Requirement Analysis for Client Server System Configuration Design System Based On Case Base

Ayako Hiramatsu, Yoshitomo Ikkai, Hiroshi Morihisa, Takenao Ohkawa and Norihisa Komoda

Department of Information Systems Engineering
Faculty of Engineering
Osaka University
2--1, Yamadaoka, Suita
Osaka 565
JAPAN
e-mail: ayako@ise.eng.osaka-u.ac.jp

Abstract: Client server system(CSS) configuration design consists of the following three phases: requirement analysis, configuration plan design and configuration plan analysis.The necessary functions of a desired CSS configuration design support system can be formalized by analyzing each of the phases.We have proposed a CSS configuration design supportsystem (CIDAC : Case based Integrated Design Assist system for CSS), which is based on the above. The support system consists of the following four processes: requirement analysis, design planning, design analysis-explanation, and learning.

In requirement analysis, input ambiguous requirements are transformed into concrete requirements which are forwarded to the design generation function. One type of concrete requirement data is a set of function trees. This tree represents a business system structure and helps SE choose a framework for the target computer system.

This paper also presents a method that can generate function trees using several already generated function trees. In this method, suitable parts that are selected by the SE from other cases are automatically merged in order to complete the set of target function trees.

Ayako Hiramatsu received her BE and ME degrees from Osaka University in 1995 and 1996 respectively. She is currently a Ph.D student at the Department of Information Systems Engineering, Faculty of Engineering, Osaka University. Her research interest is information systems planning. She is a member of the Institute of Electrical Engineers in Japan.

Yoshitomo Ikkai received his BE and ME degrees in Information Systems Engineering from Osaka University in 1993 and 1995, respectively. He is currently an Assistant Professor at the Department of Information Systems Engineering, Faculty of Engineering, Osaka University. His research interests include scheduling, machine learning, and knowledge acquisition. He is a member of the IEEE, the Institute of Electrical Engineers in Japan and the Society of Instrument and Control Engineering.

Hiroshi Morihisa received his BE and ME degrees from Tokyo University in 1978 and 1980, respectively. Since 1980, he has been a systems engineer at the Nippon Steel Corporation and Nippon Steel Information & Communication Systems Inc.(ENICOM). Currently, he is also a Ph.D student at the Department of Information Systems Engineering, Faculty of Engineering, Osaka University. His research interest is business information systems planning. He is a member of the Institute of Electrical Engineers in Japan and Information Processing Society of Japan.

Takenao Ohkawa received his BE, ME, and Ph.D degrees from Osaka University in 1986, 1988, and 1992, respectively. He is currently an Associate Professor at the Department of Information Systems Engineering, Faculty of Engineering, Osaka University. His research interests include knowledge processing, qualitative reasoning, and computational molecular structure construction and prediction. He is a member of the IEEE, the Institute of Electrical Engineers in Japan, the Society of Instrument and Control Engineering, the Information Processing Society in Japan, the Institute of Electronics, Information, and Communication Engineers, and the Japanese Society for Artificial Intelligence.

Norihisa Komoda received his BE, ME, and Ph.D degrees from Osaka University in 1972, 1974, and 1982, respectively. He is currently Professor at the Department of Information Systems Engineering, Faculty of Engineering, Osaka University. From 1974 to 1991, he was with Hitachi, Ltd. as a researcher of Systems Development Laboratory. He stayed at the University of California, Los Angeles, as a visiting researcher from 1981 to 1982. His research interests include systems engineering and knowledge information processing. He is a member of the IEEE, the ACM, the Institute of Electrical Engineers in Japan, and the Society of Instrument and Control Engineering. He received the Awards for outstanding paper and the Awards for outstanding technology both from the Society of Instrument and Control Engineering.

1. Introduction

In the design of client server system(CSS) configuration, a system engineer(SE) could design with more empirical skill than with other system designs. It is difficult to accumulate information about client CSS requirements.

Moreover, a SE needs knowledge in order to cope with rapid technological change. Therefore, it is difficult for a novice SE to acquire CSS configuration design techniques, thus, a CSS configuration design support is desirable.

Several development methods and support tools for CSS design have been developed in order to assist a SE[3]. These methods and tools have affected the advancement of system development efficiency. However, they do not allow for the acquisition of empirical skills and knowledge about emerging technology, which are both regarded as important. In particular, with regard to assisting a novice SE, the support method can be used without empirical skill and know-how.

One of the most effective methods for the support of a novice SE is the automatic design of a CSS configuration. With regard to automatic CSS configuration design, we need understand CSS configuration design tasks. The CSS configuration design tasks consist of three phases. At each phase, there are many obstacles with regard to automatic CSS configuration design. We will clarify problems at each phase of the CSS configuration design in order to see what function is needed with respect to an automatic CSS configuration design.

We are developing CIDAC (Case based Integrated Design Assist system for CSS)[4] which designs a CSS configuration on a computer, so that a novice SE can design a CSS configuration as good as a skilled SE does.

In CIDAC, we have used three types of data as requirement definition, taking into account the behaviors in the next `design planning' process. One type of requirement definition is a set of function trees.

This paper also presents a method that can generate function trees in CIDAC, using a case base which stores already generated function trees without related information between each case[5]. Cases are merged and modified conversationally and fixed rules are not needed.

The next Section describes CSS configuration design tasks Section 3 explains a CIDAC configuration. Section 4 explains the algorithm that generates function trees, and Section 5 shows an example of function tree generation and an experiment.

2. Analysis of CSS Configuration Design Tasks

2.1 CSS Configuration Design Tasks

In CSS, the software and hardware are freely combined and flexible system construction can be achieved. An example of a CSS configuration is shown in Figure1.

