COBOL 도메인에서의 사례기반 소프트웨어 유지보수 지원 전문가시스템.

이재원*, 이재규*, 김우주**

* 한국과학기술원 테크노경영대학원 경영정보전공
** 전북대학교 산업공학과

요약

소프트웨어 유지보수는 소프트웨어 생명주기에서 가장 비용이 많이 드는 부분이다. 따라서 이 유지보수 단계에서의 생산성의 향상은 전체주기에서 가장 중요한 과업중의 하나이다. 이를 위해 본 연구에서는 COBOL 언어 도메인에서 유지보수 할 수 있는 대상을 제안해주고 과거 정보시스템 변경사례를 프로 그램의 소스코드 수준까지 보여줌으로써 유지보수를 지원하는 사례기반 방법론을 제시하였다. 이 방법론은 한국전력공사을 응용대상으로 한 사례기반 추론 및 검색 시스템으로 개발되었다.

본 시스템에서는 하나의 사례를 한 기간에 요청된 여러 유지보수 사건중 하나의 사건으로 정의했고 이에 따라 사례를 표현하였다. 이 표현에 따라 사례를 인덱스하여 구조화된 사례베이스를 구성하였다. 사례기반 소프트웨어 유지보수 지원과정은 다음과 같은 네 단계를 갖는다. 첫째, 여섯가지 단위요구 형태와 관계된 키워드로 유지보수 요구사항의 입력내용을 표현한다. 둘째, 완전매칭 및 추론 전략을 이용하는 부분 매칭에 의해서 사례를 검색하거나 추론한다. 셋째, 추출된 사례의 결과부분을 이용하여 스텝 수준의 대상을 제안한다. 넷째, 제안된 스텝결과를 모듈추론기반 시스템으로 보완적 추론을 해 대상 영역을 좀더 축소시켜 사용자에게 제공한후 유지보수 행위후에 이를 사례로 저장하는 사후처리 과정을 수행한다.

위와 같은 방법론에 의한 본 사례기반 소프트웨어 유지보수 지원 전문가 시스템은 전문가 시스템 개발 도구인 UNiK을 이용하여 개발되었다.

Case Based Software Maintenance Supporting Expert System for COBOL Domain.

Lee Jae won*, Jae kyu Lee*, Woo ju Kim.**

* MIS Program, Graduate School of Management, KAIST
** Dept of Industrial Engineering, Chun-buk Univ.

ABSTRACT

Software maintenance is the most expensive phase in the software life cycle. Enhancement of productivity is one of the important issues in the software maintenance phase. For the reason, we suggest a case based methodology for COBOL language domain to recommend the maintainable targets and to show the IS altemation history at the level of source code to recall the historical success stories. We develop the case based reasoning and retrieval system for software maintenance with application domain, KEPCO (the Korea Electric Power COrporation). In the system, we represent the architecture of a case which is defined as one event of software maintenance. With the case architecture, we index the cases and build the case-base. The procedures for case based software maintenance supporting is composed of the following steps : 1) Requirement input with six unit requirement types and its related keyword, 2) Case retrieval and reasoning with perfect and partial matching algorithm , 3) Step recommendation with the output part of retrieved cases, 4) Post processing which consisted with filtering out and case storage.

A prototype case based software maintenance supporting expert system is implemented as a part of SWMES (Software Maintenance Expert System) using a expert system developement tool UNiK.

1. Introduction

Software maintenance is recognized as most expensive phase of the software life cycle. The system maintainer is frequently presented with code with little or no supporting documentation, so that the understanding required to modify the program carried out mainly by code.

Real world business information systems are very expensive to maintain. As soon as they are delivered to the production field, they start on going through the continuous process of evolution and modification. This process is dictated by changes to the system environment and user's requirements.

Software maintenance is defined to be any work that is undertaken after delivery of a software system [Robinson et al, 1991]. It has been recognized that software maintenance can consume as much as 70% of the cost during the life cycle of the system [Leintz and Swanson, 1980]. Research effort has been almost exclusively devoted to the development phase with the goal of reducing the cost of maintenance by using new development techniques, but the problem itself of the maintenance of existing software is ignored many times.

Software maintenance has been split into three subtasks, namely corrective, adaptive and perfective maintenance [Swanson, 1976]. It is estimated that 65% of the maintenance phase are taken up with perfective maintenance and 17% is corrective maintenance [Leintz and Swanson, 1980].

The documentation of system maintenance and requirements may or may not be updated as the program is modified. If not, this makes future maintenance difficult. For the reason, case based reasoning system using the previous maintenance cases can support software maintenance works and make future maintenance easier when there was no documentation. CBR system can be regarded as one for the documentation. Future maintenance activity can use the cases with the codes.

1.1. Problem Description in Real Business World.

KEPCO is one of the largest company which has information system infrastructure in Korea. In KEPCO, as a monopolistic electric power supply corporation in Korea, there are large and complex software systems developed with or without the aid of modern software engineering techniques. The program modification on systems developed without the aid can be virtually impossible without reading original code because there is little documentation of maintenance.

KEPCO has already developed many large-scaled software systems using COBOL language in large portion and CASE tool in later period. In recent years, KEPCO is in the process of constructing enterprise-wide integrated system. Through this process many software's resources are reused by standardization. Representative example of reuse is the coding regulation and the use of copy program for data field usage.

In the KEPCO's system already developed before CASE tools become popular and its IS environments such as COBOL domain without system documentation, we can easily find the characteristics of S/W maintenance. It is repetitive with similar requirements and has used similar codes at each time. These characteristics are suitable for using case based technique.

The large portion of software maintenance and management is carried out in the department of software development. The COBOL-used software system developed in earlier period is the major target of software maintenance in KEPCO. The bulk of this maintenance work is the perfective maintenance task classified in [Swanson, 1976]. Most time consuming and difficult part in this maintenance process are the maintainable target recognition or comprehension. Even when there is documentation exists, it has found that the time spent studying the code took up more than three and a half time the amount of time studying the documentation [Fjeldstrad et al, 1983]. It is especially serious for novice engineer.

1.2. Objectives and Scope of Research

we will consider and do the followings as our research objectives.

1. Suggest a methodology to recommend maintainable targets using CBR.
2. Show the IS alternation history at the procedure level of source code to recall the historical success stories.
3. Development case based reasoning and retrieval system for S/W maintenance.

For the above objectives of our research, we will carry out the development of a case based expert system which can do next listed things as our approaches for supporting S/W maintenance.

1. Case representation of S/W maintenance.
2. Case indexing and Case-Base building.
3. Query to Case-Base (CB).
4. Reasoning of a maintainable S/W target using CB.

2. Case Based Reasoning

There are many similar definitions about case based reasoning.

A case based reasoner solves new problem by adapting solutions that were used to solve old problems [Risebeck and Schank,1989]. Case based planning is the idea of planning as remembering [Hammond, 1989]. Case based reasoning is a general paradigm for reasoning from experience. It assumes a memory model of representing , indexing, and organization previous cases and process model for retrieving and modifying old case and assimilating new ones [Slade, 1991]. Case based reasoning can mean adapting old solutions to meet new demands, using old cases to explain new situations, using old cases to critique new solutions, or reasoning from precedents to interpret a new situation or create an equitable solution to a new problem [Kolodner, 1992]. But we know that. CBR are not completely new types of expert system building method, they represent a new, higher level knowledge representation environment and collection of new search algorithms that sit on top of existing hybrid ES building tools [Harmon, 1991].

