "Software Creation: A Systematic Approach," in *Proc. IEICE Fall Conf.*, Tokyo, pp. 6.420-421.
  11. Z. Koono, T. Baba, T. Yabuuchi, Y. Naito, Y. Shigemori and K. Miya, "Software Creation: Systematic Approach," *Technical Report of IEICE*, KBSE92-28, pp. 33-40.
  12. 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, 1993, pp. 327-331.
  13. 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.
  14. M.R. Lowry, "Software Engineering in the Twenty-First Century," in *Automating Software Design*, M.R. Lowry, et al., eds., AAAI Press, 1991, pp. 627-654.
  15. M.D. Lubars and M.T. Harandi, "Intelligent Support for Software Specification and Design," *IEEE Expert*, vol. 1, no. 4, pp. 33-41, Winter 1986.
  16. T.M. Mitchell, L.I. Steinberg and J.S. Shulman, "A Knowledge-Based Approach to Design," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-7, no. 5, pp. 502-510, September 1985.
  17. M. Ohmori, B.H. Far and Z. Koono, "Software Creation - Design Rules of Switching Service Program," (in Japanese), in *Proc. IEICE Spring Conf.*, 1993.
  18. C. Rich and R.C. Waters, "A Research Overview: The Programmer's Apprentice," *IEEE Computer*, pp. 11-25, November 1988.
  19. SDT CASE Tool ver. 2.2 Reference Manual, Telelogic, Sweden, 1992.
  20. J. Treur and P.J. Veerkamp, "Explicit Representation of Design Process Knowledge," in *Artificial Intelligence in Design' 92*, J.S. Gero, ed., Kluwer Academic Publishers, 1992, pp. 677-696.
  21. 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.