Software Creation: An Expert System for Applying Design Process Knowledge in Automatic Software Design

Behrouz H. FAR Takeshi TAKIZAWA Zenya KOONO

Department of Information and Computer Science, Saitama University
埼玉大学工学部情報工学科

A goal of this project is automating software design by accumulating knowledge and experience of human designers. CREATOR2, an experimental expert system that simulates behavior of human designers by applying design process knowledge is introduced. A novel point is using a unified representation scheme for the design process knowledge, composed of design rules for detailing and tacit knowledge, and the design product knowledge. Experiments on designing switching software are reported.

1 Introduction

Automating software design is the theme of the Software Creation Project [7, 8, 9, 10] and some preliminary results and implementations have already been reported [1, 7, 8, 9]. This paper introduces a viewpoints on capturing and applying the design process knowledge composed of design rules and tacit knowledge, using multiple strategy in applying tacit knowledge, and an Object-Oriented (OO) implementation of it in an experimental expert system. Figure 1 depicts the idea [11]. The main idea is to follow design steps of human designers. The design rules are extracted from an actual design by comparing the documents in successive design phases. This knowledge is reused by the CREATOR2 expert system. 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.

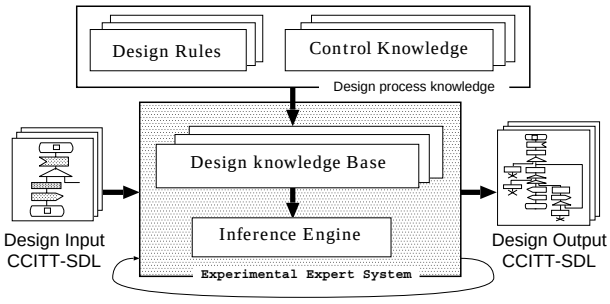

Figure 1: Overview of the Software Creation project.

2 Software Design Knowledge

Here we concentrate on representation of software design knowledge, i.e. *design product knowledge* and the *design process knowledge*, in reusable form, and introducing an structure for integrating them in CREATOR2 system.

2.1 Design product knowledge

Design product knowledge relies on the perspective that the design system is viewed. It includes domain-specific concepts and constraints of the task. We have used the Specification and Description Language (CCITT-SDL) [3] for encoding the design product knowledge and preparing the input/output files. SDL has both graphic (SDL/GR) and text-based (SDL/PR) representations. Fig. 2 shows an example of SDL/GR and SDL/PR. CASE tools for converting these two forms and editing such files are available [15].

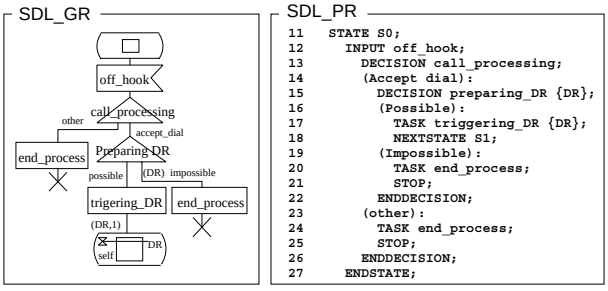

Figure 2: Example of SDL/GR and SDL/PR.

In SDL, the system is viewed as a collection of 'blocks' embodying 'processes'. A process is represented by an *Extended Finite State Machines* (EFSM) [3]. Processes communicate by discrete signals. An advantage of using SDL is its ability to concentrate on 'process' as the basic module of design and exchange messages between processes that corresponds with our OO view of the system.

連絡先： 〒338 埼玉県浦和市下大久保 255, 埼玉大学工学部情報工学科
Tel. 048-852-2111 Ext. 4552, Fax. 048-854-1984,
e-mail far@cit.ics.saitama-u.ac.jp

In CREATOR2, design product knowledge is represented by a structure of *class* and *instance* objects. The class objects are defined as counterparts of SDL/GR symbols, such as STATE, TASK and DECISION. Every 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. This data is recorded in the slots of the instance objects. Fig. 3 shows frame representation for the STATE symbol [5]. A design input file is represented by a structure of such frames in the CREATOR2, as shown in this figure.