[Risebeck et al, 1987] and [Slade, 1991] described the representative memory model and process model of CBR. For the process model, variuos types of knowledge structures are supported.

- Indexing rules [Schank et al, 1986].
- Case memory
- Similarity metrics
- Modification rules
- Repair rules

We use the method of building blocks of matching and ranking process. In this we construct parallel representation by finding structural correspondences between new case and old stored cases [Kolodner, 1993]. Computing similarity method of corresponding features using abstraction hierarchy is quoted and hierarchical network, but it is based on nearest neighbor method. Requirement engineering in [Renzhang et al, 1995] was referenced somewhat. Most similar related work to our domain, software maintenance, is SQUAD, Software quality control advisor [Kitano et al. 1992]. It collects cases from throughout NEC corporation and emphasize the cost effectiveness of case base.

2.1. Software Reuse

Reusable software asset is any software-related and, non-hardware item seemed valuable. Example software asset are software part (e.g. systems, function, procedures, package, data type definitions, variable definition, "objects"), designs (e.g. structure charts, DFDs), specifications (e.g. business models, and proposals for building software), test data, code templates, and notebooks of software documentation.

Reusability of software asset has been defined as a general engineering principle whose importance derives from the desire to avoid duplication and to capture commonality in undertaking classes of inherently similar tasks [Ramesh and Rao, 1994]. Part based software reuse is the use of parts from previous systems to build new one. It is widely seen as key technology for improving software flexibility, quality and productivity [Arnold and Frakes, 1991].

Our research are largely related on program comprehension and code analysis/parsing [Robinson et al, 1991] for automatic frame generator from COBOL. Inverse (reverse) engineering, annotation and documentation on existing program and so on are software reengineering and related to our research too. There are researches on software learning /maintenance using CBR [Williams, 1988] and software reuse using CBR [Fouque and Matwin, 1993].

3. Knowledge and Case Representation

Through a analysis of maintenance requirement with representative 19 official documents of case management in low tension charge system of KEPCO, we could refine the maintenance type and a representation method of that maintenance requirement. All cases are classified in new, modify and deleting with their maintenance work type. We also could find a requirement content can be represented with rule typed structure and formula treatment typed structure. This rule type and formula type structure can be represented with composition of requirement type and its keyword on that requirement type. Requirement types are six types like I/O, If-clause, assignment, DB-interface, constants dealing and supplementary others.

With historical contents of seriesal and single software updating event , we could get maintenance cases and KEPCO's maintaining method of their information system. There are many requirement style like regulation alternation, process or work flow improvement, simple formula alternation and so on. We found representative two requirement structure for a case's maintenance requirement representation such as rule type and formula type structure. Rule type can be said as **IF-THEN** structure and formula type can be said only **THEN** structure. These classification of requirement structure is important in case memory organization and reasoning. It will be dealt more in case representation section and in reasoning process below.

A classified event representation method with the maintenance working types can be differentiated for a requirement representation of case.

On this analysis, we first construct basic framework for IS analysis and representation.We basically build blocks for representation of IS and its requirement requirements with own type and syntax.

- Keyword Base construction in focus of related regulation, work process and forms of official documents.
- Representation of software maintenance requirements using rule, formula treatment structure which composed with requirement types and related keywords for S/W maintenance.
- Representation of IS using related keywords and IS representation methodology with own syntax.
- Construction of relation represent method between IS and keyword.

3.1. Representation of system knowledge

Representation method of IS knowledge is originated from the research of our overall system named SWMES.

All Information systems have its task. This task has hierarchical relationship with program and data base. We consider here the task of our system as example the low tension charge system of KEPCO.

A program have steps and each step can have its steps. Each step contents has their own function. Data base has similar structure to program. It have attributes contrast to the step in program but attribute contents is only the content which do not has its own function.

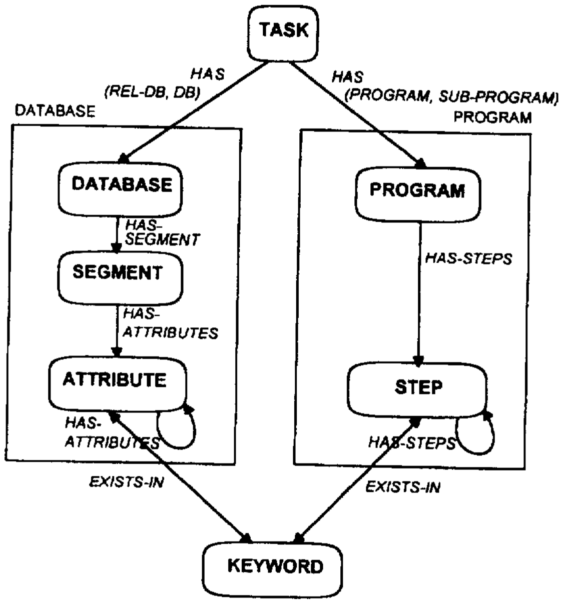

Figure 3-1. Structure of information systems

All steps in program can be classified with six step types by their functional roll. COBOL syntax in our consideration, 'ADD', 'MOVE', 'COMPUTE', 'DIVIDE' and so on, can be a example. We can classifyas a 'ASSIGNMENT' type with its own related reserved words. 'LOOP' type has a reserved word like 'PERFORM', 'UNTIL', 'BY' in the order. 'Input/ Output (I/O)' type has a reserved word 'DISPLAY'. 'DB-INTERFACE' type has a reserved words 'OPEN', 'READ', 'WRITE' and so on. The other types are similar to above type with their functional roll.

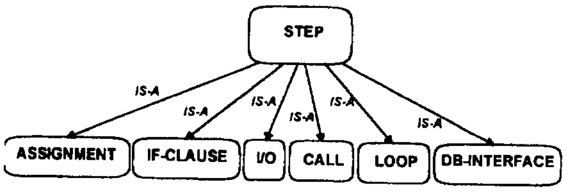

Figure 3-2. Categorization of a step by functional roll.

We are considering a system knowledge representation method. A task is represented as a example with our exemplary service of 'low tension charge system'.

{{ Low tension-Charge SYSTEM
IS-A : TASK
PROGRAM : CAU11JOJ CAM30JOJ ....
SUB-PROGRAM : CAKEYED1 LOWDBSET CADBXUPS CAFLDEXM CACHKYMD CADA YCALCAM30EXT CALSHIFT CAMSGSET CAJ94101 CAJ92201 ....
REL-DB : Low tension-Charge DB ....

```
DB : CAPMEDNG ...
}}
```

COBOL program structured with the four division named identification division, environment division, data division and procedure division. We used only procedure division in our system. Procedure division has the instruction statements executing functional capability in the its section or paragraph. Section and paragraph name is attached with that module name to step representation name.

COBOL program structure
    IDENTIFICATION DIVISION.
    ENVIRONMENT DIVISION.
    DATA DIVISION.
    PROCEDURE DIVISION.
        SECTION.
        PARAGRAPH.
        Instruction with reserved word.