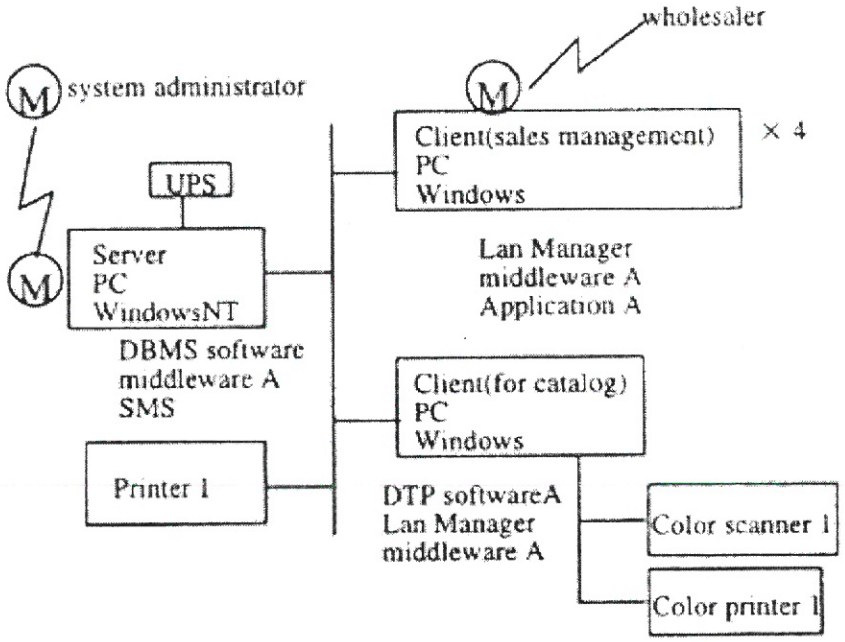

Figure 1. An Example of A CSS Configuration

CSS configuration design tasks, which are shown in Figure 2, consist of three phases: *"`requirement analysis'"*, *"configuration plan design"* and, *"configuration plan analysis"*. At the requirement analysis phase, a SE understands the current conditions of a clients system and clients' needs through discussion and

completes a requirement definition. At the configuration plan design phase, a SE constructs a hardware and software configuration from a requirement definition. At the configuration plan analysis phase, the SE analyzes whether the designed configuration is reasonable, and a proposed explanation to clients is generated. Based on this proposal, the client discusses the CSS configuration with the SE.

grasp of the requirements is indispensable in order to develop a system that clients want. Therefore, requirement analysis is important for system design [6,7].

The clients' first requirements are usually vague, and they include numerous definite purposes and functions. A SE usually interviews clients in order to obtain additional or precise information. Taking clients' present condition

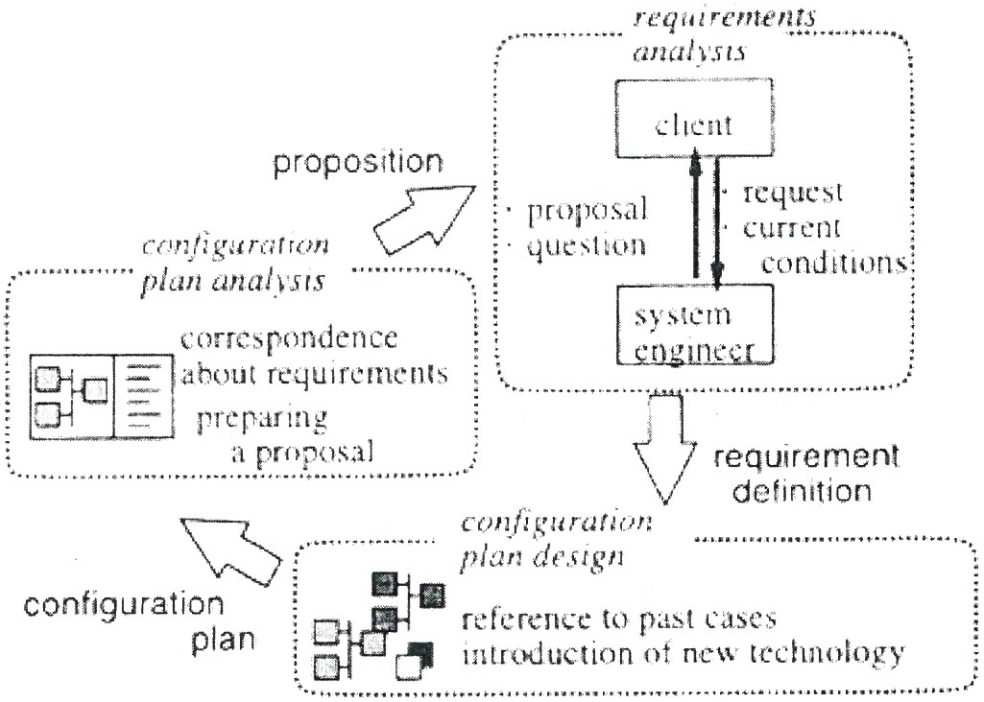

Figure 2. CSS Configuration Design Tasks

In practical CSS configuration design tasks, the requirement analysis phase and the configuration plan analysis phase are regarded as more important than the configuration plan design phase, because they require more empirical knowledge. Therefore, only the support of the configuration plan design phase is inadequate as regards the overall CSS configuration design. In other words, it is desirable to integrally support a series of CSS configuration design tasks.

As a first step in support system development, we will analyze a series of CSS configuration design tasks. Summaries of an analysis about these problems are given in the next Section.

2.2 Requirement Analysis

At the requirement analysis phase, system requirements are elicited from clients. A precise

into account, a SE asks suitable questions which are easy -to- answer by clients using experimental know-how. Through client interviews, a SE can analyze what kind of functions are necessary and can arrange the results into a requirement definition.

The problems at this phase are as follows:

- Information from clients is vague and defective
- Suitable questions are necessary in order to construct a precise requirement definition.
- Reasonable assumptions are necessary in order to complete a requirement definition from defective information.

2.3 Configuration Plan Design

A SE plans a CSS configuration from various combination of software and hardware by using the requirement definition that was generated in the previous phase. A SE seeks a previously designed plan which is similar to the target system, and modifies the parts that are not applicable to the present requirement definition. A SE also makes modifications in order to introduce new mainstream technologies. Since partial modifications by the requirement definition often cause contradictions in overall configuration plans, a SE flexibly deals with the requirement by taking these inconsistencies into account.

The problems at this phase are as follows.

