

SEKE'95

The 7th International Conference on Software Engineering and Knowledge Engineering

Sponsored by

**Knowledge Systems Institute (Founder and Organizer)
The Johns Hopkins University Applied Physics Laboratory
University of Pittsburgh**

In Cooperation with

**ACM (SIGSOFT)
IEEE Computer Society (TC on Software)**

Technical Program, June 22-24, 1995

Rockville, Maryland, USA

Software Creation: Using Specification and Description Language (SDL) for Capturing and Reusing Human Experts' Knowledge in Software Design

Behrouz H. FAR Hui CHEN Zenya KOONO
Department of Information and Computer Sciences
Saitama University
255 Shimo-okubo, Urawa 338, Saitama, Japan

ABSTRACT

Conventional knowledge engineering techniques for acquiring experts' knowledge can not produce quality knowledge due to improper knowledge documentation and informal knowledge acquisition method. We propose a method for knowledge acquisition based on documentation using Specification and Description Language (SDL). SDL is used to describe both the target system and the design process. The main idea is to follow deterministic problem solving behavior of human experts and document it. Then knowledge can be extracted by comparing documents of the successive steps. This knowledge is recorded and reused in similar or novel cases. We propose an implementation of this method in a distributed expert system for software design. The system is implemented on a number of platforms, each consists of an SDL CASE tool and an expert system for applying the design knowledge. The expert systems can communicate and share their knowledge and resources through the internet. This system serves as an experimental platform for the study of groupware design by simulating design of a team of human experts. We have found that through acquiring enough knowledge, this system can generate software in the same way that human designers do.

1 Introduction

In Software Creation Project we have been studying knowledge acquisition of a design process considering human designers. The main idea is to follow design steps of human experts [13, 14, 15]. The design knowledge is extracted from an actual design which is documented using Specification and Description Language (SDL) [3]. This knowledge is later reused by an expert system. Hierarchical nature of the design is assumed

as a priori (see Figure 1). Some implementations have already been reported [13, 14, 15, 8, 9].

The main theme of this paper is giving an introduction to the project from the knowledge documentation and acquisition point of view. Furthermore, in the previous reports a single platform implementation of the system has been introduced. Here we propose a multi-platform implementation of the system in a distributed expert system for groupware design using the World Wide Web (WWW) technology.

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 [14, 15]. Later, *control* knowledge is used for selecting and applying such rules. The research starts from the lowest and the most detailed step where the detailed documents are available and goes upward hierarchically to more knowledge intensive areas [16, 17].

Specification and Description Language (SDL) has been applied successfully in this project. We have found that SDL is useful when recording various steps of design, as shown in Figure 1. Formal and precise syntax and semantics of SDL is useful to extract correct pieces of information. Furthermore, SDL allows recording *comments* added to the main design routine. Those comments are pieces of data that should be included in the main course of design later on.

We have used this method to derive experts' design rules for software design. We have recorded design rules for several design cases and implemented those rules in the CREATOR expert system. Those rules are reused when designing similar software and/or modifying the original one.

The structure of this paper is as follows: Section 2 describes the knowledge components that are acquired. Section 3 introduces the distributed expert design assistant system and gives an overview of the