Each functional instruction statements in step have six categorized step types and own steps named as a representation frame. CAM30JOJ-3000-OVER- FARE-COMPUTE is a example of section name. Paragraph name is similar. Like this named section or paragraph has step of instruction statement. These step name is composed with upper section name and serial step number.

```
{{ PROGRAM
REL-TASK :
LANGUAGE :
MAIN-SUB : (MAIN | SUB)
TYPE : (ON-LINE | BATCH)
INPUT-DB :
OUTPUT-DB :
CALLED-MODULE :
INPUT-PARAMETER :
OUTPUT-PARAMETER :
HAS-STEPS :
KEYWORD :
}}
```

Each slot value in above frame is the program specification. 'REL-TASK', 'LANGUAGE', 'MAIN-SUB', 'TYPE' slots are environmental descriptions. 'INPUT-DB' and 'OUTPUT-DB' are program related DB name and its attributes which are field name or variable name in the program defined or copied in the data division. 'CALLED-MODULE' is the subprogram name called by this named program or other main program executed. A 'CALLED-MODULE' is called by a 'INPUT PARAMETER' and 'OUTPUT PARAMETER' which defined DB name in the data division. 'HAS-STEPS' slot has all step name of six type and its own type appeared in the procedure division of program. 'KEYWORD' slot has all keywords appeared in the program.

We will deal 'STEP' and six step type representations. These steps are appeared in the section or paragraph of COBOL structure.

```
{{ STEP
A-STEP-OF :
HAS-STEPS :
}}
```

Step is for the section and paragraph representation. Each section will be a step of upper level program frame and paragraph will be a step of section appeared with the step.

```
{{ ASSIGNMENT
IS-A : STEP
LHS-KW :
RHS-KW :
}}
```

ASSIGNMENT step is made with the reserved words such as 'ADD', 'MOVE', 'COMPUTE', 'DIVIDE', 'MULTIPLY' and so on. 'LHS-KW' is the assigned keyword and 'RHS-KW' is the assigning keyword.

```
{{ IF-CLAUSE
IS-A : STEP
KEYWORD :
}}
```

It represent a conditional step. Related reserved words are 'IF', 'ELSE', 'THEN', 'WHEN', 'SEARCH' and so on.

```
{{ LOOP
IS-A : STEP
KEYWORD :
}}
```

Related reserved words for this loop representation is 'PERFORM' following 'UNTIL'.

```
{{ I/O
IS-A : STEP
TARGET-OBJECT :
INPUT-KW :
OUTPUT-KW :
}}
```

I/O step is appeared with the reserved word 'DISPLAY', 'PRINT' and so on. 'TARGET-OBJECT' is the instrument or object for input or output.

```
{{ DB-INTERFACE
IS-A : STEP
R/W-TYPE :
REL-DB :
KEYWORD :
}}
```

DB interface is executed for the DB reading or writing.

```
{{ CALL
IS-A : STEP
INPUT-PARAMETER :
OUTPUT-PARAMETER :
CALLED-MODULE :
}}
```

CALL is for a subprogram call or executing other main program with the parameter. The parameters are in 'INPUT' or 'OUTPUT PARAMETER'. 'CALLED-MODULE' is the subprogram or other main program name. These all steps and system knowledge representation are systematically have the inverse relationt to the relation of all frames like A-STEP-OF to the slot HAS-STEPS.

```
{{ KEYWORD
FORWARD :
BACKWARD :
SYNONYM :
EXISTS-IN :
}}
```

Keyword representation is constructed with parent and child relation of 'FORWARD' and 'BACKWARD'. 'SYNONYM' is for the same keyword in its meaning. Keywords were gathered from rule, regulation, DB dictionary and other screen and table.

DB and its segment, attribute have a lot of slots but explanation on each slot will be omitted.

```
{{ DATABASE
REL-TASK :
DB-STRUCTURE : (HIDAM | HDAM)
ACCESS-METHOD :
KEY-FIELD :
USED-PROGRAM :
HAS-SEGMENT :
}}
{{ SEGMENT
REL-TASK :
A-SEGMENT-OF :
PARENT :
CHILD :
KEY-FIELD :
HAS-ATTRIBUTES :
LENGTH :
USED-PROGRAM :
}}
{{ ATTRIBUTE
ATTRIBUTE-OF :
HAS-ATTRIBUTES :
LENGTH :
TYPE :
KEYWORD :
USED-PROGRAM :
}}
```

Other system knowledges of 'FILE' similiar to DB knowledge is exist. Above all structure is used for automatic knowledge acquisition from COBOL source codes

3.2. Case representation

Case representation method is very important part of case based reasoning as it decided the efficiency of CBR and so various.

We can assume and use one case document to one software maintenance. One case management document is the set of whole case report which was worked at the same period by a request for system maintenance. Each *case management document* has single or multiple *requirement name of maintenance.* Requirement name of maintenance have elemental requirement expressions in its structure with the requirement representation type. Most requirement names of maintenance in one case management document has no relation to each others. Single or multiple result *module name of maintenance* is connected to the upper written requirement name of maintenance.

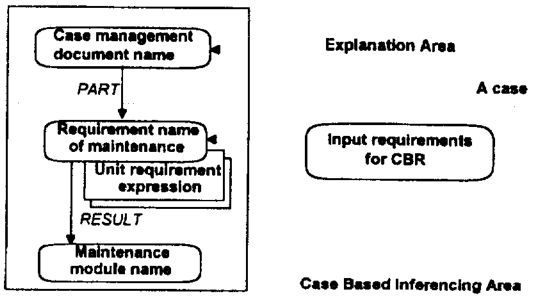
Case management document name
PART
Requirement name of maintenance
Unit requirement expression
RESULT
Maintenance module name
Explanation Area
A case
Input requirements for CBR
Case Based Inferencing Area

Figure 3-3. Structure of software maintenance requirement and case.

One case is defined as ***a requirement name of maintenance***. It has a upper level case management document name of S/W maintenance as event's supplementary explanation of all for the circumstances and lower level related maintenance module names as event's results. The elemental requirement expressions have the most important case indexing vocabulary for case representation as slots of that frame. This case indexing vocabulary is basically constructed with the contents of maintenance work in the requirement name of maintenance frame. As cases are blocked by using requirement representation types like rule and formula type structure, many works like finding correspondence between input case and retrieved case was solved with the block's structure.

Structures of requirement representation are like rule, formula, parameter and other process improving activity type. But we have defined the representative requirement structure as **rule type** with IF-THEN style and **formula type** with only THEN style.

We can construct requirement structure with example case KEPCO(IS)109.05 -2754.

- " If over usage hour per month is greater than 450, additional extra 50% of electric power charge is applied to the over usage quantity for the next electric power contract category such as (General, Industry, Temporary (B)) "
- " Cut 50% basic usage charge off in the month for no electric power usage."

Above requirement representation contents are the rule type expression so it can be represented with IF-THEN structure of the rule.

A case must have input data area (problem description part) and output area (solution description part) which will be applied through the overall system procedure. Requirement representation in our system will be the input area and related module and step representation frame will be the output area.

