

PROCEEDINGS

**International Conference on
Communication Technology
(ICCT'94)**

**Volume
1
of 2**

**COMMUNICATION - - HIGHWAY OF ECONOMIC
GROWTH AND SOCIAL PROGRESS**

June 8 - 10, 1994 Shanghai, China

CO - SPONSORS

China Institute of Communications (CIC)

The Chinese Institute of Electronics (CIE)

China Academy of Posts and Telecommunications (CAPT)

Tsinghua University

Beijing University of Posts and Telecommunications (BUPT)

Shanghai Jiao Tong University (SJT)

TECHNICAL CO - SPONSOR

IEEE Communications Society (IEEE COMSOC.)

SUPPORTED BY

Ministry of Posts and Telecommunications (MPT)

National Natural Science Foundation of China (NSFC)

Shanghai Posts and Telecommunication Administration (SPT)

Tianjin Posts and Telecommunication Administration (TPT)

Shanghai Bell Telephone Equipment Manufacturing Co., Ltd. (Shanghai Bell)

ALCATEL China Ltd.

Northern Telecom, Canada

Siemens, Germany

AT&T of Shanghai, Ltd.

Shanghai Video & Audio Corporation Limited (SVA), China

China Electronic Equipments System Engineering Company (CESEC)

Beijing LAN XUN Communication Technical Center (LAN XUN)

K.C. Wong Education Foundation, Hong Kong

First Research Institute of Ministry of Posts and Telecommunications (FRI, MPT)

Tianjin NEC Electronics & Communications Industry Co., Ltd. (TJNEC)

China International Conference Center for Science and Technology (CICCST)

TECHNICAL PROGRAM OVERVIEW

ICCT'94

Communication--Highway of Economic Growth and Social Progress

June 8 (Wednesday)	8:40-12:30 am	Grand Ball Room Plenary Session											
June 8 (Wednesday)	2:00-5:30 pm	Session 11, VIP Room Network Management	Session 12, Dynasty I Switching Technology	Session 13, Dynasty II IN Service Development	Session 14, Dynasty III Mobile Communication	Session 15, Dynasty IV SDH/SONET	Session 16, Ball Room IV Optical Fiber Communication	Session 17, Rose Room Broadband Communication					
June 9 (Thursday)	8:30-12:30 am	Session 21, VIP Room Satellite Communication	Session 22, Dynasty I Image Coding and Compression	Session 23, Dynasty II ATM Architecture	Session 24, Dynasty III Speech Coding and Recognition	Session 25, Dynasty IV Wireless Communication	Session 26, Ball Room IV Communication Software	Session 27, Rose Room WDM and Fiber Amplifier					
June 9 (Thursday)	2:00-5:30 pm	Session 31, VIP Room Packet Switching and ATM	Session 32, Dynasty I LAN/MAN Architecture	Session 33, Dynasty II Broadband Network Survivability	Session 34, Dynasty III Multimedia Communication	Session 35, Dynasty IV IN Software and Protocol	Session 36, Ball Room IV Coding and Modulation	Session 37, Rose Room Circuit and System Design					
June 10 (Friday)	8:30-12:30 am	Session 41, VIP Room Image Communication	Session 42, Dynasty I Computer Network and Application	Session 43, Dynasty II Coding for Digital Communication	Session 44, Dynasty III Telecommunication and Project 863. of China	Session 45, Dynasty IV CDMA and Spread Spectrum Communication	Session 46, Ball Room IV Speech Coding and Compression	Session 47, Rose Room Modulation and Equalizer	Session 48, Jasmine Room Management and Human Resources				
June 10 (Friday)	2:00-5:30 pm	Session 51, VIP Room Data Communication	Session 52, Dynasty I Neural Network in Telecommunications	Session 53, Dynasty II Messaging Service and Systems	Session 54, Dynasty III WDM and Fiber Network	Session 55, Dynasty IV PCN and VSAT	Session 56, Ball Room IV Software and Protocol	Session 57, Rose Room Image and Speech Signal Processing					

Software Creation : Reuse of Design Knowledge of Switching Software

Chen Hui Behrouz H. Far Zenya Koono

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

ABSTRACT

This paper describes the outline, the approach and current status of the Software Creation project. The main idea is to design software automatically by simulating human designers, and reuse the design knowledge in a hierarchically organized work process. The presently implemented system consists of a SDL-based software design tool and an expert system. The performance of this system is improved gradually by learning from human design. Experiments on designing switching software are reported. Presently, this system converts the initial design input to 60-150 times source code.