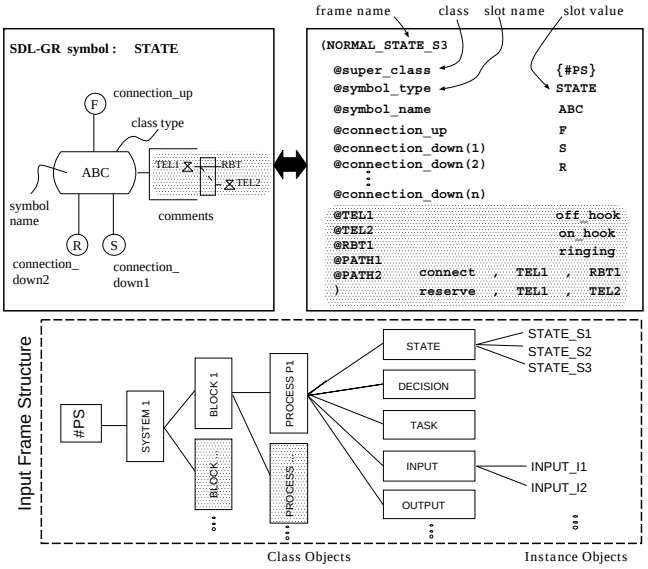

Figure 3: Frame representation of a SDL/GR symbol.

2.2 Design process knowledge

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 SDL/GR 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. A method for deriving design rules has already been introduced [1, 7, 8, 9, 13].

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

There are other design rules that embody certain procedures, and 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 of design rules using tacit knowledge. We have distinguished between two strategies for applying design rules, i.e. *intra-process detailing* and *inter-process detailing*.

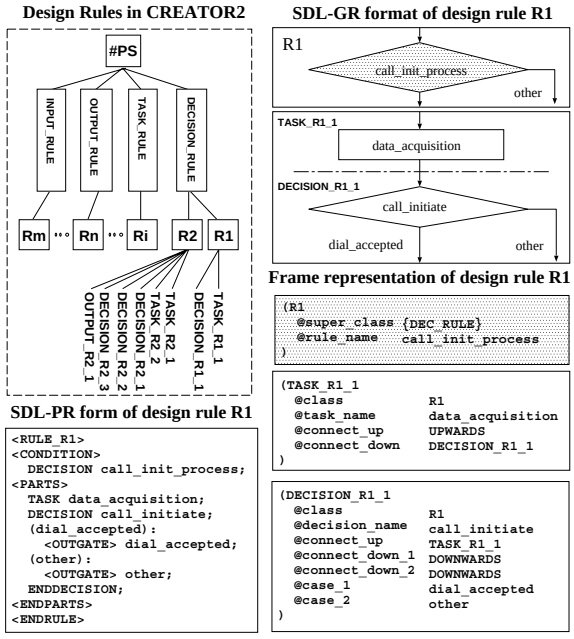

Figure 4: Frame representation of design rules.

In intra-process detailing, tacit knowledge is used for applying design rules within a given SDL process. This is shown in the lower part of Fig. 5. For instance, a frame representing a SDL/GR symbol is replaced by a collection of other frames that exhibit the same function in more detail in a SDL process.

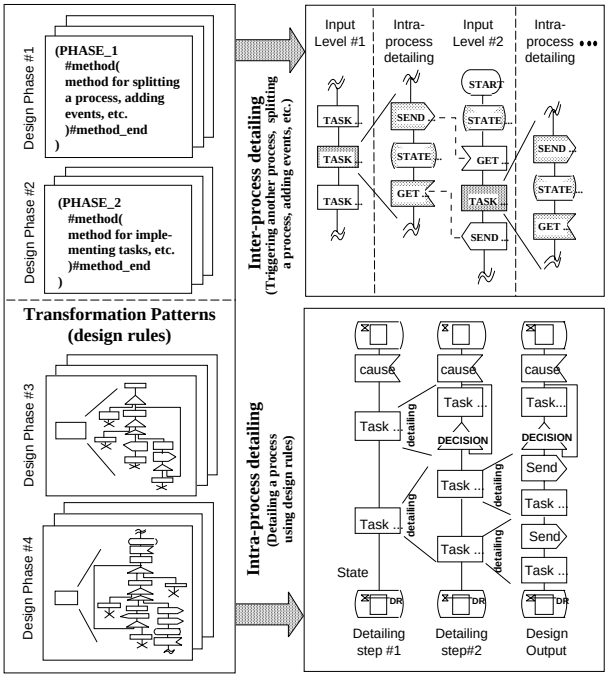

Figure 5: Intra- and inter-process detailing.

