COBOL 환경 하의 정보 시스템에 대한 사례 기반 유지 보수 지원 방법론

김우주* · 이재원** · 이재규***

A Case Based Maintenance Support for Information Systems in COBOL Domain

Wooju Kim*, Jae W. Lee**, Jae K. Lee***

요 약

정보 시스템에 대한 유지 보수 문제는 소프트웨어 유지 보수 분야에서 지속적으로 존재하는 문제 중의 하나이며, 특히 이러한 정보 시스템들이 아주 오래 전에 COBOL이나 계층형 데이터베이스와 같은 기술을 바탕으로 개발되었음에도 불구하고 대부분의 조직에서 중요한 역할을 담당하고 있는 것이 현실이다. 따라서 이들 정보 시스템에 대한 효율적이고, 신속한 유지 보수는 조직에 있어 매우 중요한 업무임에도 불구하고, 대부분의 조직에서 이를 제대로 수행하고 있지 못하다. 체계적이고 효과적인 정보 시스템에 대한 유지 보수 업무 수행을 위해 본 연구에서는 과거 유지 보수 사례를 이용하여 사례 지식을 축적할 수 있는 유지 보수 사례의 표현 방법 및 유사성 평가 방법과 이를 바탕으로 새로운 유지 보수 요구에 적합한 과거 사례와 유지 보수 대상 코드를 제안해 주는 사례 기반 유지 보수 대상 인식 방법론을 제시하였다. 이들 제안된 방법들의 타당성과 성과는 한국전력의 실제 정보 시스템을 대상으로 검증되었으며, 현장 업무에 적용되고 있다.

Keyword : CBR(Case Based Reasoning), Software maintenance system

* 전북대학교 산업공학과
** 한국과학기술원 테크노경영대학원
*** 한국과학기술원 테크노경영대학원

I. 서 론

정보 시스템에 대한 유지 보수 단계가 정보 시스템 생명 주기(Life cycle)에서 가장 힘들고, 시간을 많이 소비하는 단계로서 인식되기 시작한 이래로, 유지 보수 단계를 효율적으로 수행하기 위한 많은 노력이 경주되어 왔으며, 그 결과 유지 보수 단계를 지원하기 위한 많은 시스템 개발 방법론과 CASE(Computer Aided Software Engineering) 도구들이 개발되었다(Lirov, 1991). 그러나 이들 대부분이 새로이 개발되는 시스템 상황에서의 유지 보수 문제만을 고려하고 있기 때문에 기존의 시스템의 경우에는 완전히 전체 시스템을 다시 개발하지 않고는 이들 방법론과 도구들을 적용하기 어려운 실정이며 만일 전체 시스템을 다시 개발한다 하더라도 이를 위해 대부분 감당하기에 어려운 예산과 노력을 필요로 하기 때문에 불가능한 경우가 많다.

최근의 문헌들에서 이러한 진부화된 정보 시스템(Legacy system)들에 대한 유지 보수의 어려움을 정보 시스템 문제라 부르고 있으며(Bennett 1995), 이들 시스템들 대부분은 현재 시점에서 오래 전에 COBOL이나 계층적 데이터베이스 같은 구형의 컴퓨터 언어나 기술을 이용하여 개발되었다. 한편 요즈음의 대규모 조직이나 기업들은 대부분 많은 수의 정보 시스템을 보유하고 있으며, 따라서 이들에 대한 유지 보수 문제는 이들 조직이나 기업에 있어 매우 중대한 현안으로 떠오르고 있다. 특히 이러한 정보 시스템들이 유지 보수에 어려움을 가지게 되는 공통적 이유는 다음과 같이 세 가지로 요약될 수 있다.

(1) 정보 시스템의 규모가 매우 크며, 동시에 관리해야 할 프로그램과 데이터베이스의 종류가 매우 많다. 더구나 이들 프로그램과 데이터베이스간 관계의 복잡성마저 매우 높은 실정이다.

(2) 정보 시스템에 대한 유지 보수 작업이 서로 다른 다수의 인력에 의해 오랜 기간 동안 진행되어 왔다.

(3) 정보 시스템에 대한 설명이나 설계 자료가 대부분의 경우 전무한 실정이다.