Requirements for a maintenance work are described with one or more unit requirement which have one within six requirement types. Each unit requirement with one requirement type and only the IF-CLAUSE type have to be duplicated for representation of conjunctive and disjunctive requirement keyword value. For the representation, **'If A and (B or C) then CALCULATE D and A'**, We need next four unit requirement expressions in its requirement name frame with IF-THEN structure.

Unit requirement with type IF-CLAUSE with keyword ***A***.
Unit requirement with type IF-CLAUSE with keyword ***B*** *or* ***C***.
Unit requirement with type ASSIGNMENT with keyword ***A*** .
Unit requirement with type ASSIGNMENT with keyword ***D***.

Case representation is based on own block's structural correspondence between input case and retrieved case. Case structure is from Fig 3-3.

```
{{ Case Management Name for Maintenance
ACCEPTED_DEPT :
DATE_OF_ENFORCEMENT :
REFERENCE_DEPT :
TITLE :
REQUEST_DEPT :
PART :
}}
```

Case management name for maintenance act the role of explanation for the case's request background of maintenance.

```
{{ Requirement Name for Maintenance
IS-A : CASE
PART-OF :
UNIT_OPERATION_NAME :
REQUIREMENT_UNIT :
RELATED_CASE_NAME :
CONTENTS :
RESULT :
}}
```

'UNIT OPERATION NAME' is defined within the task, low tension charge system, like our 'charge control'. If our system is enlarged to cover overall company wide scaled information system, it will be the important classification dimension and case indexing vocabulary. 'REQUIREMENT UNIT' is a set of elemental expressions for case based inference and case indexing vocabulary selection process. When requirement of maintenance is rule type, it will be filled with 'IF-CLAUSE' and then part like 'ASSIGNMENT', 'I/O' or 'DB-INTERFACE' and so on. When requirement of maintenance is formula type, it will be filled with non 'IF-CLAUSE' elemental requirement expression.

```
{{ Elemental(unit) requirement type
IS-A : REQUIREMENT_UNIT
REQUIREMENT_TYPE :
REQUIREMENT-KEYWORD :
RELATED- KEYWORD :
MAINTENANCE_WORK_TYPE :
PART-OF :
}}
```

'MAINTENANCE WORK TYPE' and has one of three type, 'NEW', 'MOD' or 'DEL'. 'REQUIREMENT CONTENTS' is the one case' whole request contents in text. 'MAINTENANCE WORK TYPE', 'REQUIREMENT TYPE' in six type and 'REQUIREMENT KEYWORD' are major case indexing vocabularies in our case representation. One case of requirement can have more than one elemental requirement expression. All elemental requirement expressions are represented to each other with conjunctive relationship on their requirement keyword. Disjunctive relationship are represented with keywords overloading. 'REQUIREMENT KEYWORD' is the keyword which have used in our system's semantic network for case indexing. If the input keyword is not enrolled, we must check the synonyms of past used ones at first and there are no such keyword on semantic network then inference on that keyword is impossible. Later it enrolled on network by the automatic keyword indexing management system in our system at the end of case inference work.

```
{{ Related Result Module of Maintenance
RESULT-OF :
MODULE_WORK_TYPE :
PROGRAM_TYPE :                    ;
MAIN-SUB :
STEP :
RELATED_STEP_NAME :
MANIPULATION_CONTENTS :
}}
```

Output result of case conceives the all maintenance step information of the related result module regard of all maintenance work types. 'RELATED_STEP_NAME' is constructed with step name, step type, step work type and matching variable to the requirement keyword in the related program source code. Step name consist with program name, section or paragraph name in COBOL syntax and the serial number of checking reserved word within one sentence in that section or paragraph.

Example document 'KEPCO(IS)190.05-2754' contain two major cases and two supplementary modified cases. Most of all, elemental requirement name of case detailed more than other parts. Related result module of maintenance part are briefly described with most related two modules as there are too many related modules to cover all contents in this paper.

```
{{ KEPCO(IS)190.05-2754
IS-A : Case Management Name for Maintenance
ACCEPTED_DRPT. : Dept. Manager of Data processing
DATE_OF_ENFORCEMENT : 94/01/01
REFERENCE_DEPT: Chief of Electronic data processing
     center, Seoul.
TITLE : Complementation of the rules for electric power
     supply.
REQUEST_DEPT: Dept. of business and operation.
     The office of business.
PART : KEPCO(IS)190.05-2754-1,
     KEPCO(IS)190.05-2754-2,
```

KEPCO(IS)190.05-2754-3,
KEPCO(IS)190.05-2754-4

}}

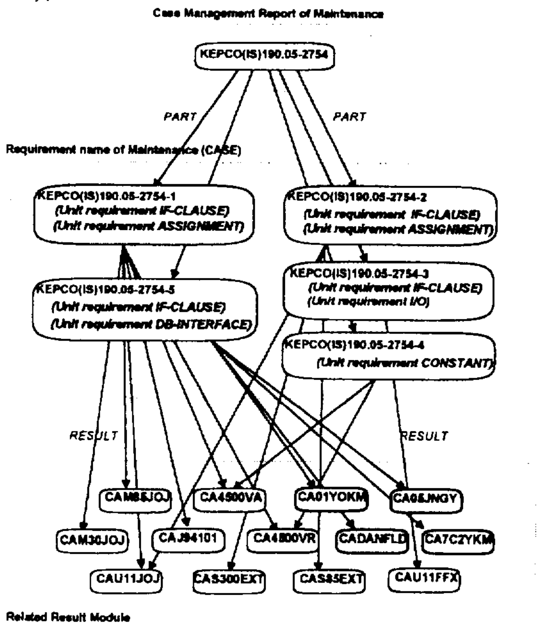
Case Management Report of Maintenance
KEPCO(IS)190.05-2754
PART
PART
Requirement name of Maintenance (CASE)
KEPCO(IS)190.05-2754-1 (Unit requirement IF-CLAUSE) (Unit requirement ASSIGNMENT)
KEPCO(IS)190.05-2754-2 (Unit requirement IF-CLAUSE) (Unit requirement ASSIGNMENT)
KEPCO(IS)190.05-2754-5 (Unit requirement IF-CLAUSE) (Unit requirement DB-INTERFACE)
KEPCO(IS)190.05-2754-3 (Unit requirement IF-CLAUSE) (Unit requirement I/O)
KEPCO(IS)190.05-2754-4 (Unit requirement CONSTANT)
RESULT
RESULT
CAM85JOJ
CA4500VA
CA01YOKM
CA06JNGY
CAM30JOJ
CAJ94101
CA4500VR
CADANFLD
CA7C2YKM
CAU11JOJ
CAS300EXT
CAS85EXT
CAU11FFX
Related Result Module

Fig 3-4. example of case representation.