On the other hand, in inter-process detailing, tacit knowledge is used for applying design rules that trigger other SDL processes. For example, an instance of a new process is created. This is shown in the upper part of Fig. 5. 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 the events between them) or look-back (e.g., finding the preceding state, matching events, etc.) search strategy are applied.

3 Experimental Expert System

We have developed the CREATOR2 system based on the idea of integrating different expert units to achieve a common goal. The system is composed of 6 expert units that are specialized in a certain area (see Fig. 6).

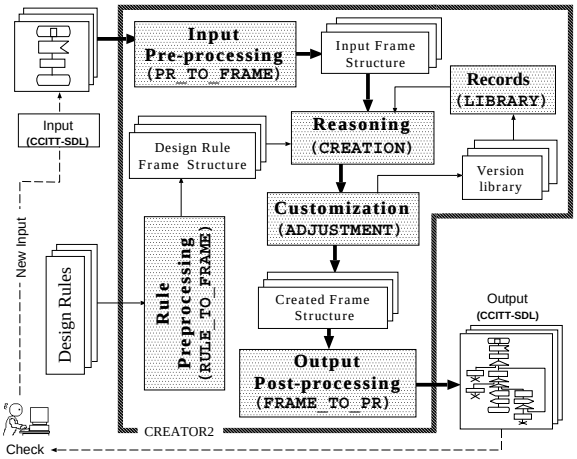

Figure 6: Prototype CREATOR2 system.

(1) The human designer prepares an initial design input file using SDL/GR graphic symbols. This is converted to SDL/PR by the SDT CASE tool [15], 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. Already used versions are recorded and applied in similar situations. 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.

(2) The CREATOR2 system is implemented on Hitachi 3050 Workstation using ES/KERNEL/2 expert system shell [4] that works together with the SDT CASE Tool [15]. SDT is used for graphic editing, converting text based SDL/PR to graph based SDL/GR, and converting SDL/PR or SDL/GR to C code. The other design tasks, conversion, reasoning and detailing are performed by the CREATOR2 system. Presently, 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.

(3) The CREATOR2 system is composed of 6 expert units (plus a dedicated `#ROOT` program) each performing a limited and specialized task. 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 system. (See Fig. 4). The extracted design rules are also converted to the frame structure using the `RULE_TO_FRAME` expert. (See Fig. 3).

The `CREATION` expert stays as the core of the CREATOR2 system. In the knowledge base, there are already two group of frames for input file and design rules. The `CREATION` expert is responsible for checking the input frame structure, fetching design rules and adding child frames of the matched design rules to the input frame structure. Fig. 7 shows an example of a method for intra-process detailing by the `CREATION` expert. Design proceeds by applying design rules at various levels during inter- and intra-process detailing. All design steps are recorded according to their order of appearance and the system can explain each step if asked to.

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 hardest task of automatic design because every slot of a candidate frame must be checked and all newly created frames should be accounted for. The `LIBRARY` expert keeps record of the customized design rules. 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 the CREATOR2 system.

Finally, the `FRAME_TO_PR` expert converts the final frame structure to text based SDL/PR that can be used by the SDT CASE tool.

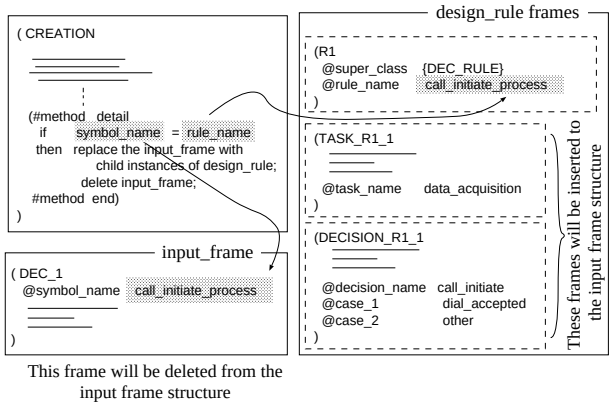

Figure 7: An example of detailing procedure.