본 연구에서는 먼저 국내 유일의 전력 회사인 한국 전력에서 현재 사용하고 있는 정보 시스템인 전기 요금 관리 시스템을 우선적인 연구 대상으로 하고 있다. 이 시스템은 지금으로부터 27년 전에 개발되었으며, 시스템에 대한 주요한 유지 보수 작업이 약 400회 이상 이루어져 왔다. 또한 이 시스템은 COBOL 언어와 IMS 데이터베이스를 이용하여 개발되었으며, 이들 기술은 개발 당시에는 가장 최신의 기술이었음은 물론이다. 이 시스템 하나만 보더라도 약 900여 개의 프로그램으로 구성되어 있으며 현재 이들에 대한 설계 내역 자료는 거의 없는 실정이다. 따라서 원문 코드로부터 직접 시스템의 운영 상황을 파악하기란 매우 힘든 일임에 분명하다. 이러한 상황하에서 시스템의 유지 보수 문제를 해결하기 위해 원문 코드부터 분석하기 시작하여 새로운 시스템을 재개발한다는 것은 많은 자원을 필요로 할 뿐만 아니라 그에 따른 위험 부담 또한 매우 높은 실정이다. 한편 아직도 이 시스템은 조직에서의 자신의 역할을 충실히 수행하고 있는 실정이므로 오히려 경영층에서는 새로운 시스템을 개발하기보다는 기존의 정보 시스템에 대한 유지 보수 작업을 체계적이고 효율적으로 지원할 수 방법에 대해 더 많은 관심을 가지고 있다. 이러한 배경하에 본 연구는 정보 시스템 문제를 해결하기 위해 최신의 CASE 도구들을 이용하여 구 시스템을 재개발하는 방법(Williamson, 1991)에 대한 하나의 대안으로서 정보 시스템을 존속시키면서 동시에

이에 대한 유지 보수 업무를 신속하고 효율적으로 수행할 수 있도록 지원하는 방법을 모색하고자 한다.

우리는 먼저 정보 시스템 문제에 대한 이러한 접근 방법을 지원하기 위해 본 연구의 대상 시스템에 대한 유지 보수 업무 분석을 실시하였고 그 결과 다음과 같은 유지 보수 업무의 특성들을 파악할 수 있었다.

첫째, 대부분의 유지 보수 업무는 대규모의 파격적인 유지 보수라기보다는 조직의 내외부 상황의 소규모 변화를 시스템에 반영하기 위해 수행된다는 점이다. Lientz와 Swanson(1980)의 연구에서도 이러한 사실과 일치되는 결과를 보이고 있다.

둘째, 유지 보수 요구를 발생시키는 이러한 조직의 내외부 상황의 변화는 대부분 유사한 형태를 가지며, 매우 반복적으로 발생한다는 점이다.

셋째, 정보 시스템에 대한 유지 보수 업무 수행에 있어서의 대부분의 시간과 노력이 시스템 유지 보수 자체보다는 유지 보수 대상을 인식하는데 소비되고 있다는 점이다. Fjeldstrad와 Hamlen (1983)의 연구 결과도 이러한 사실을 뒷받침해 주고 있다.

위의 세가지 유지 보수 업무의 특성 분석을 바탕으로 본 연구에서는 유지 보수 사례를 이용한 유지 보수 대상 인식 방법론을 제안함으로써 정보 시스템 문제에 대한 가장 효과적이고 효율적인 한 대안을 제시하고자 한다. 이러한 목적을 달성하기 위해 본 연구에서는 다음의 세가지 세부 사항들을 수행하고자 한다.

(1) 유지 보수 사례의 표현 방법 개발

유지 보수 사례의 속성 설계가 각각의 유지 보수 결과의 의미를 충분히 반영할 수 있어야 하며, 각각의 서로 다른 유지 보수 결과 간의 유사성을 이들 속성들을 이용하여 측정 가능하여야 한다.

(2) 유지 보수 요구의 표현 방법 개발

유지 보수 요구의 표현 방법은 유지 보수 담당자가 사용하기 편하면서도 동시에 관련 유지 보수 요구를 기술하기에 충분한 표현력을 가져야 한다. 이와 더불어 유지 보수 사례 표현과도 호환 가능하도록 설계되어야만 한다..