- Because of individual research, a novice SE can only use or have a small past effective plan or amount of knowledge;
- It is difficult for a novice SE to cope with rapid technological change because of the difficulty of absorbing a large quantity of knowledge about new technology;
- When a satisfactory requirement is impossible, a SE needs to eliminate anything that clients do not like.

2.4 Configuration Plan Analysis

Through talking to clients about a designed plan, a SE gets new information and corrects the requirement definition. Given that clients have little knowledge about system configurations, a SE should write an explanation of the system configuration plan. In these reports, the following items are usually analyzed:

- How the plan satisfies the requirements in question.
- Which (or why) requirements cannot be satisfied.
- What assumptions can be added to the insufficient requirement information.
- To what degree the proposed plan is superior to other plans.

A SE should clarify two points: the correspondence between a required function and the parts of a designed configuration, and assumptions on or alternations to the requirement definition.

The problems at this phase are as follows:

- Because there is no one-to-one correspondence between a required function and the parts of a designed configuration, it is difficult to explain the correspondence.
- Requirements that have been assumed by a SE may not be obvious, because a SE often unconsciously assumes or alters clients' requirements.
- It is necessary for a novice SE to understand the effects of each part of the configuration plan.

3. CSS Configuration Design Support System : CIDAC

As mentioned in the above section, CSS configuration design tasks have various problems. By taking them into account, necessary functions for a support system can be created, as shown in Figure 3. We have proposed a CIDAC framework which is an integrated support system for CSS configuration design (see Figure 4). CIDAC is constructed by using the following four functions:

3.1 Conversational Requirement Analysis Function

The conversational requirement analysis part performs the following two roles. The clients' request is formed into the requirement definition that is needed in the next part (this can be understood by a computer); also it supplements insufficient requirements.

In order to acquire precise requirement information, a similar case, which was one of the past requirement analysis results generated by a skilled SE, complements insufficient information and generates questions. Even a novice SE who has little skill in information

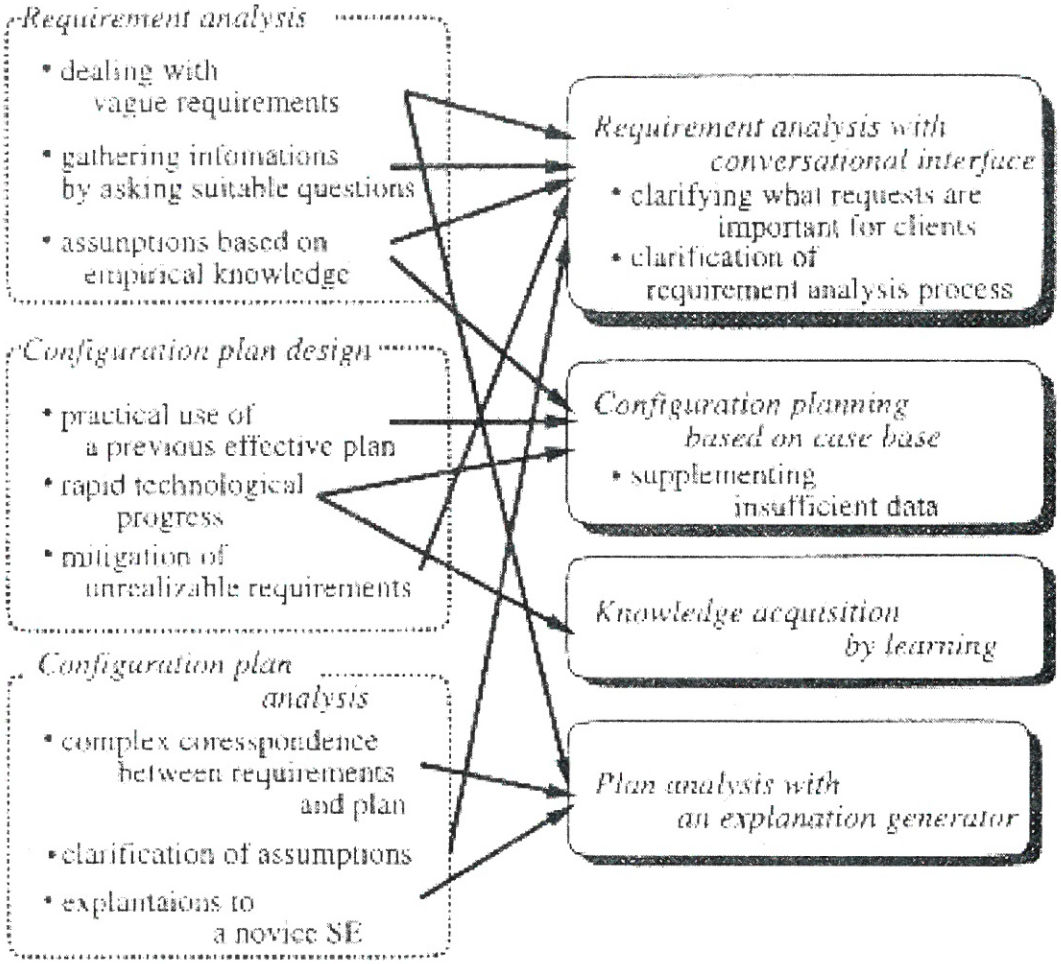

Figura 3. Necessary Functions for the Support System

acquisition, can gather information from clients. In order to explain the assumptions in the plan analysis-explanation function, this function provides requirement analysis process data which consist of information that is extracted from the cases referred to.

3.2 Case Based Configuration Planning Function

When a SE designs a CSS configuration plan, he/she will tend to refer to past cases and modify them according to clients' requirements. Therefore, CIDAC generates a configuration plan that uses a case base. Based on business system requirements and the CSS network requirements in the requirement analysis function, a case was retrieved and the framework for the target system was chosen. By using detailed requirements, the framework can be modified. Moreover, the usage of a case base can create complementary configuration components which cannot be chosen from the requirement definition.

To cope with rapid technological change, parts data and trend knowledge are used. Parts data are information about hardware or software products. Trend knowledge indicates the technology trends that show what products are in the mainstream. Moreover, for restricted components combinations by means of hardware-software compatibility, compatibility knowledge indicates the hardware and software sets which are effective in the same CSS configuration.