{{ KEPCO(IS)190.05-2754-1
IS-A : Requirement Name for Maintenance
PART-OF : KEPCO(IS)190.05-2754
UNIT_OPERATION_NAME : Charge Control.
REQUIREMENT_UNIT :
(IF-CLAUSE, NEW, (450 hour over usage quantity per month))
(IF-CLAUSE, MOD, (Contract category))
(ASSIGNMENT, MOD, (Electric power charge, Over usage quantity charge))
RELATED_CASE_NAME :
KEPCO(IS)190.05-2754-3
KEPCO(IS)190.05-2754-5
REQUIREMWNT_CONTENTS : " If over usage hour per month is greater than 450, additional extra 50% of electric power charge is applied to the over usage quantity for the next electric power contract category such as (General, Industry, Temporary (B)) "
RESULT : KEPCO(IS)190.05-2754-1-CAM30JOJ
KEPCO(IS)190.05-2754-1-CAM85JOJ
KEPCO(IS)190.05-2754-1-CAJ94101
}}
{{ KEPCO(IS)190.05-2754-1-CAM30JOJ
IS-A : Related Result Module of Maintenance
RESULT-OF : KEPCO(IS)190.05-2754-1
MODULE_WORK_TYPE : MOD
PROGRAM_TYPE : COBOL
MAIN-SUB : MAIN
STEP : 130
RELATED_STEP_NAME :
(CAM30JOJ-3321-FARE-COMPUTE-5-1, CALL, NEW, (CAJ94101), (CADANLD, CAOVERFLD))
(CAM30JOJ-3321-FARE-COMPUTE-5-2, CALL, MOD, (CAJ92201), (CADANLD))
(CAM30JOJ-3321-FARE-COMPUTE-5-2, CALL, DEL, (CAJ91601), (CADANLD))

⋮

MANIPULATION_CONTENTS : "Append extra electric power charge calculation for 450 hour over usage quantity per month "
}}
{{ KEPCO(IS)190.05-2754-1-CAJ94101
IS-A : Related Result Module of Maintenance
RESULT-OF : KEPCO(IS)190.05-2754-1
MODULE_WORK_TYPE : NEW
PROGRAM_TYPE : COBOL
MAIN-SUB : SUB
STEP : 351
RELATED_STEP_NAME :
(CAJ94101-0000-MAIN-PROCESS-2, ASSIGNMENT, NEW, (CAJOJ-OVER-KWH-FARE, CAJOJ-OV-KWH, CAJOJ-OV-KWH-BASE))
(CAJ94101-3200-KWH-FARE-COMPUTE-3, IF-CLAUSE, NEW, (CAJOJ-OVER-SIGN))
(CAJ94101-3210-OVER-FARE-COMPUTE-1, ASSIGNMENT, NEW, (CAJOJ-OV-KWH-BASE))
(CAJ94101-3210-OVER-FARE-COMPUTE-2, IF-CLAUSE, NEW, (CAJOJ-OV-KWH-BASE))
(CAJ94101-3210-OVER-FARE-COMPUTE-2-1, ASSIGNMENT, NEW, (CAJOJ-OV-KWH-BASE, CAJOJ-OV-KWH))
(CAJ94101-3210-OVER-FARE-COMPUTE-2-2, ASSIGNMENT, NEW, (CAJOJ-OV-KWH-BASE, CAJOJ-OV-KWH, CAJOJ-OVER-KWH-FARE))
MANIPULATION_CONTENTS : "Extra electric power charge calculation for 450 hour over usage quantity per month : Sub-module "
}}

4. *Case Based S/W Maintenance Supporting Methodology*

We will consider case based inference methodology for maintainable step recommendation process, perfect matching, partial matching process and post processing.

1. ***Requirement input*** - A new case input with incomplete requirement contents are entered as natural language format. It formalized within our structure like rule type, formula type. Through the requirement structure, requirement contents are expressed with one of six typed unit requirement or multiple ones. Each elemental requirement are expressed with consideration of rule, keyword through user interface and are grouped within the upper requirement structure. All these requirement input process will be carried out with requirement representation flow for case indexing. Go to step 2.
2. ***Perfect matching process*** - We first try exact matching case retrieval on their unit requirement expressions. The case based inference engine try to match new case to old ones by comparing requirement type and keyword of unit requirement expression of

requirement maintenance exactly. If this matching process are completed then go to step 4, else go to step 3.

3. ***Partial matching process.*** - If there are more than one matched elemental requirement expression, we perform matching. We abstracted the keyword of the unmatched requirement type in the semantic network with similarity. We select all partial matched cases in the case memory through this way and cut off same, duplicated case or one of series of maintenance case. And we more cut off and modify cases to fit to the requirement structured unit requirement. Go to step 4.
4. ***Step recommendation*** - After the case retrieval process of perfect and partial matching, this process offer maintainable step recommendation with retrieved case's related result module of maintenance frame. In this process, we narrow the target steps in the related result module by step processing. Step processing are briefly constructed with related step name. Go to step 5.
5. ***Post processing*** - If step recommendation results are input in this process, the results are filtered with module search inference once more then the result offered to user. If it is incomplete, then post processing that contained the full-scaled module search inference for the same requirement input will offer the maintainable step recommendation with step level. The user will manipulate the related COBOL program code to fit to the original maintenance requirements. After all this maintenance process, another case will be indexed and stored in the case memory. Stop process.

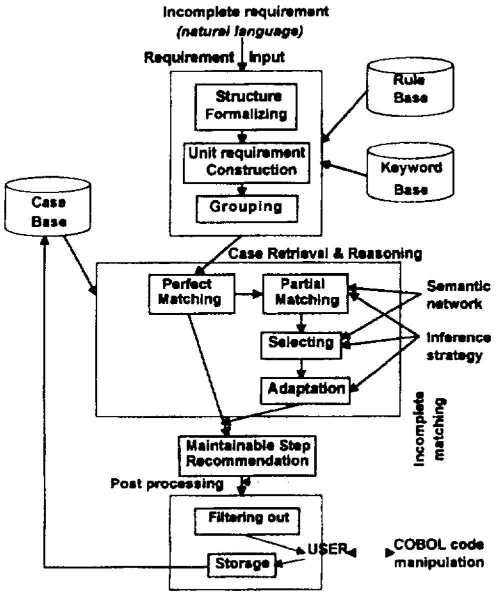

Figure 4-1. Layout of case based S/W maintenance supporting process.

We can consider another requirement representation of KEPCO(IS)193.09-292-3.

{{ KEPCO(IS)193.09-292-3

IS-A : Requirement Name for Maintenance
PART-OF : KEPCO(IS)193.09-292
UNIT_OPERATION_NAME : Charge Control.
REQUIREMENT_UNIT :
(IF-CLAUSE, MOD, (450 hour over usage quantity per month))
(IF-CLAUSE, MOD, (Contract category))
(IF-CLAUSE, NEW, (Combined apartment))
(ASSIGNMENT, MOD, (Electric power charge , Over usage quantity charge))
REQUIREMENT_CONTENTS : " When apply 450 hour over usage charge per month, except customer of the combined apartment from collecting a charge for the next electric power contract category such as (General, Industry, Temporary (B)) "
}}

4.1. Partial Matching Process

Failed at perfect matching process to find suitable case, we try for another case retrieval process with the same requirement input.

Finding correspondence enables the partial matching. If we didn't find matching case on that unit requirement with the same REQUIREMENT TYPE and REQUIREMENT KEYWORD, we can validate that there is no suitable case in that keyword node in that case indexing. So we then try at first to search all partial matched cases on that case memory node with the requirement structure and UNIT OPERATION NAME.

