

Software Creation An Intelligent CASE Tool Experiment for Switching Software

Jilong Hua Hui Chen Mari Ohmori Behrouz H. Far Zenya Koono

Department of Information and Computer Sciences
Faculty of Engineering, Saitama University
255 Shimo-okubo, Urawa 338, Saitama, Japan
e-mail: ka@cit.ics.saitama-u.ac.jp

Abstract

This paper reports on an automatic software design system used for a switching control system. The design knowledge is gained from diagrams produced by a CASE tool during the initial human design, is stored in each corresponding expert system unit, and is reused for reproducing the design. The design procedure consists of splitting a call process, inserting tasks and detailing. Various considerations for implementing the system in an Intelligent CASE Tool are discussed.

1 Introduction

System software has an increasing trend of around 10% per year. As a result, the so called software maintenance is an important problem especially in telecommunications industry. This paper describes an answer to this question by automatic software design using an Intelligent CASE tool. Software Creation Project [1, 2] aims at automating software design by reusing design knowledge acquired during the initial software development. The design knowledge is extracted from design documents. During the following software maintenance period, when new features are added, the changes may be achieved by adding new design rules. The added rules together with the initial design rules are then reused by the Intelligent CASE Tool resulting in large saving of software maintenance cost. In this paper, principle of automatic software design is reported. The examples are taken from switching software. The case of switching administration program was published in the last conference [3].

2 Principle

Figure 1 shows the structure of the automatic design system. The upper part shows the initial

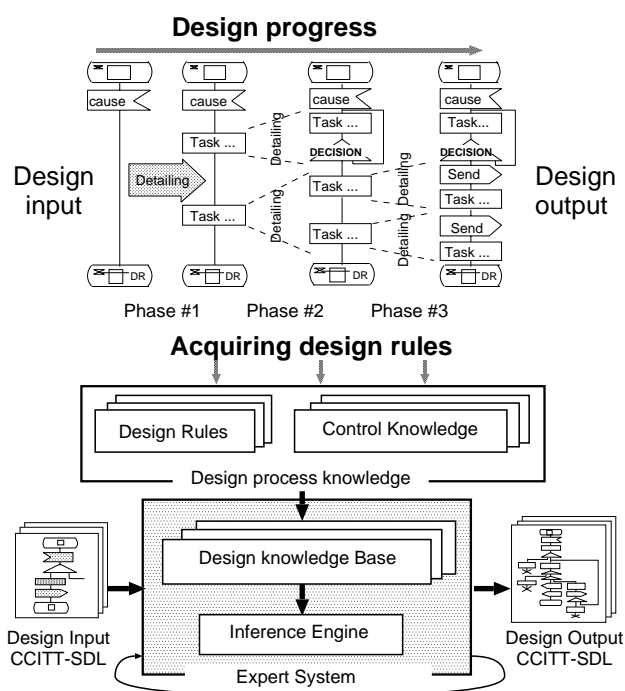


Figure 1: Automatic software design system

human design of a system. From adjacent two documents, a design rule (a transformation relation from an input to the output) is extracted and stored in an expert system. When the same input is applied to the expert system, the system produces the same output. As far as design knowledge exists, the system can find appropriate rules and design automatically. As the system grows, new design knowledge is added to the expert system, which is much more economical and quicker than to modify the software by engineers.

For representing the design information, ITU - Specification and Description Language (SDL) [4] is used. As an example for the switching system, a PBX switching control program comprising of many Extended Finite State Machines (processes) such as caller and called is used.

The detailing of a *plain ordinary telephone service*, POTS, goes as follows [2]: In the initial phase of design, a service specification of a call level state transition diagram is given. In the next step, phase #1, the *call* process is split into *caller* and *called* processes; then several decisions such as originating and terminating service control are added. Phase #2 inserts necessary tasks for state transitions, and phase #3 and phase #4 repeat simple detailing. Throughout all these steps, design information is expressed by SDL.