3.3 Plan Analysis-Explanation Function

The plan analysis-explanation part[8] acts as a mediator that tells a novice SE about data which are used in design or requirement analysis. Three types of explanations are generated in this part. They are given in the Section entitled "Analysis of CSS configuration design tasks":

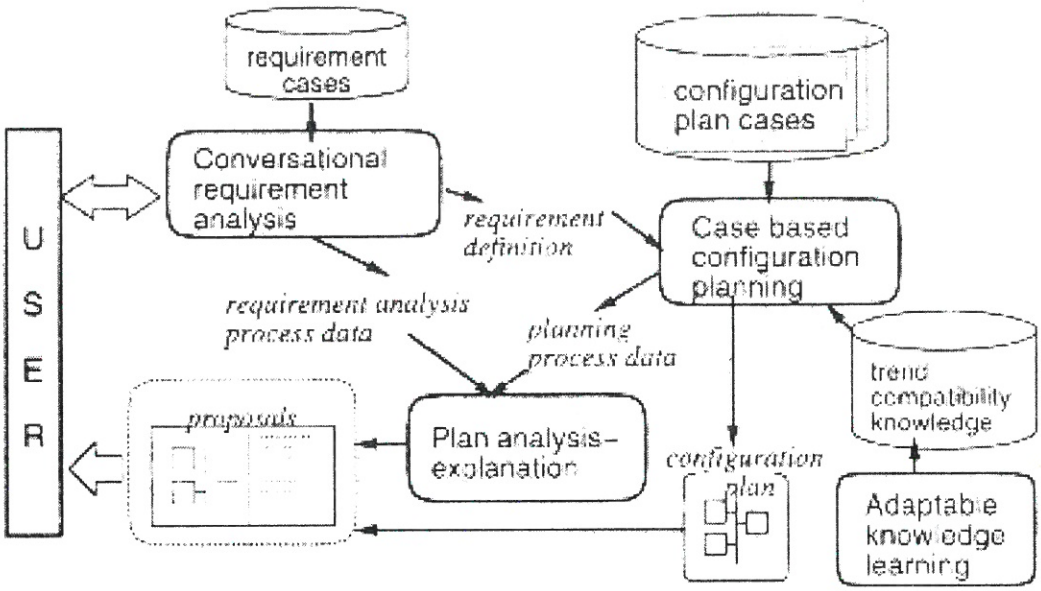

Figure 4. A CIDAC Configuration

1. Explanation for the correspondence of the configuration plan to clients' requirements.

2. Assumptions on the conditions under which the requirement analysis and the design process were performed.

3. A definition of the components of the CSS configuration.

The requirement analysis process data from the requirement analysis function and planning process data from the configuration planning function are merged in order to generate these explanations.

If all of the data have been explained, a novice SE may have a difficult time in understanding all the explanations and may overlook the important points. Therefore, CIDAC selects the important points that can explain the requirement analysis process and the planning process data. Explanations for specified points are generated in the form of natural language.

3.4 Adaptable Knowledge Learning Function

In this part, trend knowledge and compatibility knowledge used in the configuration planning function are extracted from the configuration plan cases.

4. Function Tree Generator

4.1 A Requirement Definition

Requirements from clients can be divided into three types; business systems, CSS networks and detailed requirements. These are shown in Figure 5. The business system requirement is related to clients' business processes. The CSS network requirement chooses the CSS basic configuration and is directly related to CSS components such as data traffic and security. The detailed requirement is an optional request which can be met by adding components.

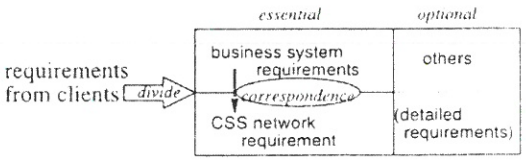

Figure 5. Division of Requirements

The business system requirement does not directly reflect the CSS configuration. Therefore, data that can be used for correspondence between business systems and CSS networks are necessary. In order to show these kinds of data, we provided a set of function trees which represented the arranged business structures systematically as business system requirements. These data are used not

only for the retrieval and modification of a CSS configuration plan case in the next function, but also for generating questions in order to gather requirement information, and to generate explanations in the plan analysis-explanation function. The CSS network requirement should not contain insufficient or vague information. We have provided these data as a table in which quantity or quality corresponds to each fixed item. The detailed requirements do not consist of fixed items. The detailed requirement in question is represented as a flexible data table in which the items are unspecified. As a result, we have taken three types of data as the requirement definition. Moreover, by a prioritization from clients in each requirement definition, the competition in the next function can be coordinated.

4.2 The Definition of A Function Tree

A function tree as shown in Figure 6 represents systemically arranged business structures as a business system requirement. These data are an important source of information which is a factor in choosing the framework for a target system.

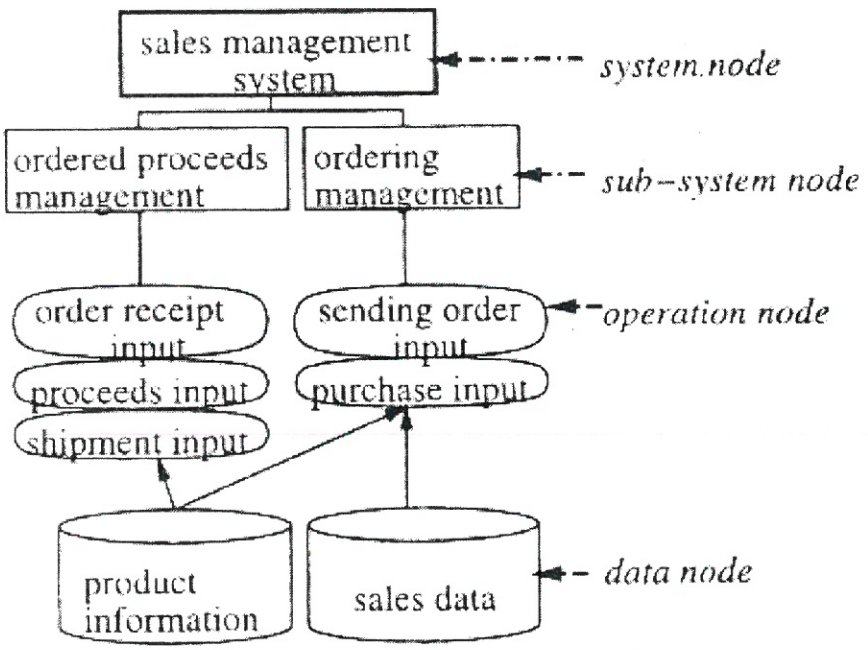

Figure 6. An Example of A Function Tree

A function tree consists of directed arcs which represent data flows, undirected arcs which represent materializing functions and the following four types of nodes:

System node: shows one organized role (which is performed by a server machine).