Most of case's requirement structure of partial matching process is rule type structure. As all requirement types are independent to each other type except for IF-CLAUSE, we can example the representative requirement input with THEN part description of ASSIGNMENT type.

With this requirement input, we search all cases on the branch of case memory with the UNIT OPERATION NAME and IF-THEN arc. Related cases are retrieved with IF part's keyword comparison regard of THEN part's coverage or THEN part's keyword comparison on that requirement type. Retrieving with joint IF-THEN part's keyword is already tried in previous perfect matching and failed. If we assume abstracted keyword on the semantic network conceive their specified keyword's result step then this trial is possible. We will not consider this trial with semantic network but the concept will be dealt.

Selecting suitable cases from partial matched cases is executed with inference strategy. Selected cases are verified with elemental requirement's maintenance work type, NEW, MOD, DEL.

These verified cases are delivered to the next process, maintainable step recommendation process. Step processing is the process to show or output the related step infromation to user. As our case based inference is based on parallel representation make computation of correspondence easy, recommendation is carried out easily with the description of requirement input.

4.2. Inference Strategy

Followed assumption or heuristics are dealt as rule in the case retrieval process. There are five assumptions for case based inference like followings.

***Assumption** 1. Only IF-THEN and THEN styled requirement input is permitted.*

So the requirement input with only IF part is impossible for case retrieval processing. There are no requirements for S/W maintenance with only condition. Requirement must have action description.

***Assumption** 2. conjunctive relation between and within requirements types is not permitted.*

***Assumption** 3. Keyword selection in representation of elemental requirement must be specified within semantic network as possible as.*

***Assumption** 3-1. One keyword can have one to many relations with related variables by the COBOL data definition levelized, but one variable must have one to one relation with keyword.*

***Assumption** 3-2. If overloading condition part of requirement input can be sustituted with one specified level keyword with more than one elemental requirement, then it must be.*

In requirement input process, there are four primitive rerequirement input types.

1) Only THEN part type
 (ASSIGNMENT, MOD, (*keyword*))
2) Simple IF-THEN type
 (IF-CLAUSE, MOD, (*keyword1*))
 (ASSIGNMENT, MOD, (*keyword2*))
3) Duplicated IF part type
 (IF-CLAUSE, MOD, (*keyword 1*))
 (IF-CLAUSE, MOD, (*keyword 2*))
 (ASSIGNMENT, MOD, (*keyword 3*))
4) Complex type
 (IF-CLAUSE, MOD, (*keyword 1*))
 (IF-CLAUSE, MOD, (*keyword 2*))
 (ASSIGNMENT, MOD, (*keyword 3*))
 (DB-INTERFACE, MOD, (ketword 4))

This complex type is the mixed type of the second and third types. We will consider inference strategies which is most important in this case based S/W supporting process. We classified this inference strategy into two groups, basic inference strategies and infemece strategies.

***Basic Strategy** 1) Beside the types within IF-THEN structured representation of requirement, all requirement type is basically independent to each other type.*

Because one instruction statement by the requirement type is basically constructed with one reserved word in source code.

***Basic Strategy** 2) For more than one keyword although with same requirement type, only the IF-CLAUSE type is duplicated for the conjunctive relation between keyword.*

So to say duplication of IF-CLAUSE type is the narrowing search space to intersection set concept. Other requirement types' duplication and overloading do the same role as union concept to enlarge the search space. Addition to these basic inference strategy 2, another consideration is needed.

***Basic Strategy** 2)-i) If there is intersection, then the intersection is recommended as maintainable step result else then two primitive type's step results are recommended.*

***Strategy** 1. Duplications of IF-CLAUSE requirement types make diminish the related result space in intersection between related case's result space.*

***Strategy** 2. Duplication or overloading of requirement types except for IF-CLAUSE make broaden the related result space to union space.*

***Strategy** 3. If NEW maintenance work typed requirements are input, then case retrievals try partial matching with its elemental requirements excepted the NEW maintenance work typed elemental requirement.*

***Strategy** 3-1. If NEW maintenance work typed requirements in its THEN part are input, then case retrievals try partial matching with its unit requirements excepted the NEW maintenance work typed elemental requirement. If there are more than one partial matched cases, then similarity measured by abstraction hierarchy with requirement keyword on semantic network.*

***Strategy** 4. If there is no perfect matching cases for the complex requirement input with conjunctive and disjunctive elemental requirement relationship, then pratial matching is carried out with decomposed primitive set of requirement input within non disjunctive relationship.*

***Strategy** 4-1. Similarity in partial matching measures with the number of matched elemental requiement and the requirement type of each unit requirement.*

***Strategy** 5. THEN part of requirement input must be matched for partial matching process, unless maintenance work type of THEN part's elemental requirement is NEW.*

***Strategy** 6. If requirement input is more general than exist case, then related all specific cases are retrieved.*

The mean of specificity in requirement representation is more condition part existing, so to say more 'IF-CLAUSE' overloading, for the same then part. For example, 'IF A(MOD) and B(MOD) THEN C(MOD)' is more specific than 'IF A(MOD) THEN C(MOD)'.

***Strategy** 7. If MOD maintenance work typed requirement are input, then case retrieval proceass try the latest MOD typed case.*

If up to now processing description of requirement input and matching process was called by requirement processing, next processing which called in process flow explanation as maintainable step recommendation can be called step processing.

To the examples of earlier part of this paper, we can refer another case in the series of maintenance events as a example for inference.

{{ KEPCO(IS)193.09-986-1
IS-A : Requirement Name for Maintenance
PART-OF : KEPCO(IS)193.09-986
UNIT_OPERATION_NAME : Charge Control.
REQUIREMENT_UNIT :
(IF-CLAUSE, MOD, (450 hour over usage quantity per month))
(IF-CLAUSE, MOD, (Contract category))
(IF-CLAUSE, NEW, (Control code))
(ASSIGNMENT, MOD, (Over usage quantity charge))
REQUIREMENT_CONTENTS : " When apply 450 hour over usage charge per month, make new control code for exception, 'EV', from collecting a charge for the next electric power contract category such as (General, Industry, Temporary (B)) "
}}

For the above requirement input, we can apply inference strategiy 5 and 9. As a result of partial matching process with searching the branch of case memory, IF-CLAUSE & IF-CLAUSE & ASSIGNMENT, we can find two most related cases. That two cases are followed without related result part such as KEPCO(IS) 190.05-2754-1 and KEPCO(IS)193.09-292-3.

From these cases, related result modules of that cases are derived but it is not need to modify the results because all elemental requirements are identical except for NEW typed unit requirement with keyword, control code.

We can here compare the performance with the full scaled module search inference and keyword based search utility in IBM machine for the above example case. In table 4-1, result number is the searched step number.

	Module search	Keyword search	Case based search
# of search module	Full scale(8/17)	Full scale(8/17)	Case memory
# of recalled module	3	8	3
# of searched step	23,476	295	569
# of precise step	12	295 (Not Correct)	12
Search time	6 minute		

Table 4-1. Comparision of performance with other systems.