Figure 2 shows progress of detailing as design progresses. The final detailed design is fed to a SDL CASE, which can generate C or CHILL source code automatically. Detailing factor by the expert system is around 10-15 times and the conversion from SDL to C is around 6-10 times, thus giving 60-150 times overall detailing is achievable.

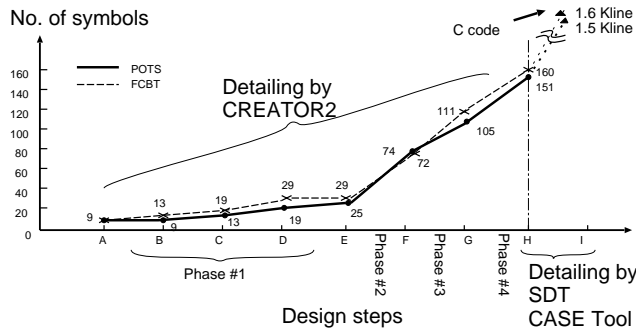


Figure 2: Progress of detailing during design

Design is a kind of transformation of design information, which details the input information hierarchically. This hierarchical detailing of design information is collected through design procedure using the CASE tool. Figure 3 shows the concept of the Intelligent CASE Tool system. The left side of this figure shows conventional human design working with a CASE tool. The design information is represented by a number of symbols (in this case SDL process diagram symbols). Then during each design step, a symbol of design

input is detailed to several symbols at the output level. The right side of Figure 3 shows several expert system units which store design knowledge. The expert system units detail the design information automatically instead of the human designers. They constitute an Intelligent CASE Tool.

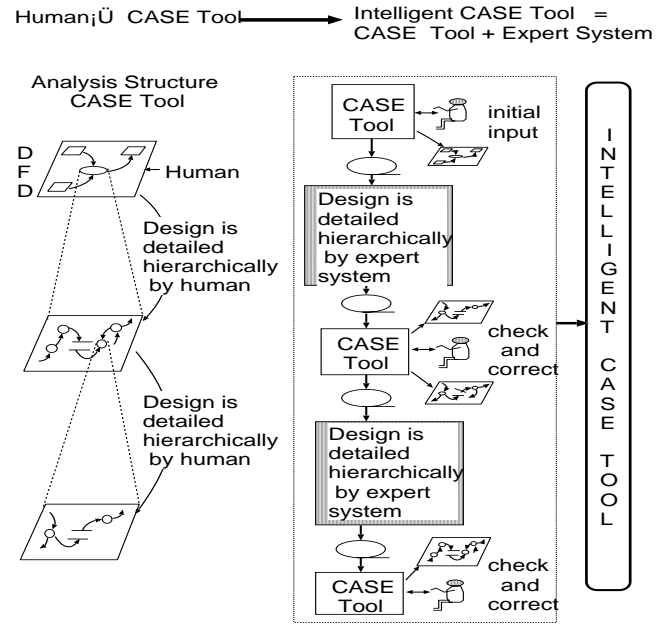


Figure 3: Intelligent CASE tool concept

3 Design process of switching control program

3.1 System architecture

Design process for a program to be designed usually consists of two kinds of detailing. In the initial phase of the design, usually there are some major dedicated design algorithms, whereas in the last part of the design, most design is simple type of hierarchical detailing. The entire expert system consists of several expert system units.

3.2 Design process

Figure 4 shows entire design process of a switching control program. The input service specification is given by a SDL process diagram (state transition diagram) for normal call procedure. It was found that it is better to add quasi-abnormal operation (e.g. abandoned call) prior to other designs.

Other remaining quasi-abnormal operations after this phase may be arranged to be automatically added as the design progresses [5].