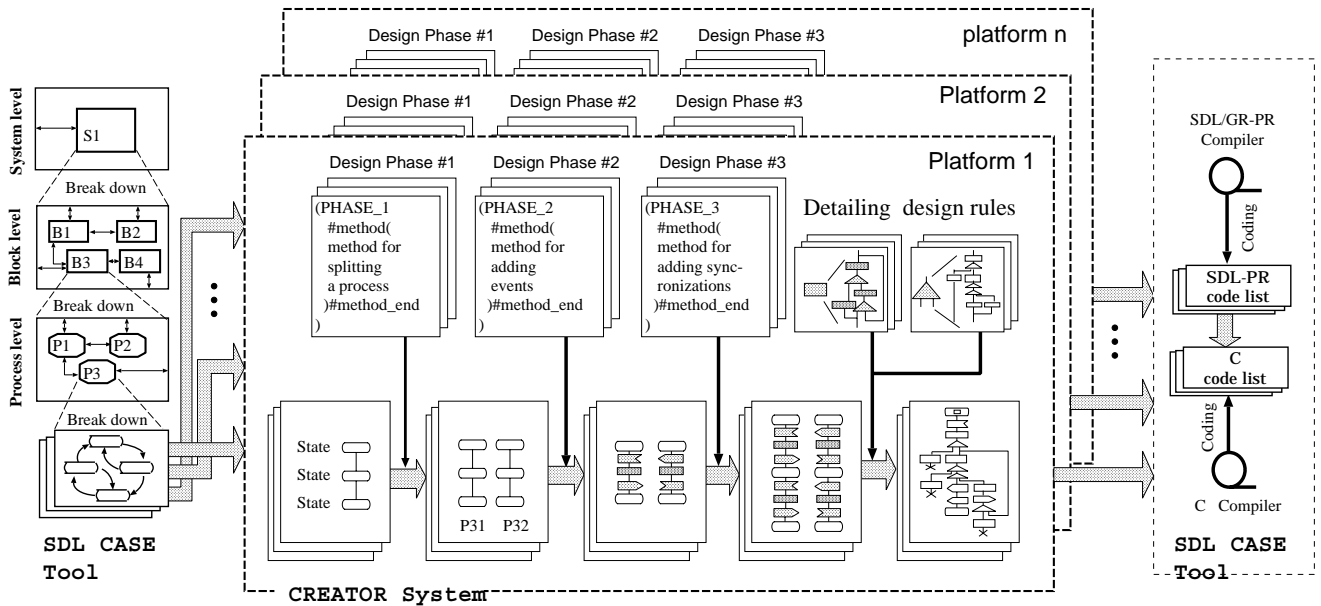


Figure 1: Overview of the knowledge acquisition procedure.

implemented system. We give a short discussion on the capabilities of the suggested method and the implemented system in Section 4 and give a conclusion in Section 5.

2 Components of Human Experts' Knowledge

There are two inter-related knowledge categories involved in human design: *design process knowledge* and *design product knowledge*. Here we concentrate on documentation and representation of both, in reusable form, particularly by introducing a structure for integrating these two.

During the intermediate design steps, human experts rely on their *design process knowledge* that is accumulated over years of solving similar problems. The process knowledge includes certain transformation patterns, or *rules* for converting an initial idea to a more detailed one, and *control knowledge* for operationalizing and applying such rules. The *design product knowledge*, on the other hand, consists of domain-specific concepts and constraints of the task.

2.1 Design product knowledge

The design product knowledge relies on the perspective that the target system is viewed. It consists of domain-specific concepts, constraints and models of the target system. We have used the Specification and Descrip-

tion Language (SDL) [3] for representing this knowledge.

In SDL a *system* is described in terms of *blocks* and *processes*. A process is described by a number of graphic simple symbols, such as TASK and DECISION and each symbol may have some data attributes. Furthermore, system STATES are defined and behavior of the system is described in terms of state transitions. Other *data* objects can be saved as *comments*. SDL has both human friendly graphical (SDL/GR) and machine friendly textual (SDL/PR) representation (see Figure 2). There are some CASE Tools, such as SDT [19], allowing editing, verifying and simulating behavior of the target system and translate between SDL/GR and PR.

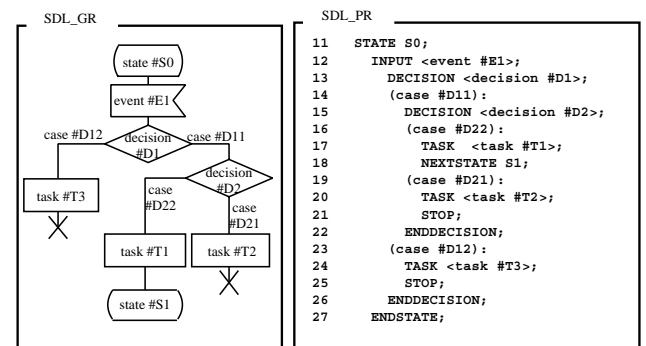


Figure 2: SDL/PR and SDL/GR

2.2 Design process knowledge

Design process is viewed as a progression towards a goal by applying already recorded design rules. The process knowledge involves *rules* acquired from experts, and *control knowledge* to make such patterns operational.