Another evaluation result table, table 4-2, is the applicable inference strategy number to the various condition of requirement input. In the table 4-2, deep shaded area in the region which is not applicable area of inference strategy. Full-scaled module search inference must be applied to this area. Shaded area with 'P' is the perfect matching applicable area.

Primitive requirement input type	Requirement Input Condition						
	only THEN part type	Simple IF-THEN type	Duplicated IF part type	Complex type	IF (NEW) THEN	IF-THEN (NEW)	IF-THEN other type (same keyword)
only THEN part type	P	5	5	2,4,5	3,5	■	■
Simple IF-THEN type	5,6	P	1	2,4, 5,6	3,5	3-1	■
Duplicated IF part type	5,6	6	P	2,4,6	3,5,6	3-1,6	6
Complex type	4,5,6	4,6	4	P	3,4	3-1,4	4,5,6

Table 4-2. Result of Applying inference strategies to various requirement input condition.

4.3. Case Memory

We consider here the case organizing method in memory, so to say our cases will be indexed at the time of case storage. It is for more efficient retrieval. However the method will be, the organization of cases is important consideration in building a case based reasoning system. The method of case representation is too. It influence the efficiency and capacity of the system.

We found followed characteristics for case organization. Each lited element can be the dichotomising point in their order to index cases maintenaned.

- Unit operation name - classification by operation like daily processing, charge control, movement control.
- Requirement structure - with rule structure and formula structure.
- Structure classification - global classification with elemental requirement like IF & ASSIGNMENT, IF & I/O, IF & DB-INTERFACE, ASSIGNMENT, I/O , DB-INTEFACE, and so on.
- Case structure with elemental requirement.
- Maintenance work type - with NEW, MOD, DEL. It always be the right upper index to the case storage in case memory.

Above all upper written characteristics are the meta knowlwedge for case organization and case based reasoning. Case memory organization is compared to flat memory construction. So to say upper written five characteristics are the dimensions for goal seeking of related case retrieval. All cases are stored in this case memory structure by the dimension. If there are requests for case retrieval with the dimensions later, then we match keywords within their unit requirement expression and retrive related cases. Among the retrieved cases, latest case is the most suitable case for the request of case retrieval.

Next figure is the depiction for case memory organization with example document 'KEPCO(IS)

190.05-2754'.

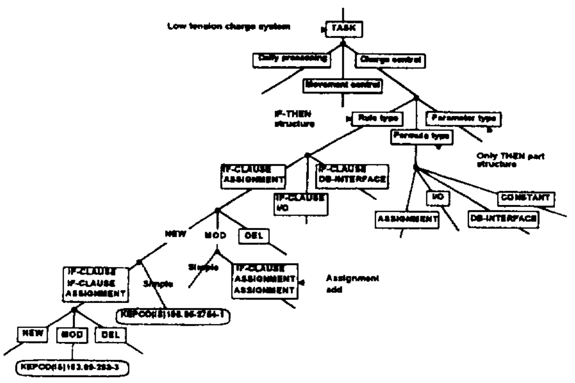

Figure 4-2. Case memory construction.

4.4. Post Processing

If upper case retrieval processes didn't find any cases related to the requirement input, then module search inference must be executed with the same requirement input. If not, the recommendation results are further filtered with full-scaled module search inference to specified procedure level and then delivered to user. The user manipulates the recommended COBOL code to maintain with the requirement input. Those manipulated results are stored as case with system aid like indexing rule.

In this process we consider automatic generation of related step information by using *automatic frame generator*. Automatic frame generator makes the program knowledge consisted with step representation of six step types, IF-CLAUSE, ASSIGNMENT, I/O, DB-INTERFACE, CALL, LOOP using original compiled COBOL code. Those step representations also use the step work type, NEW, MOD, DEL. Automatic frame generator is a kind of program parser. The parsing of program to step frame used the COBOL reserved words.

Step representations in above paragraph comprise the process of maintaining consistency of case base. Above all it is important for historical series of maintenance event. Maintaining consistency of case representation in case base can be carried out with result step representation. The result step representation is followed code manipulation with recommenation. Full result step information is attached to the related result module in case representation.

Let's see this process with KEPCO(IS)190.05-2754. A example is a part of KEPCO(IS)190.05-2754-CAM30JOJ as followed.

(CAM30JOJ-3321-FARE-COMPUTE-5-1, CALL, NEW, (CAJ94101), (CADANLD, CAOVERFLD))

(CAM30JOJ-3321-FARE-COMPUTE-5-2, CALL, MOD, (CAJ92201), (CADANLD))

(CAM30JOJ-3321-FARE-COMPUTE-5-2, CALL, DEL, (CAJ91601), (CADANLD))

Remarkable condition or constraint in above representation is, *result representation in the related result moulde frame must be written with only modified parts of the module then or new created parts on new requirement.* From this remark, if requirement input is entered with the same scope as previous creation case, then not only the past creation case but also later modified cases on that requirement input must be considered.

5. S/W Maintenance Supporting ES

Cased based supporting expert system is embedded to the module based full scale search expert system of KEPCO software maintenance supporting systems project (SWMES project) with UNiK-FRAME and C language.

Our SWMES was implemented thorough the environment of UNiK (UNified Knowledge). It constructed with 5 knowledge base on UNiK-FRAME and 2 inference machine, case based inference engine and module search inference engine.

- **Program-knowledge Base** - It is derived from KEPCO system's compiled COBOL codes through the automatic frame generator. Source codes are like RUN JCL code, PSB Gen code, DBD Gen code, MFS code, COPY IO(DB, file), table and screen inluding compiled codes.
- **Rule Base** - It is acquired from company regulation and so on.
- **Keyword Base** - It is extracted from various text like rule, regulation, convention and so on. Each keyword is connected to one or more variables of COBOL program.
- **Data Base**
- **Case Base**

Addition to theses knowledge bases, there is another one knowledge base.

- **S/W source code Base** which contain the compiled IMS COBOL source code which run in IBM hardware environment. It is the input data for automatic frame generator.

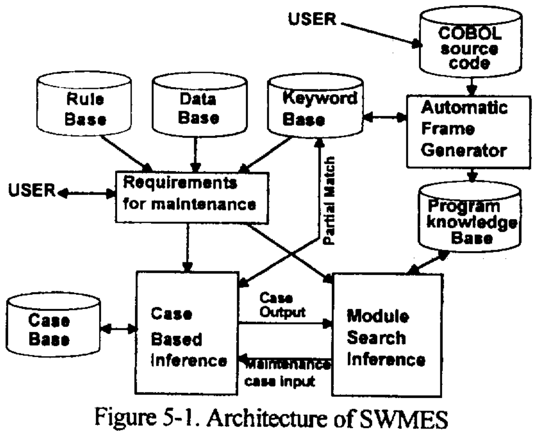

Figure 5-1. Architecture of SWMES

Our case based S/W maintenance supporting expert system is composed with the variations of the following CBR concepts written in UNiK and C language.

- Discrimination network [Hammond, 1989] for case memory construction.
- Parallel representation [Hinrichs, Kolodner, 1992] for finding correspondences.
- Abstraction hierarchy [Kolodner, 1993] for similarity between keywords.

Module search inference machine as one inference machine in SWMES except for case based inference engine find matching steps by full scaled search on COBOL program knowledge base with interaction of keyword base. It make requirement input with the similar process like in case based inference.