1. INTRODUCTION

Switching software has been increasing at a rate of around 20% per year. As the result, the software size of most systems has reached to an order of millions of lines, and the typical annual increase has reached several hundreds kilo-lines of code. Studies in (Advanced) Intelligent Network has been started to enable service creation based on *reuse* of existing services. Presently, reuse depends on two factors: *standardization* of services and *customization* for creating new services.

Besides these two factors, in our *Software Creation project* we have considered *design knowledge extraction*, aiming at storing and reusing the background knowledge required to reproduce the design process in the same manner that human design experts do. The main idea is to follow design steps of human designers. Human design-

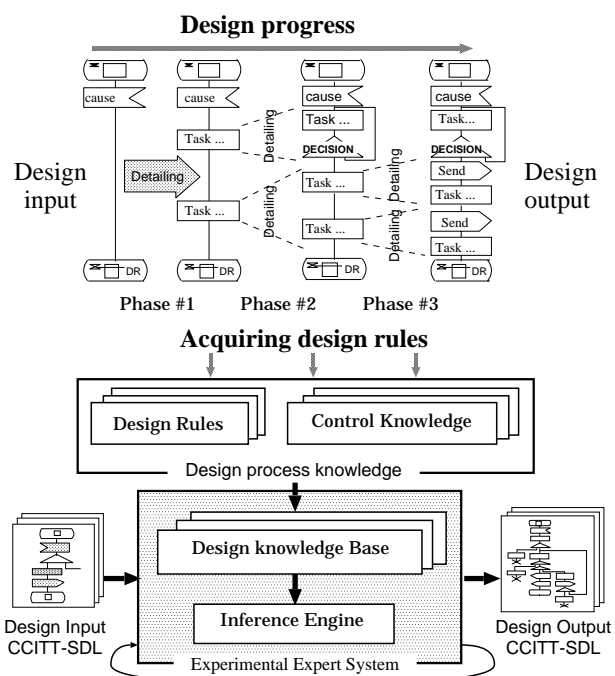


Figure 1: Structure of the automatic design system

ers' knowledge is extracted from an actual design and is reused by an expert system. The design rules are extracted by comparing the design documents in successive design phases. This enables one adding new features to the existing services easily and smoothly. This is considered as a new approach for creating software [1, 4, 5, 2]. We will introduce the key concepts and basic approaches of this project in the area of switching control and switching administration programs.

2. SOFTWARE CREATION SYSTEM

Figure 1 shows architecture of the Software Creation system [4]. The upper part shows the progress of an initial human design. Each flowchart indicates a set of documents produced at a certain design step. From adjacent two sets of documents, a conversion rule – from one flowchart symbol in the preceding document to several symbols in the following one – is extracted as a design rule. Thus gained design rules are accumulated in an expert system, named Creator, shown at the lower part of Figure 1.

In reuse of this knowledge, if the same or a similar input is given, the system reproduces the same design as long as design rules exist. If detailing by the previously extracted design rules does not work, new design rules particular to this new case must be extracted and added. By repeating this design many times the core of the design rules which is common to many design cases is formed and the performance of this system is improved gradually by adding new design rules. After the system has accumulated enough design rules, the system can design software with a high degree of automation.

The design documents at each step are produced by using a formal specification and description language, ITU (CCITT)-SDL [7]. In SDL a system is viewed as a collection of ‘blocks’ embodying concurrent ‘processes’. Processes are represented by an *Extended Finite State Machines (EFSM)* that communicate with discrete signals. SDL has both graphic (SDL/GR) and text-based (SDL/PR) representations. The switching control program is developed using SDL.

3. REUSE OF DESIGN KNOWLEDGE IN SWITCHING CONTROL PROGRAM

In the case of switching control program, the detailing goes smoothly, as all detailings may be derived from following the initial service specification.

The switching control program is expressed by a number of Finite State Machines (FSM) that communicate with each other using discrete signals and behave together as a large and complex FSM.