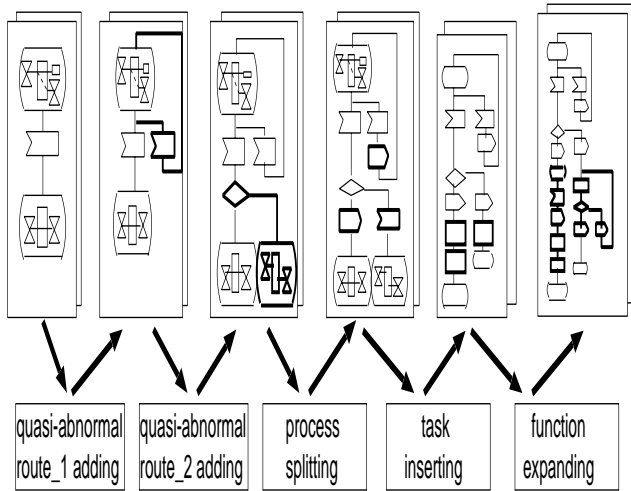


Figure 4: Design process of switching control program

3.3 Call splitting

In order to make the switching control program compact, a caller and a called are controlled by respective processes. As the design started from a call level state diagram as a specification, the next step is to split the call state diagram to those of caller side and the called side. In our system, the splitting is performed graphically as shown in Figure 5. A call level diagram is converted graphically as following:

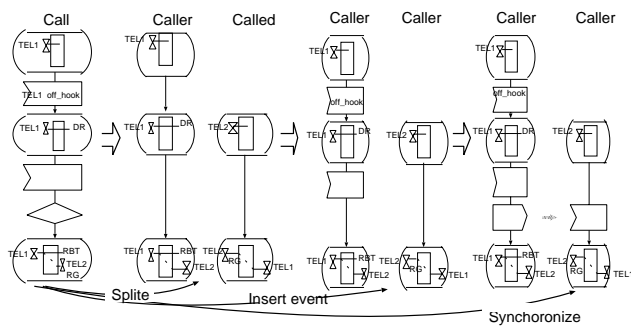


Figure 5: Algorithm of process splitting

1. Splitting: call state diagram as shown in each state symbol is split to a caller side state diagram and a called side state diagram.
2. Insert event: events triggering each state transition in call level diagram are copied to caller or called side SDL process diagrams considering their origin.
3. Synchronize: as several states are left without triggering events, they are added to synchronize all state transition.

3.4 Task insertion

Thus gained SDL process diagrams still lack tasks for performing state transitions. They are derived from differences of each resources in adjacent two states. For each expert system unit to identify a state diagram, call state diagram is added to each state symbol. State diagram must be computer readable.

3.5 Simple detailing

The last stage design process is simple hierarchical detailing type [1]. Figure 6 shows an example. The input side simple decision is detailed to the more complicated logic. Most detailing belongs to this type and a common expert system unit with Knowledge Base of design rules may be used.

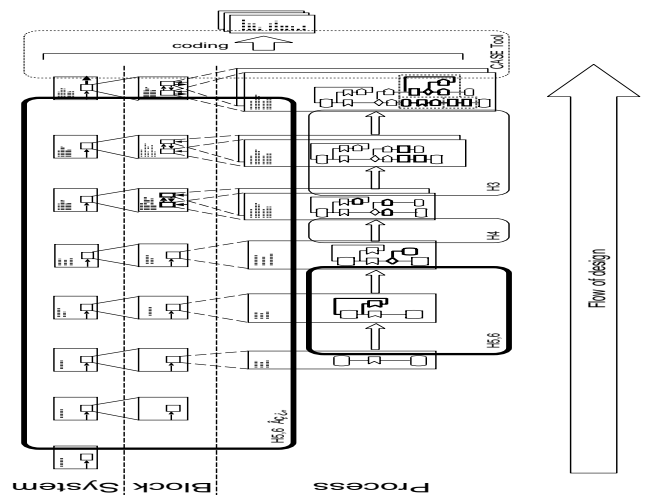


Figure 6: A simple hierarchical design rule

4 Expert system

4.1 System design

Important requirements for the expert system are its rapid development for saving total development cost and its architecture to be effective for any kind of expert system's processing. The former has been attained by a standardized systematic design procedure [6]; and for the latter, several points are reported below.