2.2.1 Rules:

Rules are used for replacing given graphical symbols with a number of other symbols in detailing the function, generating a task from and successive states, deciding upon next steps, and adding events, etc. Conventionally, in expert systems such rules are recorded in the form of production rules. Using SDL, we can visualize rules by a number of patterns, such as those shown in Figure 3. In this way we can categorize rules into groups and define templates for encoding rules.

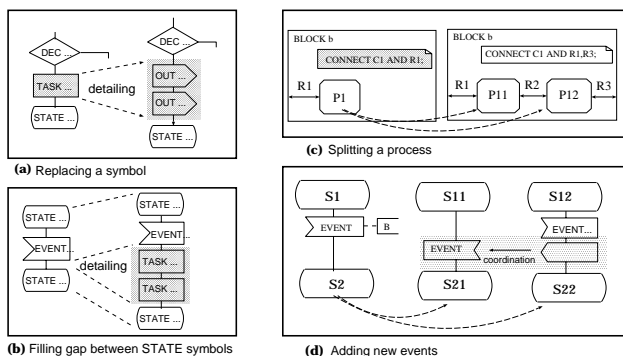


Figure 3: Several rule patterns.

SDL does not fully support data definitions. There are some other design rules that are derived by analyzing the data definitions. We use a kind of natural language with restricted syntax and semantics to reformat the data definitions. Sentences in the restricted syntax language are distinguished by their type, subject, condition(s), verb, etc. Such restricted language sentences are parsed and design rules are derived automatically.

2.2.2 Control knowledge:

A main part of human design involves selection and application of rules using control mechanisms. It is believed that control knowledge is neither declarative as in the rules, nor sequential as in the product knowledge. Here each step may be triggered based on a certain symptom. Some recent studies show that this knowledge may be applied in an opportunistic way [21]. The need for a scheme that can integrate the pure top-down approach with the data driven strategy has already been mentioned [5]. This is the meeting point of conventional expert system technology and case-based reasoning. Here we apply the symbol matching and

generate a sequence of symbols for state transition based on the recorded cases.

It should be noted that opposite to the rules, control knowledge does not depend on a particular case and can be implemented independently.

3 Application: A Distributed Expert System for Groupware Design

3.1 Overview

In *Software Creation* project automatic software design by following design steps of human designers is studied [13, 14, 15, 17]. A family of software CREATOR expert systems are developed and their basic features have already been published [13, 15, 8, 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. Knowledge-base in the CREATOR expert systems is accumulated gradually using the documentation and acquisition technique.

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 the 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 and maintaining its rationale.

The Software Creator System family were originally implemented on a single platform basis. However, actual design involves designers at remotely located sites to communicate and share their ideas. In this paper an implementation of a distributed expert system for groupware design, using the World Wide Web (Web) technology is introduced.

Figure 4 shows an overall structure of the system. This system is composed of a number of different platforms (workstation, PC, etc.) that run their own local expert system client. Each platform with a running expert system client is called an *expert unit*. Local units are connected by the Local Area Network (LAN) and

communicate using the NFS and HTTP¹ protocols.

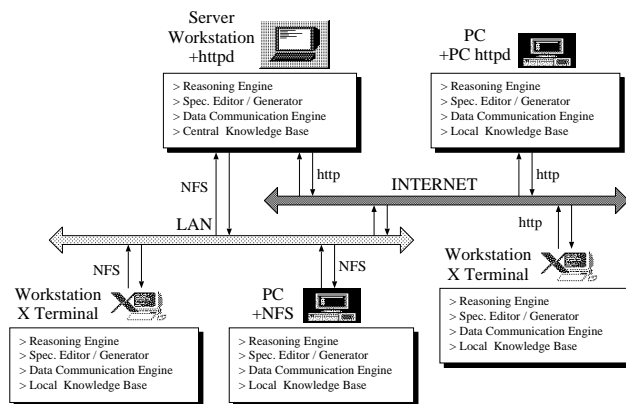


Figure 4: Distributed expert system architecture for groupware design

Remotely located units have access to the internet. This is possible in many different way such as using modem and SLIP (or PPP) accounts. Remote hosts communicate with other units through the HTTP protocol.