The detailing of a call service, POTS, goes as follows [6]: In the initial phase of design a ser-

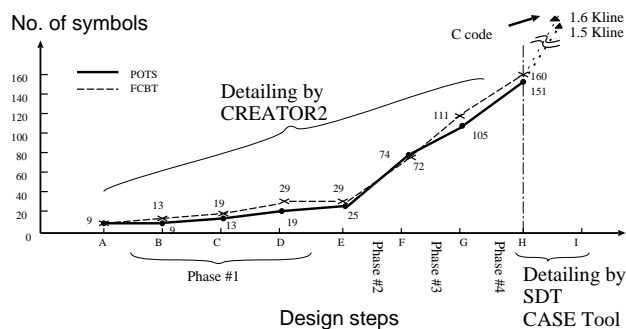


Figure 2: Progress of detailing of a switching control program

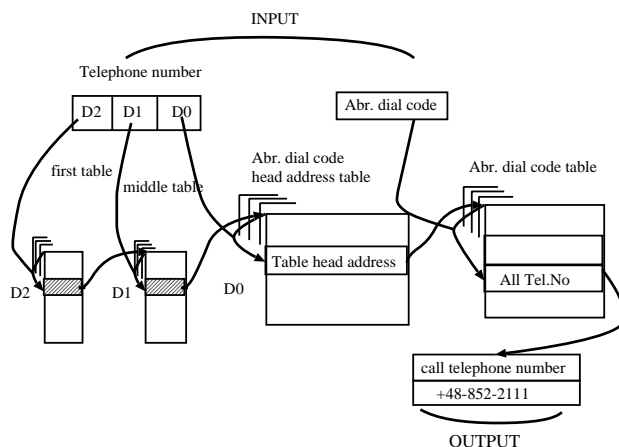


Figure 3: Abbreviated dial data structure

vice specification of a *call* FSM (‘process’ in SDL terms) is given. In the next step, the *call* process is split into *calling* and *called* processes, that communicate with each other using discrete signals. A difference between two adjacent processes is given by a design rule. Furthermore, a difference of thus defined two states in a process defines the necessary state transition, that is associated with a design rule. The state transition is repeatedly detailed by design rules to the most detailed level. All these steps are expressed by SDL, and SDL documents are converted to a frame structure inside the Creator Expert System, suitable for reasoning and knowledge processing. The final detailed design is transformed again to SDL, and a SDL CASE tool converts the detailed design to C code. Figure 2 shows the results of such detailing. Detailing factor by the expert system is around 10-15 and the conversion from SDL to C is

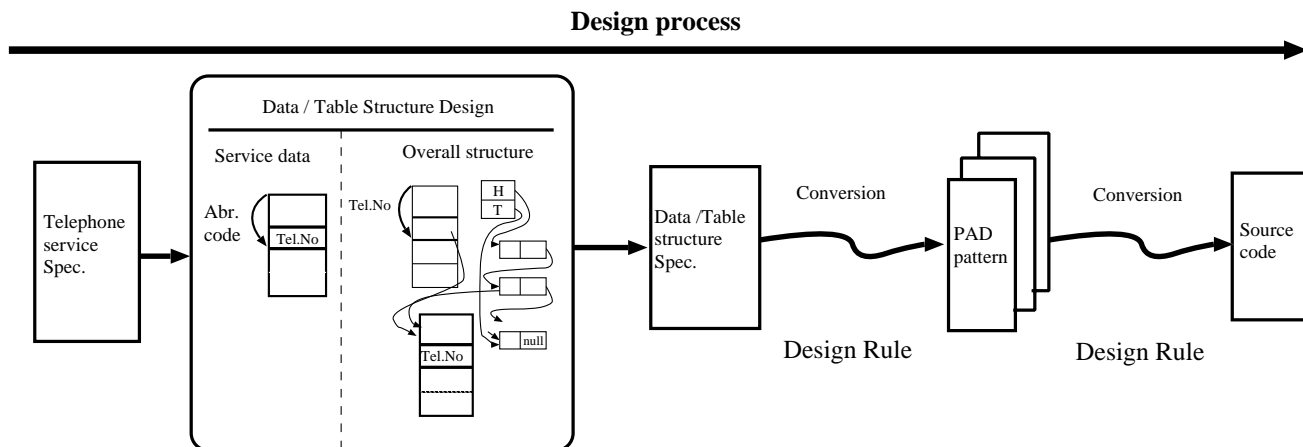


Figure 4: Design Process of switching administration program

around 6-10, thus giving 60-150 overall detailing.

4. REUSE OF DESIGN IN SWITCHING ADMINISTRATION PROGRAM

Switching administration program is for operation staff of a switching system. It enables or disables a customer to use a service feature by controlling various tables. Figure 3 shows data structure for an abbreviated dialing service. It consists of abbreviated dial lists in the right most corner and several stages of access tables to reach them. A subscriber's telephone number is associated with an abbreviated dial list, and the abbreviated code specified is converted to a full digit telephone number.