4 Experimental Results

Switching software is considered as the problem domain. 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 splits it to two 'caller' and 'called' processes [13]. These two processes are further detailed to elementary tasks and decisions. It takes about 12 minutes to design simplified POTS composed of about 50 SDL/GR symbols.

Similar experiments are performed for other switching services such as Full-Call-Back Transfer (FCBT) [13]. Fig. 8 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 SDT CASE tool converts it to C codes of 10 times, resulting in 60-100 times code expansion.

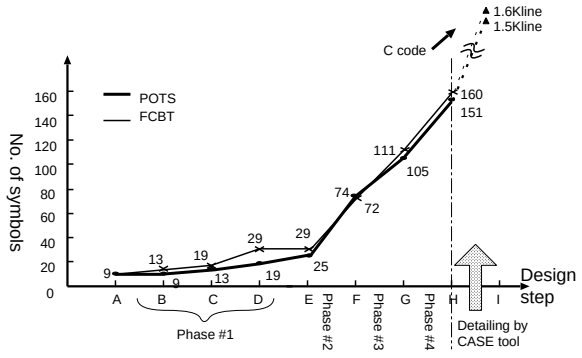

Figure 8: Detailing results.

5 Discussion

Research in automatic software design is inspired by top-down approach of the generic task-oriented methodologies, such as [12, 14, 6], in which software design is viewed as incremental editing and refinement of a text-based generic object. In this project we have applied graphical symbol based detailing rather than text-based one. This allows integration of the present 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, 16]. 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 software design knowledge. We use this for designing software other than switching software.

Many systems have considered a single strategy in design [14, 6]. In CREATOR2 various strategies are applied at different steps. We use both the top-down multi-layer decomposition strategy to proceed towards a goal and the data-driven strategy within a single layer.

6 Conclusion

This paper presents an implementation of the experimental expert system, CREATOR2 that follows the design steps of human software designers by extracting and reusing the design process knowledge. Experimental result of developing switching software is reported. The CREATOR2 is currently a domain-specific program synthesis system rather than a general purpose software design system. It serves as an experimental platform for the study of human design and building sophisticated knowledge based software engineering systems.

Acknowledgement

Authors thank the Information and Telecommunications Division, Hitachi, Ltd. and Information Systems Division, Hitachi, Ltd. for financial support of the switching administration area and technical assistance of this work, respectively, and the Telecommunications Advancement Foundation for financial support of the switching control program area.

REFERENCES

[1] T. Baba, et al., "Software Creation: The First Results," in *Proc. IEICE Fall Conf.*, Tokyo, Japan., September 1992, pp. 6.420-421.

[2] S. Bhansali, et al., "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] ES/KERNEL/2-W Reference Manual, Hitachi Ltd., 1991.

[5] B.H. Far, et al., "Software Creation: An SDL-Based Expert System for Automatic Software Design," in *Proc. 6th SDL-Forum*, Darmstadt, Germany, November 1993, to be published.

[6] 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.

[7] Z. Koono, et al., "Software Creation: A Systematic Approach," in *Proc. IEICE Fall Conf.*, Tokyo, Japan, 1992, pp. 6.420-421.

[8] Z. Koono, et al., "Software Creation: Systematic Approach," (in Japanese), *Technical Report of IEICE*, KBSE 92-28, 1992, pp. 33-40.

[9] Z. Koono, et al., "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, July 1992, pp. 261-265.

[10] Z. Koono, et al., "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.

[11] Z. Koono, et al., "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, USA, June 1993, pp. 332-336.

[12] T.M. Mitchell, et al., "A Knowledge-Based Approach to Design," *IEEE Trans. Pattern Anal. and Machine Intell.*, vol. PAMI-7, no. 5, pp. 502-510, September 1985.

[13] M. Ohmori, et al., "Software Creation - Design Rules of Switching Service Program," (in Japanese), in *Proc. IEICE Spring Conf.*, March 1993.

[14] C. Rich, et al., "A Research Overview: The Programmer's Apprentice," *IEEE Computer*, pp. 11-25, November 1988.

[15] SDT CASE Tool ver. 2.2 Reference Manual, Telelogic, Sweden, 1992.

[16] J. Treur, et al., "Explicit Representation of Design Process Knowledge," in *Artificial Intelligence in Design' 92*, J.S. Gero, ed., Kluwer Academic Publishers, 1992, pp. 677-696.