(3) 사례 기반 유지 보수 대상 인식 방법론 개발

유지 보수 대상을 인식하는 일은 전체 유지 보수 업무에 있어 가장 많은 비중과 시간을 차지하고 있는 부분이다. 만일 과거 유지 보수 사례를 이용하여 정보 시스템에 대한 유지 보수 대상들을 신속하고 완벽하게 인식할 수 있다면 이는 정보 시스템 문제를 상당 부분 해결할 수 있는 하나의 방안으로 볼 수 있다. 이를 위해 유지 보수 요구에 대해 의미 있는 유지 보수 사례를 추출할 수 있는 사례 추출 방법론(Case Retrieval Method)과 추출된 유지 보수 결과를 보다 완전한 유지 보수 대상을 제시할 수 있도록 조정해 줄 수 있는 사례 기반 해 조정 방법(Case Based Solution Adaptation Method)이 마련되어야 한다.

본 연구에서는 이상에서 언급된 방법론들에 대한 이론적 설계뿐만 아니라 이 방법론들을 한국 전력공사의 실제 정보 시스템인 전기 요금 관리 시스템에 적용하여 구현해 봄으로써 그 경험적 타당성을 입증하고 있다. 이제 이하에서의 논문의 구성을 미리 소개하고자 한다. 우선 2장에서 본 연구 수행의 기반이 되었고, 또한 본 연구의 결과가 구현되어 하부 시스템으로 활동하고 있는 METASOFT 유지 보수 지원 도구(*Lee et al.*, 1997)의 구조에 대해 소개하고자 한다. 3장에서는 정보 시스템의 속성들에 대해 간단히 설명하고, 4장에서는 METASOFT에서 사용하고 있는 정보 시스템에 대한 설계 정보 표현 방법과 이에 따른 정보 시스

템 유지 보수 사례 및 유지 보수 요구 표현 방법을 설명하고자 한다. 한편 4장에서 마련된 표현 방법을 바탕으로 5장에서는 사례 기반 유지 보수 대상 인식 방법론을 제안하고 실제 예를 통해 방법론의 적용 절차를 예시하고자 한다. 이에 더불어 그 유지 보수 지원 성과에 대한 계량적 분석도 함께 보여주고 있다. 마지막으로 6장에서는 본 연구의 연구 결과가 갖는 의의와 그 파급 효과를 요약 정리하고자 한다.