4.2 Unified knowledge representation

The entire expert system consists of many expert system units. Front end units are dedicated ones and the last stage unit is a universal type. Design information throughout the system has a standardized notation for SDL process diagram. Figure 7 shows the idea. SDL process diagram is like a flowchart. A symbol in design input is transformed to several symbols in design output. The right side figure shows the unified representation covering all symbols [6]. Thus standardized symbols are expressed by frames.

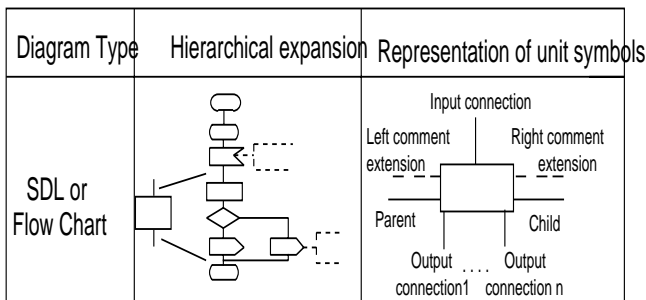


Figure 7: Frame expression of SDL simple

4.3 Call state diagram

In order to obtain the specification information during each step of design, a call state diagram is included in each state symbol of SDL process program. A diagram is connected to a state symbol by extension. It consists of graphic representations of both side terminals and a switch path hierarchically. Talking path connections between them are listed in a connection table at the top level frame, and state information for each resources are recorded in lower level frames.

4.4 Algorithm for inserting tasks

Using above unified representation and call state diagrams, all the expert system's processing may

be achieved. Figure 8 shows an example. Task for each resources are generated from differences between an initial state and subsequent state. An actual process diagram might have several subsequent states. Figure 8 shows task insertion to each link paying considerations for common tasks. The processing is performed by an algorithm called preorder traversal.

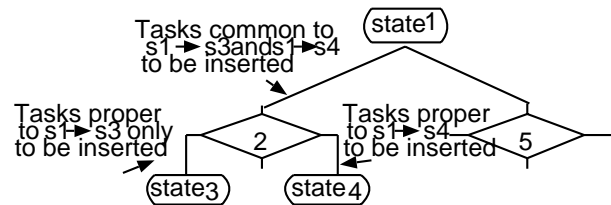


Figure 8: Preorder traversal method

5 Conclusion

The original concept of automatic software design featuring Intelligent CASE Tool has been introduced. On an example of a switching control software, algorithm as well as data structure for expert system units have been reported.

References

- [1] Z. Koono, T. Baba, and T. Yabuuchi, "Software Creation: A Trial for Switching Software", 5th Joint Conference on Communications, Networks, Switching System and Satellite Communications, 1992.
- [2] Z. Koono, B. H. Far, T. Baba, Y. Yamasaki, M. Ohmori and K. Hatae, "Software Creation: Towards Automatic Software Design by Simulating Human Designers", The Fifth International Conf. on Software Engineering and Knowledge Engineering, 1993.
- [3] Chen Hui, Behrouz H. Far, and Zenya Koono, "Software Creation: Reuse of Design of Switching Software", Int. Conference on Communication Technology, ICCT '94, 1994. pp. 63-66.
- [4] CCITT: "CCITT Recommendation Z. 100, Specification and Description Language (SDL)", ITU, 1992.
- [5] M. Ohmori, T. Baba, B. H. Far, and Z. Koono, "Software Creation: Design Knowledge for Design of A Switching Control Software", Technical Report of IEICE, KBSE 95-21, 1995.
- [6] H. Chen, K. Machida, B. H. Far, and Z. Koono, "Software Creation: A Systematic Construction Method of Expert Systems Used for Design", The Third World Congress on Expert Systems, 1996.