Similar to conventional systems, each expert system unit, no matter local or remote, has its own *local knowledge-base* and *reasoning engine*. Compared to the conventional expert systems, a main difference is that all expert units have an additional *communication engine*.

This architecture is general enough to implement any kind of groupware expert system. Expert system units of this structure and particularly the communication engine are introduced in detail in the following subsections.

3.2 Web system

The World Wide Web (Web) allows people at remotely located sites to communicate and share their ideas using a common communication protocol that can handle text files, images, sounds, forms, etc. [22].

A common use of the Web system is running a client application, using a browsing tool, by pointing to a local or proxy server to browse data written in the hypertext format that contains anchors that address other URLs (Universal Resource Locator) [20]. Using the Web technology, one can address a file by simply calling the protocol (http, ftp, gopher, etc.), host and path, respectively. Presently, a common use of Web system is running a client application, using a browsing tool, pointing at a local or proxy server [18] to browse data written in the hypertext format [11]. The hyper-

¹ HTTP is an internet protocol which runs over a TCP connection in order to transfer information by hypertext [1].

text files contain anchors that address other URLs, and connection to other servers is possible.

In this paper we use the Web system technology for sharing knowledge bases in teamwork development activities.

3.3 Expert system unit architecture

The system is implemented in the experimental groupware design system, by assisting a team of human designers through blending the knowledge acquisition, documentation and reasoning procedure, using local and remote knowledge bases.

All the expert system units of the groupware design system have a common internal architecture as shown in Fig. 5. This is an enhanced version of our standard architecture for design [9].

Each expert system unit is composed of:

- A data translator for converting specified input and output to frame data structure and vice-versa.
- A reasoning engine together with a simulator, customization and learning modules.
- Local knowledge base, including a domain-oriented library of design rules, data definitions and other design documents.
- A window-based user interface that allows a user interactively select the input, view simulation results, view and edit specification and documents.
- A design documentation module for recording the partial results and making hypertext design documents (optional).
- A data communication module, for launching http, ftp, etc. applications and communicating to the other units.

The data translation, reasoning and data communication modules are briefly described in the following subsections.

The server workstation is implemented on Hitachi 3050 Workstation using ES/KERNEL/2 expert system shell [7] 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. C is used to implement the basic communication methods. Perl is used to add additional features to the communication methods.

3.4 Data translation module

The human designer prepares an initial design input using graphic symbols by a SDL CASE tool. This is converted to SDL/PR and is fed to the system. The