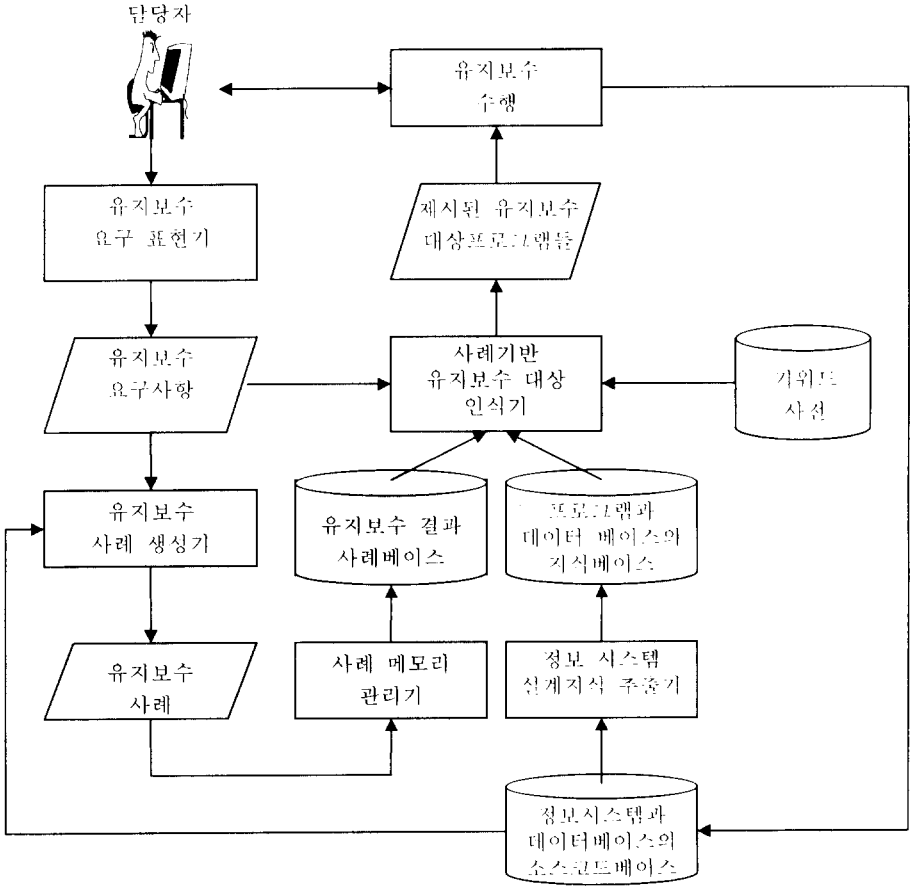

그림 1. METASOFT의 구조

II. METASOFT의 구조

본 연구에서의 정보 시스템 문제에 대한 접근 방향을 지원할 목적으로 지식 기반 접근 방법론과 함께 METASOFT라는 도구가 개발된 바 있으며, 그 경험적 타당성도 입증된 바 있다(*Lee et al.*, 1997). 한편 본 연구에서 제안하고자 하는 사례 기반 유지보수 대상 인식 방법론은 Lee등(1997)의 연구 결과를 바탕으로 METASOFT의 유지 보수 대상 인식 과정을 보다 효과적이고 신속하게 지원할 수 있도록 하기 위한 목적으로 개발되어 METASOFT의 하위 구성 시스템으로 구현되어 있다.

본 장에서는 우선 METASOFT의 기본 구성 요

소와 본 연구에서 제안하게 될 유지 보수 대상 인식 방법론이 정보 시스템에 대한 전반적인 유지 보수 업무를 어떻게 지원하게 되는지를 전체적으로 조망해보면서, 이미 구축된 METASOFT의 네 가지 기본적 구성 요소와 새로이 추가된 사례 관련 구성 요소들에 대해 설명하고자 한다. 다음의 (그림 1)은 METASOFT의 전체 구조를 보여주고 있는데 여기서 검게 처리된 부분이 본 연구에서 METASOFT에 새롭게 추가된 부분임을 나타내고 있다.

(그림 1)에서 보여지고 있는 META-SOFT에서의 유지 보수 과정은 우선 정보 시스템 설계 지식 추출기(Documentary Knowledge Extractor)가 정보 시스템 원문 코드로부터 프로그램들과 데이터베이스에 대한 설계 지식을 역공학(Reverse engineering)을 통하여 추출하여 지식베이스 를 구성하게 된다. 한편 유지 보수 요구 표현기는 유지 보수 담당자로부터 유지 보수 요구를 입력받게 되며, 사례 기반 유지 보수 대상 인식기(Case Base Maintenance Point Identifier)는 유지 보수 사례 베이스와 정보 시스템 설계 지식베이스를 이용한 일련의 사례 기반 추론 과정을 통하여 유지 보수 요구에 대한 보수 대상 프로그램과 데이터 항목을 유지 보수 담당자에게 제공하게 된다. 제공된 프로그램과 데이터 항목들을 바탕으로 유지 보수 담당자는 유지 보수 요구에 따른 유지 보수 작업을 수행하게 되는데 이 때 프로그램 원문과 데이터 항목들에 대한 보수 결과는 정보 시스템 설계 지식 추출기에 의해 다시 지식베이스에 자동 반영되게 된다. 이와 함께 유지 보수 요구와 그 보수 작업 결과는 유지 보수 사례 생성기(Maintenance Case Generator)를 통하여 하나의 유지 보수 사례로서 표현되며, 이 정보는 다시 최종적으로 사례 기억 관리자(Case Memory Organizer)에 의해 사례 베이스에 저장된다. 이상의 과정을 통하여 METASOFT는 하나의 유지 보수 업무를 처리하고 미래에 오게 될 새로운 유지 보수 요구에 대해 대응할 준비를 갖추게 된다.