A central part of administration program enables a subscriber to use the service as follows: Upon the service order to allow a subscriber abbreviated dialing service, it tests a *busy/idle* bit of the subscriber, and makes it *busy* if it is *idle*. It also tests the *service feature availability* bit, and proceeds further if the outcome of check is *no*. It dequeues an idle abbreviated dialing list from the resource control queue, and connects it to the access table as shown in Figure 3. It then turns the *service feature availability* bit to *yes* and restores the *busy/idle* bit to *idle*.

Though administration program is rather non-sequential in nature, it is also expressed by SDL for keeping uniformity in the project. All tables are encapsulated in a table access process,

or FSM, having read/write/read-and-write functions. Central parts of administration program are programs performing respective logic operation as mentioned earlier and end with accessing table access processes. Various ways of reusing preceding design have been studied. The following introduce some approaches.

4.1. Old method

A similar approach to reusing design knowledge in switching control is followed and here design rules are extracted from successive design documents from human designed programs [3]. However, it is found that the detailing here is not as evident as the switching control program. This is mainly because of having too many parameters and factors that must be considered when reusing the design rules, and that they are included during design by imitation.

4.2. New method

In order to reuse design knowledge more effectively, another approach of reusing a higher level design is considered.

Figure 4 shows a progress of the design including its data/table design. First, service, data and their relevant control data/table are designed. Then control flow charts, specifying various data/table handling are derived from design rules. The detailed design is made from control flow charts. Design rules have been extracted from the relevant designed data/table structure

and process pattern of administration program. A number of different kinds of data structures are designed, and design rules are obtained. The difference of programs is adjusted by recursive use of design rules. In order to use design rules effectively, design rules have been abstracted and standardized.

4.3. Comparison and discussion

Figure 5 shows the results for the two methods. This figure depicts the accumulated number of design rules (vertical axis) vs. the accumulated number of administration services (horizontal axis). During the course of study, it was found that the number of design rules grows gradually and then saturates, showing a *learning effect*. This resembles a learning effect appearing in various human performance. At the flat area of the curve, full automatic design is possible for adding additional features.

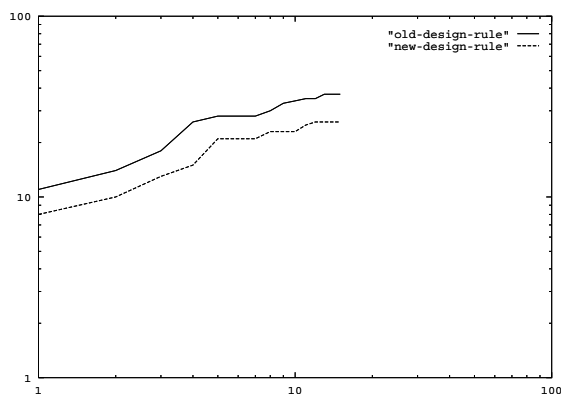


Figure 5: Learning effect of accumulated design rules

In the first method, it is found that designing one command requires about 0.53 design rules when 100 commands have been designed. This means that half of the design can potentially be done automatically.

Although design curves for the two approaches introduced above are quite similar, in the new method the degree of freedom and flexibility is higher than old one.

5. CONCLUSION

The outline, the approach and current status of the Software Creation project was introduced.

A novel view point on reuse of software design knowledge was given and an implementation of system composed of an expert system and a SDL CASE tool was reported. Experiments on extracting and accumulating design rules in two areas of study: switching control- and switching administrative- programs were given, along with discussion on system performance. It was argued that by accumulating design rules, extracted from human design, it will be possible to increase the automation degree of software design.

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, Japan. pp. 6.420-421, 1992.
- [2] 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.
- [3] 波多江 健一郎, B.H. Far, 河野 善弥, "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.
- [4] Zenya Koono, Behrouz H. Far, Takeshi Baba, Yasukiyo Yamasaki, Mari Ohmori and Ken-ichiro 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.
- [5] 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 *Proc. 5th Int. Conf. on Software Eng. and Knowledge Eng., SEKE' 93*, CA, 1993, pp. 332-336.
- [6] 大森 麻理, B.H. Far, 河野 善弥, "ソフトウェアクリエーション-交換接続プログラムの設計ルール," (Software Creation - Design Rules of Switching Service Program), (in Japanese), in *Proc. IEICE Spring Conf.*, 1993.
- [7] *CCITT Recommendation Z.100, Specification and Description Language (SDL)*, ITU, Geneva, 1992.