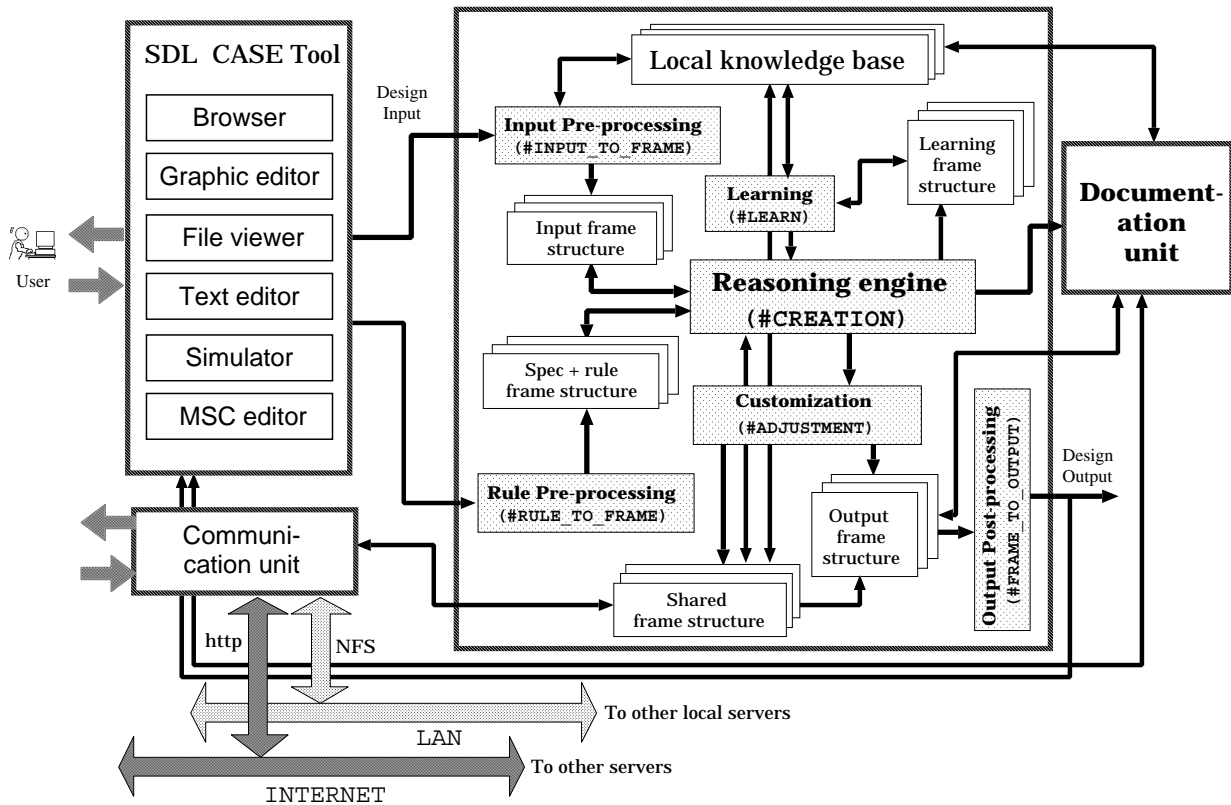


Figure 5: Expert system unit architecture

#INPUT_TO_FRAME module is used for parsing and converting the SDL/PR to the frame structure suitable for processing by the reasoning module. Each word of the SDL/PR file is read and interpreted and added to the created frames. Presently, a subset of the SDL symbols and some additional pictorial elements are supported.

There is also a set of design rules that are extracted from a similar case design. The design rules are converted to the frame structure using the RULE_TO_FRAME module. The RULE_TO_FRAME module receives as its input a design rule in the SDL/PR format and converts it to the frame structure.

Also, the results detailing and customization are recorded in the created frame structure which is finally converted to text based SDL/PR using the #FRAME_TO_OUTPUT module. This is fed to the SDL CASE tool and the designer can check and modify the results, interactively.

3.5 Reasoning module

This is the main reasoning engine of the experimental expert system. There are already two set of frames for the input file and design rules. The CREATION module is 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. All design steps are recorded according to their order of appearance and the system can explain each step if asked to.

Figure 6 shows an example of reasoning by the CREATION module. Here a frame of the input frame structure is fetched. First, the CREATION module decides to which system, block or process this particular frame is belonged. This is done through using a number of routines coded in the methods. Next, it is examined against the recorded rules. If a match is found, it may be considered for replacing this frame with the other ones indicated in the matched rule. Finally, the frame is copied into the output frame structure while adjusting its links to other frames.

The results of creation and reasoning are delivered to the #ADJUSTMENT module 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.

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.