III. 예제 정보 시스템

(그림 2)는 COBOL로 작성된 CAM30JOJ라는 한 프로그램의 일부를 보여주고 있다. PROCEDURE DIVISION에서 하나의 프로그램은 SECTION들(e.g. 0000-START-PROCESS, 2000-GET, and 3000-KWAN-CHECK-RTN)로 구성되며, 각 SECTION들은 다시 일단의 명령문들로 구성된다.

각 SECTION의 명령문들을 수행시키기 위해서는 'PERFORM'이라는 명령어를 이용하여 해당 SECTION의 이름을 호출하게 되며, 각 명령문들은 보통 단일 명령문으로서 이루어지나, IF문과 같은 복합 명령문을 사용할 경우도 있다.

본 연구에서는 정보 시스템에 대한 유지 보수 지원의 수준을 높이기 위해 (그림 2)와 같은 원문의 각 명령문 단위까지 유지 보수 대상을 인식하고자 한다. 한편 유지 보수 요구 자체는 이러한 COBOL 명령문 형식으로 기술될 수 없기 때문에 이들 상호 간의 연계성이 확보되지 않고는 유지 보수 요구를 이용하여 단위 명령문 수준의 유지 보수 대상 인식이 불가능하다. 따라서 우리는 이들 단위 명령문들이 상위 수준에서 표현되는 유지 보수 요구에 연계될 수 있도록 단위 명령문의 속성을 분석하여 8가지 범주를 갖는 속성 분류 체계를 수립하였다. <표 1>은 이러한 8가지 단위 명령문의 속성 분류와 이에 해당되는 COBOL의 예약어들을 보여주고 있다.

```
                .....
018800
018900 PROCEDURE DIVISION.
019000     0000-START-PROCESS SECTION.
019300                        OPEN I-O   MAST-F.
019500                        PERFORM 1000-DATE-READ.
019600                        PERFORM 2000-GET.
019800                        PERFORM 3000-KWAN-CHECK-RTN
                .....
020400                        STOP RUN.
                .....
021000     2000-GET SECTION.
                .....
021800                        IF CA00-MST-CD = SPACE
021900                           COMPUTE CAJOJ-ED-KWH-FARE = CA-USAGE * 10
022000                           ADD 1 TO MST-CNT.
                .....
037800     3000-KWAN-CHECK-RTN SECTION.
                .....
```

그림 2. CAM30JOJ 프로그램의 원문 코드 예

표 1. 명령문의 속성에 따른 예약어들의 분류 체계

원문절의 범주	COBOL 예약어
SECTION-NAME	SECTION
ASSIGNMENT	MOVE, SET, ADD, DIVIDE, SUBTRACT, MULTIPLY, COMPUTE
IF	IF, WHEN, SEARCH
LOOP	PERFORM-UNTIL-BY
I/O	ENTRY (All options excepting DB-INTERFACE's options), ACCEPT, DISPLAY
DB-INTERFACE	ENTRY (Option : GU, GN, GHN, GHU, GNP, GHNP, ISRT, DLET, REPL, CHNG, PURG, GHUP, GUP)
FILE-INTERFACE	OPEN, READ, WRITE, REWRITE, CLOSE
CALL	CALL

정보 시스템의 또 하나의 요소인 IMS 데이터베이스의 원문이 (그림 3)에서 보여지고 있는데, 현재 예시된 데이터 베이스인 CAPMLOW1은 개체(Entity)에 해당되는 일단의 세그먼트(Segment)

들로 구성되며, 각각의 세그먼트들은 'SEGM' 이라는 예약어에 이어 세그먼트 명(e.g. CA00ROOT)과 속성(Attribute)에 해당되는 필드 명들을 정의하게 된다.