Sub-system node: shows what functions there are under the system node.

Operation node: shows what kinds of operation users perform in a function.

Data node: shows what data are being used in the system.

The function tree generator has the following four problems:

1. Since function trees represent a target business process as trees, a SE cannot generate function trees without understanding the business process. A novice SE who has little experience needs enrich his/her business process knowledge base in order to promote better understanding of various businesses.

2. The business processes which have the same functions do not always construct the same function trees. A SE needs understand these differences, which are dependent on business structures.

3. Normally, information about clients' businesses is acquired through talking to clients. While generating

function trees, a SE needs ask the clients about their business, and analyze the information received.

4. In order to more clearly understand clients' problems, a SE needs decide what kind of business processes can have their systematic structures expressed as function trees. This requires an empirical know-how.

The above problems make it difficult for a novice SE who has little empirical knowledge to create function trees. By considering these problems, CIDAC automatically generates a CSS configuration, with an easy-to-analyse interface even for a novice SE.

4.3 A Method for Generating Function Trees

Since function trees which belong to the same business category are similar, referring to past function trees in order to generate new target function trees is effective. Therefore, we will use a case base in order to generate function trees.

Case base reasoning is effective for a target problem in which it is difficult to establish solving rules. It can solve a problem by modifying a past similar successful (or unsuccessful) case. In a great deal of the research that used case base reasoning [9][10][11], one of the important factors is a method that can modify a case. In particular, when there are not enough rules or data to modify a case, the collaborative case base reasoning complements these rules or data by asking users about case modifications.

In order to cope with this problem, we need retrieve not only the most similar case, but all similar cases. These are used to generate function trees and be conversationally modified. Conversational modifications with all similar cases can prevent a shortage of information. All cases which have the same business functions as a target are retrieved with keywords, and the information that can modify these cases is obtained conversationally. Modification results can therefore be connected as a set of target function trees.

In conversational modification, in order to help users easily input modified information, the input form is the alternative, whether a node in a case is suited or unsuited. Against the alternative, if users do not know a target business, they can ask their clients so that information about modifications should be obtained.

4.4 An Overview of the Generator

The proposed function tree generator asks users' opinion about certain cases, creates applicable

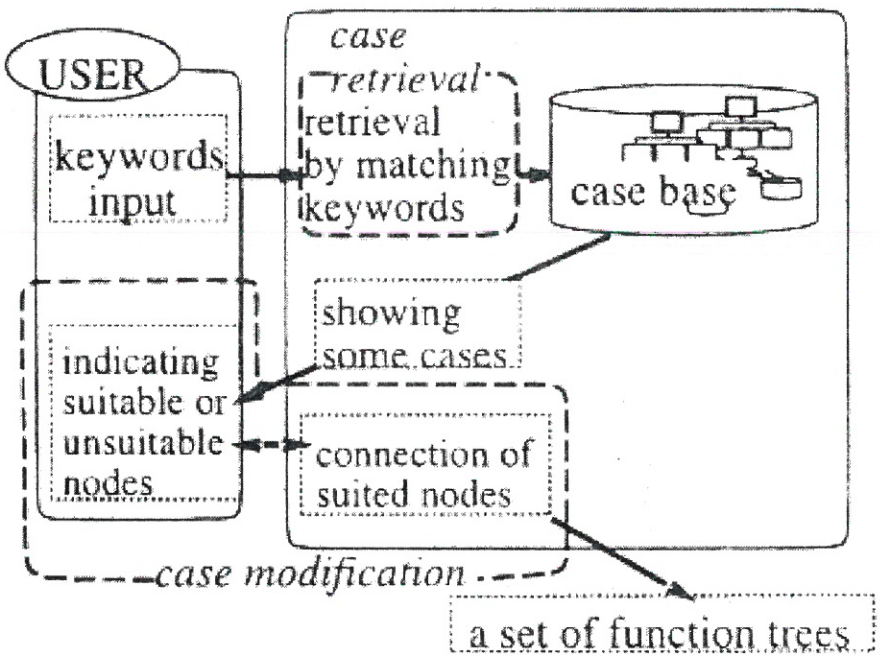

Figure 7. An Overview of the Function Tree Generator

fragmented cases and connects them. An overview of this generator is shown in Figure 7.

When fragmented cases are connected, the words in the nodes are matched. As the same words which were included in the function trees of different business processes have different meanings in this generator, a foundation case which has the most suitable parts is matched to all other cases from higher rank nodes. Using this method, hierarchical relationships between nodes can be envisaged.

A case is defined as a pair of a set of function trees which were generated in the past in addition to keywords about business processes. These past cases are stored in a case base.

21 cases are stored in the current case base. For example, when we use the keyword 'sales department', five cases can be retrieved by using this case base.

4.5 Connections Between Cases

Based on clients' requirement, users specify suitable and unsuitable nodes for each retrieved case. The former are necessary for a target system. The latter are nodes which have hierarchical relationships or means which are not necessary for a target system. By using this information, cases can be connected and a set of target function trees can be formed.

First, cases in which suitable or unsuitable nodes are specified are transformed into a foundation case and part cases. A foundation case and a part case can be defined as follows:

A foundation case : a retrieved case which has the most suitable nodes and which the unsuitable nodes have been removed from.

A part case: a retrieved case in which only the suitable nodes remain.

Each part case is connected to a foundation case. If the contents of the nodes in the part cases is the same as the contents of the nodes in a foundation case, they overlap. In this connection, since the lower ranking node is influenced by the higher ranking one, the nodes in the part cases are arranged from higher to lower rank. An example is given in Figure 8.