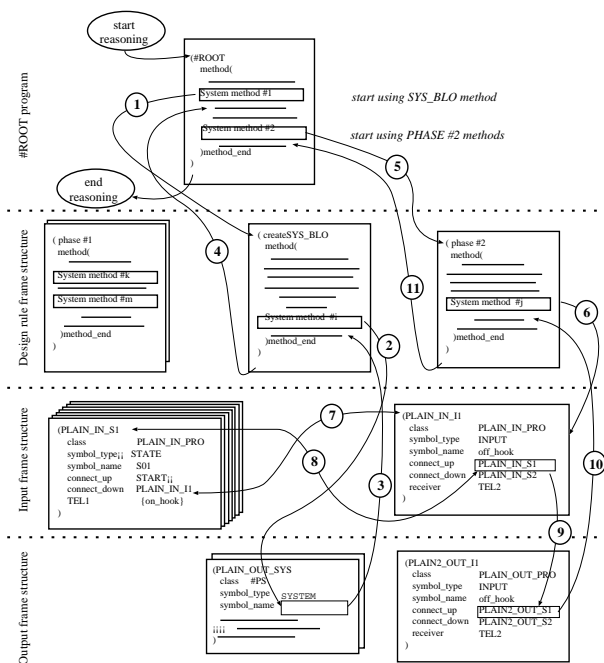


Figure 6: A detailing example by the reasoning engine

3.6 Local knowledge-base

The local knowledge-base is composed of a library of design rules and other design documents. A main advantage of this system is that documents and design rules related to a particular design task can be focused in a selected expert system unit and gradually this unit will grow up to have expertise in that particular area. All the other expert system units requiring such expertise acquire its service through http communication.

3.7 Data communication module

The data communication module is the heart of the system. It allows the reasoning engine to acquire appropriate knowledge and data from the other sites, if required. Figure 7 shows an example of the communication handled by the data communication engine.

All the expert units understand and communicate through http. They have a local index of the related information and their site names. In *step 1* a request for additional information from any unit, say unit (a), activates a POST method [11] and asks for the information from a unit that most likely has such information. Such a request is first acknowledged by recipient machine (b) that seems to have knowledge related to the current query. The local index of unit (a) is updated every time a new POST is acknowledged.

If this request fails, i.e., unit (b) does not acknowledge the request, as in *step 2*, the same query will be posted to the Server workstation (s) that maintains a

general index of the data items in the remote knowledge bases (*step 3*). In *step 4*, the most appropriate site for the requested information is selected. This is announced back to the unit (a) with the name of the unit that has such information, in this case (c). Again in *step 5* unit (a) launches a request to unit (c) using method POST. Unit (c) copies the requested information in a file (*step 6*) and gives the URL of the requested information to unit (a) (*step 7*). Then a GET method by unit (a) is launched to fetch the URL. This URL is a hypertext file that has some data entries and some other data, such as image of the graphical symbols, simulation results, etc. The data entries are parsed and added to the knowledge base of unit (a). In some occasional cases where the network traffic is high and the two sites allow remotely writable memory areas, a PUT method, say by site (c) may be launched to directly put the requested URL in the hand of unit (a).

Note that communication through the http requires all the transferable information be first written into a hypertext file and then transmitted. Therefore the data communication module is responsible for preparing the information and rewriting the information in the hypertext form before delivering it to the Web system.

In conventional application of the Web system running the GET, POST and PUT methods and data preparation tasks requires launching a browsing software². In this system we have implemented programs that can run the methods and reformat the hypertext data automatically without using a client software.

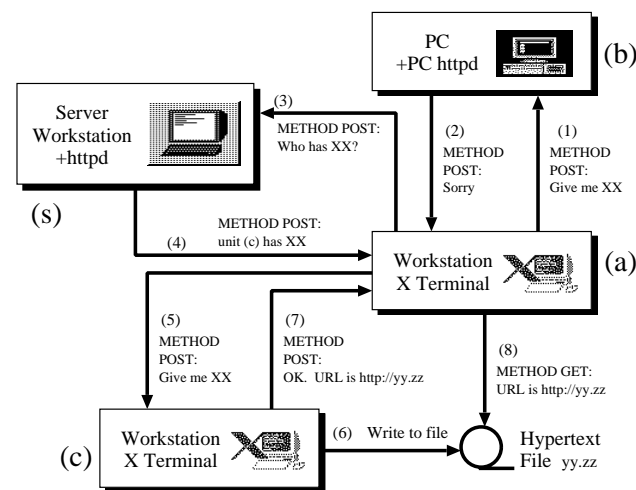


Figure 7: An example of data communication in distributed expert system for groupware design

Another advantage is that all the communication ac-

² Such as Mosaic, Netscape, Arena, etc.

tivities are done at the background and the user is not required to be aware of the http connections. Therefore, it is not necessary for the user to have networking and Web knowledge in order to use and interact with the system.