Automatic frame generator is a kind of program parser which extracted step frame consisted with six step type and its keywords, variables and parameters from the compiled COBOL code. For the purpose of easy comprehension of step framed information, SWMES provides this automatically generated step frames of one program with the step diagram of GUI(Graphic User Interface).

6. *Conclusion and Future Works*

We analyze and represent the case of software maintenance in COBOL domain. We organize the cases in memory construction and develop case based software maintenance supporting expert system.

The major contributions of this research can be summarized as follows. For the purpose of efficient result generation with past maintenance case,

1. Suggest methodology for case representation and organization of case memory on software maintenance hardly depended on keywords using requirement types, framework of IS and others.
2. Design and develop a expert system which has the following characteristics.

- Diminish search space compared to only keyword based system and module search inference system in SWMES.
- Parallel representation for easy finding correspondences between input case and stored cases.
- Organize case memory for efficient case retrieval to the requirement input.
- Generate the maintainable step results.

Followings are further research direction on this research.

1) Extend language domain addition to the current COBOL domain.

If we extend the domain language from which is used in the automatic frame generator to the applicable language, it will be very useful to support the software maintenance work on that language domain. It can be a suggestion for another or more powerful automatic frame generator on another language domain.

2) Extend the case's capability to program knowledge whcih has the capability of direct reuse case's contents.

It can be the hybrid system of CBR and S/W reuse concept. If this is possible, It can be powerful system which has the some capability of automatic programming.

ACKNOWLEDGEMENT

SWMES project was supported by KEPCO. Thanks to the business manager of KEPCO, Sang-Kyu Chun, for this paper's real world configuration data and information providing.

References

1. Bonnie W. Morris, *"SCAN: A Case-Based Reasoning Model for Generating Information System Control Recommendation"*, Intelligent Systems in Accounting, Finance and Management, Vol 3, 47-63, 1994.
2. D.J.Robinson, K.H.Bennett, B.J.Cornelius, and M.Munro, *"Approaches to program Comprehension"*, Journal of systems and software, Feb, 79-84, 1991.
3. D.Ourston, *"Program Recognition"*, IEEE Expert, Winter, 36-49, 1989.
4. E. Youdon, *"Reengineering, Restructuring, and Reverse Engineering."* Amercan Programmer, Apr. 1989, 14-20
5. Gilles.Fouque and Stan Matwin, *"A case based approach to software reuse"*, Journal of Intelligent Information Systems, Vol 1, 165-197, 1993.
6. Golding,A.R. and Rosenbloom,P.S., *"Improving Rule Based Systems Through Case Based Reasoning."*, AAAI-91 Case Based Reasoning, 1991
7. Jae-Kyu Lee and Min-Yong Kim, *"Case-Based Learning for Knowledge-Based Optimization Modeling System: UNIK-CASE"*, Expert System with Application, Vol. 6, 1993.
8. Janet L. Kolodner. *"An Introduction to Case-Based Reasoning"*, Artificial Intelligence Review 6, 3-34, 1992.
9. Janet L. Kolodner. *"Extending Problem Solver Capabilities Through Case-Based Interface"*, Proceeding of Machine learning Workshop, 167-178,1987.
10. Janet L. Kolodner. *"Improving Human Decision Making through Case-Based Decision Aiding"*, AI Magazine, 52-68, summer, 1991.
11. Janet Kolodner. *"Case-Based Reasoning"*, Morgan Kaufmannn Publishers, Inc. 1993.
12. Juan E.Vagars and Savita Raj *"Developement maintainable expert systems using case base reasoning"*, Expert Systems, Vol 10, No 4, November 1993

13.Katia P. Sycara and D. Navinchandra *"Integrating Case Based Reasoning and Qualitative Reasoning in Engineering Design"*, AI in Design J.S Gero(ed) Springer-verlag, NewYork, 1989

14.KEPCO, *"A study on the efficient computer system management - Module Bank"* KEPCO, 1989.

15.Kristian J. Hammond. *"Case-based Planning: Viewing Planning as a Memory Task. Perspective in Artificial Intelligence"*, Academic Press,Boston, MA. 1989

16.Kristian J. Hammond. *"CHEF: A Model of Case-based Planning"*, AAAI, 267-271, 1986

17.Kuipers, B.J. *"Qualitative simulation"*, Artificial Intelligence, Vol 29, 1986

18.M.F.Dunn and J.C.Knight, *"Software Reuse in an Industrial Setting: A Case Study"*, Proc. 13th Int'l Conf. on Software Engineering, 1991, 329-338

19.M.Ramesh and H.Raghav Rao, *"Software reuse: Issues and an example"*, Decision Support Systems, Vol 12, 57-77, 1994.

20.Mark Kriegsman, Ralph Barletta *"Building a Case Based Help Desk Application"*, IEEE Expert 1993,12.

21.M.T.Harandi and J.Q.Ning, *"Knowledge-Based Program Analysis"*, IEEE Software, Jan. 1990, 74-81

22.Navinchadra, D., *"Case Based Reasoning in CYCLOPS, a design problem solver"*, Proceedings of the DARPA Workshop on Case based Reasoning, May 10-13 1988, 286-301

23.Paul Harmon. *"Case Based Reasoning I"*, Intelligent Software strategies Vol. 7, No 11, 1991 NOV.

24.Paul Harmon. *"Case Based Reasoning II"*, Intelligent Software strategies Vol. 7, No 12, 1991 DEC.

25.R.Arnold and W.Frakes, *"Software Reuse and Reengineering"*, CASE Trends, Feb, 1992.

26.Robert S.Arnold, *"Software Reengineering"*, IEEE computer society press, 1993.

27.Rissland,E.L. and Sakalak,D. *"Combining case based and rule based reasoning : A heuristic approach"*, Proceeding of IJCAI-89 San Mateo, CA:Morgan Kaufmann.

28.Robert S.Williams, *"Learning to program by examing and modifying cases"*, Machine learning, 318-324, 1988.

29.Robert T.Chi, Minder Chen, Melody Y, Kiang *"Generalized Case Based Reasoning System for Portfolio Management"*, Expert Systems with applications, Vol.6, 67-76, 1993.

30.Schank,R. and C, Risebeck. *"Inside Case-Based Reasoning"*. Lawrence Erlbaum Assoc., Hillslad, NJ, 1989.

31.Simoudis,E., and Miller,J.S. *"Validated retrieval in case based reasoning"*, Proceedings of AAAI-90. Cambridge, MA:AAAI press/MIT press.1990

32.Stephen Slade. *"Case-Based Reasoning: A Research Paradigm"*, AI Magazine, 42-55, spring, 1991.

33.Sycara K. and Navinchadra, D., *"Index transformation and generation for case retrieval"*, Proceeding of the Case Based Reasoning Workshop, Pensacola Beach, Fla., May 1989, 324-328.

34.Thomas J. Schult and Dietmar Janetzko *"Case Based Expert System Shells : First and Second generation"*, proceeding on WCES '94, 1994

35.T.J.Biggerstaff, "Design Recovery for Maintenance and Reuse", IEEE Computer, Jul. 1989, 36-49