When a node in a part case has none of the same nodes or the same plural nodes as a foundation case does, the generator attempts to ask users for more information.

Therefore, potential cases are shown to users as examples of connections. These cases show all of the situations in which unconnected part cases can be included in the result trees. Users indicate the suitable or unsuitable nodes in the potential cases. These cases can be connected again by using this additional information. Finally, if necessary business functions or directed arcs which show data flow are insufficient, users can modify a set of completed function trees.

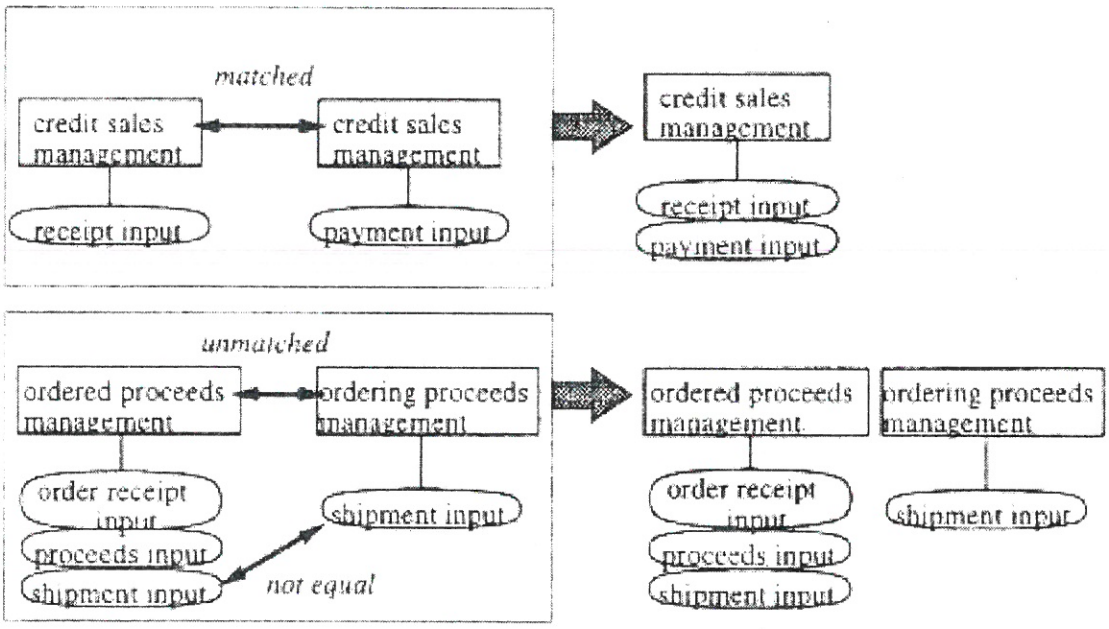

Figure 8. A Comparison of Nodes

5. An Example of Function Tree Generation

5.1 An Example of Case Connections

We will explain this connection method by means of an example. After the suitable or unsuitable nodes have been indicated, the results are formed into a foundation case and part cases as shown in Figure 9. The connection of the foundation case and the part cases is described in the following:

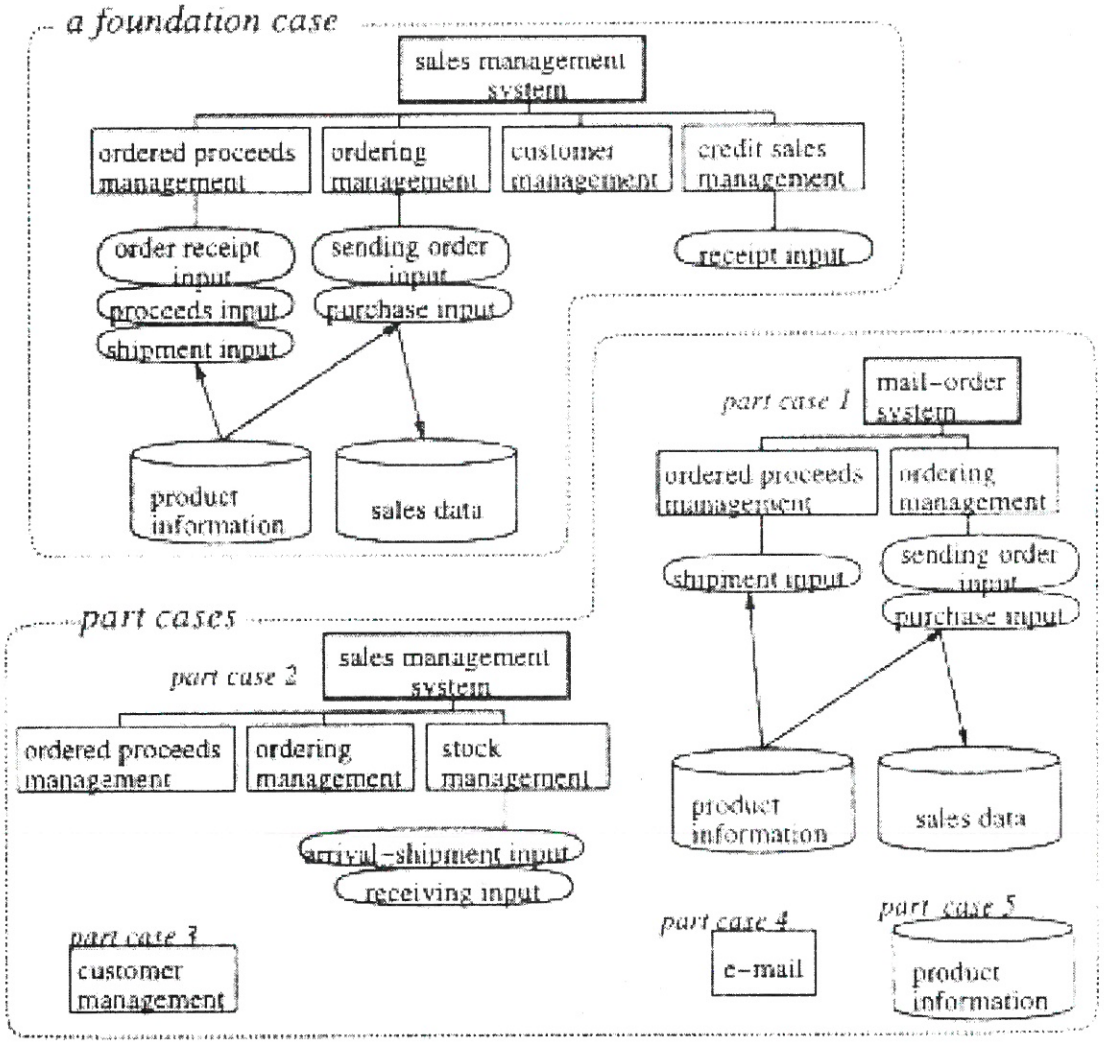

Figure 9. A Foundation Case and Parts Cases

1. **Comparisons of system nodes**

A part case 2 is connected to the foundation case, because the ``sales management system" node is included in the former. The other nodes in the part cases did not match the nodes in the foundation case.

2. **Comparisons of sub-system nodes and operation nodes**

A part case 1 is not common here, because the system node in part case 1 did not overlap a node in the foundation case. Since one of the system nodes in part case 2 was the same as a node in the foundation case, "ordered proceeds management" and "ordering management" sub-system nodes were matched and overlapped the sub-system nodes in the foundation tree. A "stock management" sub-system node in part case 2 could not be matched, so the node was attached

to the foundation case with operation nodes. A "customer management" sub-system node in part case 3 was a match. However, since the node has no system node in part case 3, it is not clear if the "customer management" subsystem node is related to the "sales management system". Therefore, "customer management" could not be connected. An "e-mail" sub-system node in part case 4 could not be connected because the same node did not exist in the foundation case.

3. **Comparisons of data nodes**

Data nodes are not influenced by any other nodes. Since "product information" and "sales data" in part case 1 are included in the foundation case, part case 1 was connected to the foundation case through the data nodes. Part case 5 was connected to the foundation case through the "product information" node.

The results of the connections are shown in Figure 10. There are the part cases which could not be connected. From these cases, potential cases which include all possible connectable situations can be generated. In this example, four trees were generated. All of the potential cases are presented to users, and they will specify suitable or unsuitable nodes so that more information should be gathered. The same way, the cases are repeatedly connected. The end result of the example is shown in Figure 11.

5.2 A Comparative Experiment

We will examine the effectiveness of function trees that are generated by connecting certain cases. Let us consider the following example:

1. A report about a business is given to eight subjects who are not SEs.

2. By referring to the report, four of the subjects indicate suitable or unsuitable nodes for six cases which are similar to the target business in. Sets of function trees have been generated by connecting

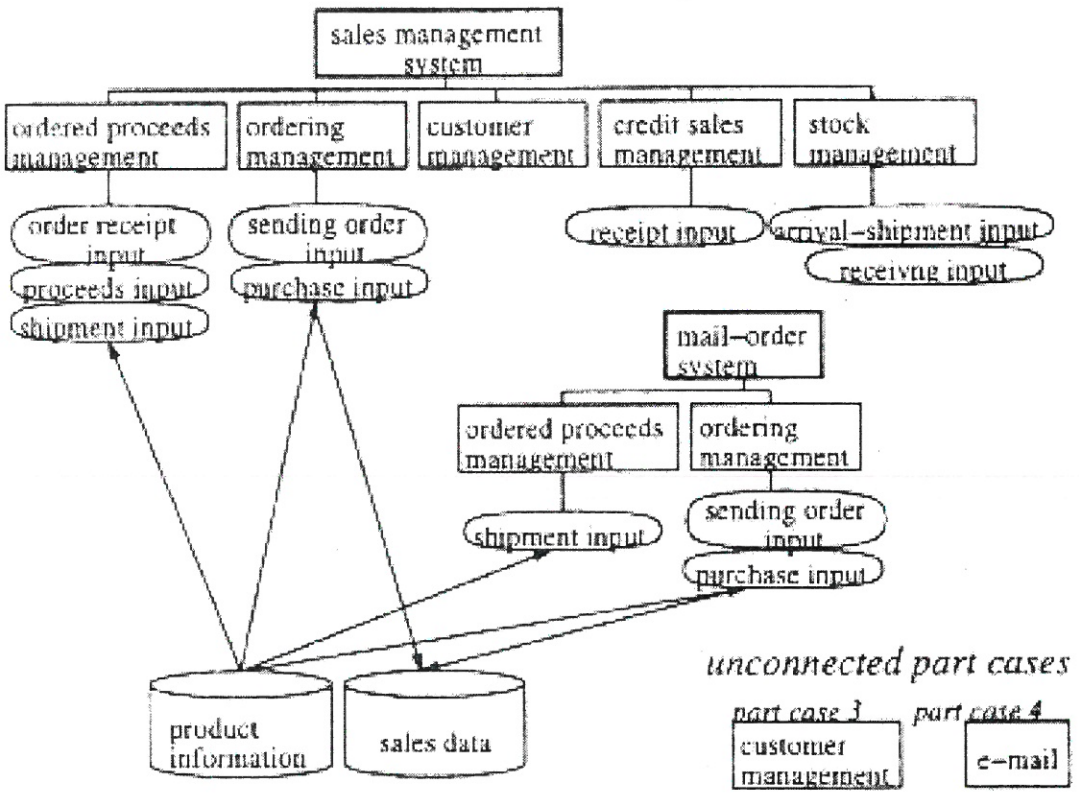

Figure 10. The Results of Connections

these results. The other four subjects generated sets of function trees by means of the report and of one similar case.

3. We compared each of the eight sets of function trees that were generated by the eight subjects with a set of function trees that were generated by a skilled SE. The latter function trees had 28 nodes.