4 Discussion

4.1 Knowledge acquisition

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 are rich enough to detect inconsistencies and redundancies.

Another interesting point is the *learning effect*. We have found that knowledge acquisition through documentation shows learning effect, therefore after accumulating enough rules, the implemented system can perform the job almost automatically. For instance, Figure 8 shows the accumulated number of design rules vs. administration service commands in designing switching administrative program [4].

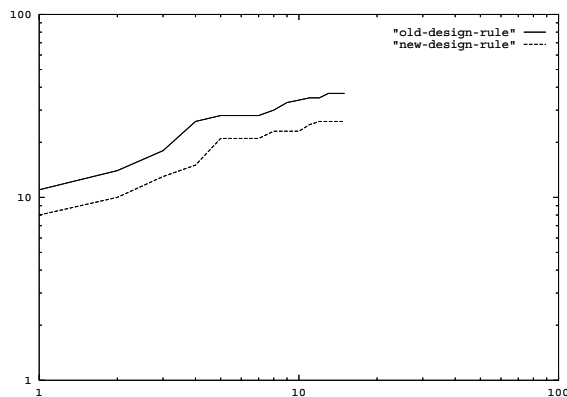


Figure 8: Learning effect of rules vs. service commands.

As is seen the accumulation curve shows rapid increase at first, then the rate decreases gradually though the increase continues. This resembles a learning curve appearing in various kinds 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 novel group of commands with new features are added. Using this curve, the extra resources required to solve a similar problem can be estimated.

4.2 Implementation considerations

The idea of *Groupware Systems* [6] has been around for a while, considering computer support of a team work.

There are already some examples of groupware applications such as bulletin boards, cooperative games, video conferencing system from and screen sharing cooperative design systems [12]. Most of these systems concentrate on techniques to integrate sound and video communication over a network of clients using the new features offered by broad-band digital network (B-ISDN).

Most of the work on distributed AI systems have concentrated on the reasoning and conflict resolution methods [2]. A common assumption in configuring the structure of such systems is that they consider clients connected through a kind of local area network (LAN). This limits the scope of application of such systems. To the best of our knowledge, there is no live project seeking integration of knowledge acquisition, distributed expert systems and knowledge bases assuming its clients are distributed over the internet. In our project various hardware (PC, Workstation, etc.) with various operating systems (DOS, WINDOWS, UNIX, etc.) and various methods of connection to the internet (LAN, ISDN, modem with PPP account, etc.) are considered and a mechanism for communication and cooperation is proposed.

On the other hand, the idea of implementing and using large scale and encyclopedia-like knowledge bases is running for more than two decades without much success. Even if such a large scale knowledge base could be implemented in some days, the cost of data search and retrieval would be quite high. We think that the only solution is using a Web-like distributed knowledge base equipped with efficient data communication. In the above sections we introduced one such architecture by blending the network and expert system technology.

4.3 Network considerations

Besides the experimental methodologies, there are three generally accepted and wide spread choices to implement a distributed architecture, i.e., Gopher, WAIS and Web. Among these, only Web is targeted and designed to fully functionalize a collaborative work space. *WAIS* allows text based data base search, using master and local indexes. However, it does not allow reference to other servers. *Gopher* allows text based search using menus, that can address other servers, in turn. The main method used is *GET* for retrieving a document. *Web* allows using hypertext besides the menus. In other words, it offers methods for *GET*, *PUT* and *POST*. Using these two latter methods is crucial to our project. A disadvantage of Web is that the above mentioned methods must be launched by using a browser client. In our project, in many cases that two knowledge based systems communicate and exchange information, launching a client software is unwanted and brings unnecessary overhead. We have developed some code to enhance the functionality of the Web methods without launching a client.