Table 1. Result of This Experiment(Average Number of Nodes)

node	*consistent*	*insufficient*	*excessive*
connecting six similar cases	26.25	2.75	4.50
referring a similar case	16.25	12.75	2.75

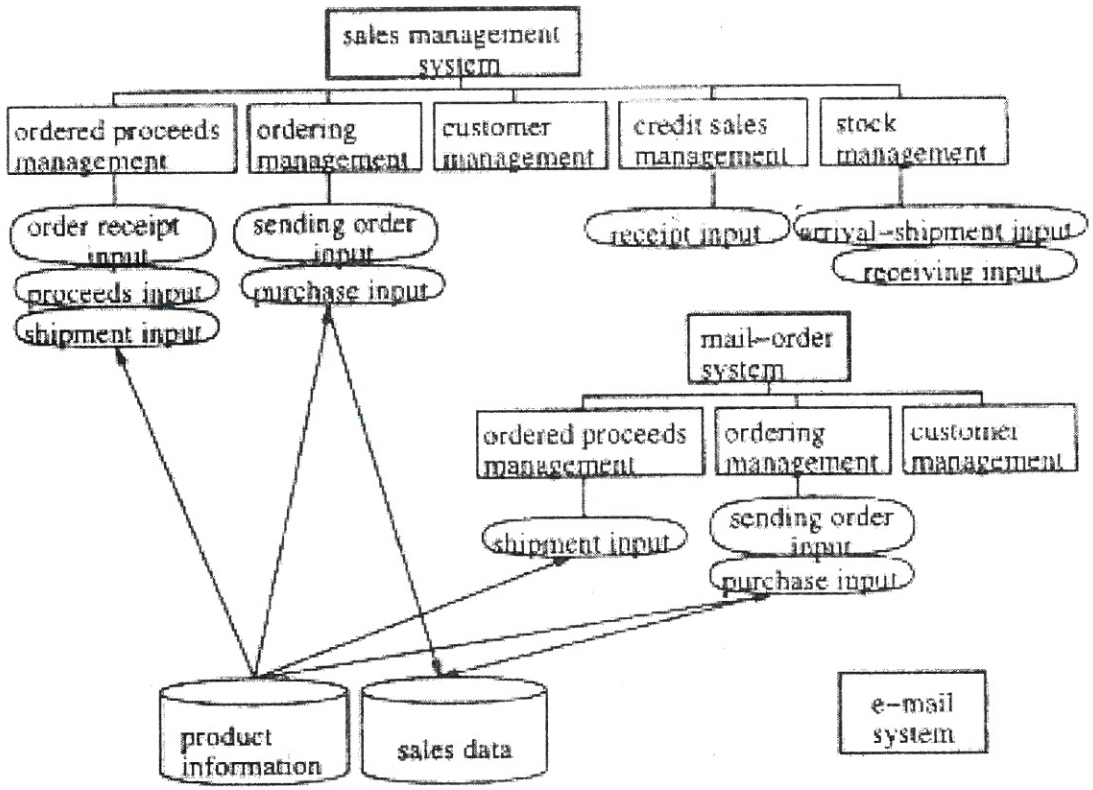

Figure 11. An Example of the Final Result

The results are shown in Table 1. The values in this Table indicate the average number of nodes. A consistent number of nodes shows how many nodes in the subject's function trees are included in the skilled SE's function trees. An insufficient number of nodes shows how many nodes in the skilled SE's function trees are not included in the subject's function trees. An excessive number of nodes shows how many nodes in the subject's function trees are not included in the skilled SE's function trees.

As a result, the set of function trees that was generated by connecting certain cases is closer to a set generated by a skilled SE than a set of function trees that was generated by referring to a similar case. It is clear that a combination by conversations is productive.

6. Conclusion

In this paper, we proposed a CSS configuration design support system: CIDAC. Its goal is to improve CSS configuration design efficiency and skill apprehension support for novice SEs.

In addition, we proposed a function tree generator in CIDAC. By using a case base which contains already generated function trees, a set of function trees can be generated by connecting certain cases. Insufficient nodes in a set of target function trees can be reduced by using certain cases. The only information which the generator needs is the users' judgment on whether the nodes of each retrieved case are suitable or unsuitable.

REFERENCES

1. FRANCIS, B., **CLIENT/SERVER The Model for the '90s**, DATAMATION, February 1990, pp.34-40.

2. DAVIS,D.B., **Clients, Servers and The Glass House**, DATAMATION, November 1991, pp.72-76.

3. RAUCH,W.B., **Distributed Open Systems Engineering,** WILEY COMPUTER PUBLISHING, 1996.

4. HIRAMATSU, A., IKKAI, Y., OHKAWA, T. and KOMODA, K.,: **Requirement Analysis for Client Server System Design Based On Case Base**, Proceedings of 15th IMACS World Congress - Scientific Computation Modelling and Applied Mathematics,. Berlin, Germany, August 1997 (in press).

5. HIRAMATSU, A., NAITO, A., IKKAI, Y., OHKAWA, T. and KOMODA, K.,: **Case Based Function Tree Generation for Client Server System Configuration Design,** Proceedings of 1997 IEEE Int. Conf. on Systems, Man and Cybernetics (SMC'97), Orlando, Florida, USA, October 1997 (to appear).

6. HUGHES, K.J., RANKIN, R.M. and SENNETT, C.T., **Taxonomy for Requirement Analysis**, Proceedings of the First Int. Conf. on Requirement Engineering, Colorado Springs, Colorado, USA, April 1994, pp.176-179.

7. GRADY, J.O., **System Requirement Analysis**, MCGRAW-HILL INC., 1993.

8. HIRAMATSU, A., IKKAI, Y., OHKAWA, T. and KOMODA, K., **A Composition Method of Explanation for Client Server System Configuration Design Based on Signify Judgment**, T.IEE Japan, Vol.177-C No.5, May 1997, pp.618-624 (in Japanese).

9. KOLODNER, J., **Case-Based Reasoning**, MORGAN KAUFMANN PUBLISHERS, 1993.

10. MAHER, M.L. and GOMEZ DE SILVA GARZA, A., **Developing Case-Based Reasoning for Structural Design**, IEEE EXPERT INTELLIGENT SYSTEMS, Vol. 11, No.3, June 1996, pp. 42-53.

11. REDMOND, M., **Distributed Cases for Case-Base Reasoning; Facilitating Use of Multiple Cases**, AAAI-90, 1990, pp. 304-309.