5 Conclusion

This paper presents an approach towards acquisition and implementation of human experts' knowledge in software design. The proposed approach efficiently encodes human knowledge. The problem solving steps of experts is documented and knowledge is extracted by comparing documents in successive phases. In this way, one can encode both the product and process knowledge within a unified framework and reuse them in novel cases. In the knowledge-base, we have distinguished between the rules and control knowledge. The former is domain oriented and derived from actual cases. The latter is general and can be used when dealing with other case. Performance curve of the system indicates extra resources required for solving similar or new problems.

Experiment on implementing a distributed expert system for groupware design is reported. This system resembles design by a team of human experts. The resources and knowledge bases are distributed and can be accessed through the internet.

ACKNOWLEDGMENT

The authors thank the Information Systems Division, Hitachi, Ltd., for supporting the expert system area, and the Telecommunications Advancement Foundation for supporting the switching control area. We are grateful to those who contributed to this project, specially former students of the CIT Lab of Department of Information and Computer Sciences, Saitama University, who developed parts of the experimental system.

References

- [1] T. Berners-Lee, R. Cailliau, H.F. Nielsen and a. Secret, "The World-Wide Web," *Communications of The ACM*, vol. 37, no.8, pp. 76-82 (1994).
- [2] A.H. Bond and L. Gasser, "An Analysis of Problems and Research in DAI," *Readings in distributed Artificial Intelligence*, A.H. Bond and L. Gasser (eds.), Morgan Kaufmann Pub. Inc., pp. 3-35 (1988).
- [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 *Proc. Int. Conf. on Comm. Technology (ICCT' 94)*, Shanghai, China, June 8-10, pp. 63-66, (1994).
- [5] 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, pp. 839-860, (1992).
- [6] C.A. Ellis, S.J. Gibbs and G.L. Rein, "Groupware: Some Issues and Experiments," *Communications of The ACM*, vol. 34, no. 1, pp. 38-58, (1991).
- [7] ES/KERNEL/2W-BS Reference Manual, Hitachi, (1991).
- [8] 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).
- [9] B.H. Far, T. Takizawa and Z. Koono, "Software Creation: An Expert System for Reproducing Human Cognitive Processes in Automatic Software Design," in *Proc. World Congress on Expert Systems' 94*, Estoril, Lisbon, Portugal, (1994).
- [10] B. H. Far, T. Tanaka and Z. Koono, "ソフトウェア設計における設計知識の体系的な構築法," (A Systematic Approach for Implementation of Human Design Knowledge in Automatic Software Design), (in Japanese), in *Proc. 1994 Annual Conference of JSAI*, Tokyo, Japan, pp. 439-442, (1994).
- [11] '<http://info.cern.ch/hypertext/WWW/Protocols/HTTP/HTTP2.html>'
- [12] H. Ishii, M. Kobayashi and K. Arita, "Interactive Design of Seamless Collaboration Media," *Communications of The ACM*, vol. 37, no.8, pp. 83-97, (1994).
- [13] Z. Koono, T. Baba and T. Yabuuchi, "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, pp. 261-265, (1992).
- [14] 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, pp. 327-331, (1993).
- [15] 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, pp. 332-336, (1993).
- [16] Z. Koono and B.H. Far, "A Systematic Approach for Acquisition of Human Design Knowledge," in *Proc. Japan-CIS Symposium on Knowledge-Based Software Engineering, JCKBSE' 94*, Pereslavel-Zaleski, Russia, pp. 243-249, (1994).
- [17] Z. Koono, B.H. Far, T. Sugimoto, M. Ohmori and Hui Chen, "A Systematic Approach for Design Knowledge Acquisition from Documents," in *Proc. 3rd Japanese Knowledge Acquisition for Knowledge-Based Systems, JKAW' 94*, Saitama, Japan, pp. 253-265, (1994).
- [18] '<http://info.cern.ch/hypertext/WWW/Proxies/Proxies.html>'
- [19] SDT CASE Tool ver. 2.2 Reference Manual, Telelogic, Sweden, (1992).
- [20] '<http://info.cern.ch/hypertext/WWW/Addressing/Addressing.html>'
- [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).
- [22] '<http://info.cern.ch/hypertext/WWW/TheProject.html>'