이러한 프로그램과 데이터베이스 원문에 존재하는 변수 명과 필드 명들은 각 프로그램과 데이터베이스를 부분적으로 설명할 수 있는 잠재적 키워드로서 인식하고 본 연구에서 명령문들에 대한 속성 분류와 함께 유지 보수 대상 인식 과정에서 사용되게 된다.

```
.....
          DBD   NAME=CAPMLOW1,ACCESS=(HIDAM,VSAM)
          DATASET DD1=CAPMLOW1,DEVICE=3390,SIZE=7680,FRSPC=(20,10)
*ROOT
SEGM      NAME=CA00ROOT,PTR=TB,BYTES=160,PARENT=0,                C
               COMPRTN=(DPIEXIT,DATA,INIT)
          FIELD NAME=(CA0KROOT,SEQ,U),BYTES=19,START=1,TYPE=C
          LCHILD NAME=(CA0IROOT,CAINLOW1),PTR=INDX
          FIELD   NAME=CA0FDONG,BYTES=10,START=28,TYPE=C
          .....
SEGM      NAME=CA01YOKM,PTR=TB,BYTES=90,                          C
               PARENT=((CA00ROOT,DBLE)),                          C
               COMPRTN=(DPIEXIT,DATA,INIT)
          FIELD   NAME=(CA0KYOKM,SEQ,U),BYTES=4,START=1,TYPE=C
.....
```

그림 3. CAPMLOW1 데이터베이스의 원문 코드 예

Ⅳ. 유지 보수 요구와 사례의 표현 방법

본 장에서는 사례 기반 유지 보수 대상 인식 방법론 개발의 기초가 되는 유지 보수 요구와 과거 유지 보수 사례에 대한 표현 방법을 제안하고자 한다. 유지 보수 사례 표현은 일반적인 사례 표현 구성 방식에 따라 크게 문제 기술 영역(Problem description part)과 해법 기술 영역(Solution description part)의 두 부분으로 나뉘게 되는데, 먼저 유지 보수 사례 표현에 있어서의 문제 기술 영역은 곧 유지 보수 요구 내역을 의미하게 되며, 유지 보수 담당자에 제공될 유지 보수 요구 표현 방법과 같은 수준에서 호환성이 확보되어야 한다. 따라서 본 연구에서는 유지 보수 요구 표현 방법과 유지 보수 사례를 구성하는 한 부분인 문제 기술 영역의 표현을 동일시하고자 한다. 한편 사례 표현의 나머지 부분인 해법 기술 영역은 해당 유지 보수 요구에 대한 유지 보수 결과를 의미하며 구체적으로는 유지 보수되었던 특정 프로그램이나 데이터베이스의 원문 코드 집합을 표현하여야 한다. 프로그램이나 데이터베이스의 일부분을 표현하기 위한 쉽게 접근할 수 있는 한 방법은 이들 부분을 직접 복사하여 원문 형태로 저장하는 방법이다. 그러나 이러한 방식은 다음의 두 가지 이유로 인해 유지 보수 지원에 있어 그 효과를 최소화시키고 있다.

첫째, 단순히 여러 원문들로부터의 부분적 발췌 정보는 해당 유지 보수 사례를 이해하는데 있어 최소한의 도움밖에는 줄 수 없다. 다시 말해서 각 원문 부분들간의 상호 관계와 흐름을 이해하기 위해서는 다시 전체 정보 시스템의 관점에서 각 원문 부분들의 상대적 위치와 논리적 분석을 필요로 하게 된다.

둘째, 궁극적으로 유지 보수 대상 인식을 위해서는 원문의 부분들의 복사 정보를 이용하여 현재 정보 시스템의 원문은 추적하여야 하는데 이를 위해 곧 원문 부분에 대한 복사 정보가 현재의 원문과 임의의 연결 관계를 유지해야만 한다. 그러나 원문 부분 복사 정보로부터 추출할 수 있는 특정 프로그램 명과 그 안에서의 절대적 위치 정보(e.g. Line Number)만으로는 이러한 추적이 불가능하다. 이는 유지 보수 결과 표현이 정보 시스템에 대한 설계 수준의 정보와 이에 대한 논리적 연계 관계를 고려할 수 있음으로써 가능해 진다.

본 연구에서는 METASOFT의 정보 시스템 설계 지식 표현 방법을 기초로 하여 유지 보수 사례 결과 표현 방법을 설계함으로써 이상에서의 원문 복사 정보를 이용한 방법의 문제점들을 해결하고 있다. 따라서 다음 절에서는 유지 보수 요구 표현이나 사례 표현 방법을 제안하기에 앞서 우선 METASOFT의 정보 시스템 설계 지식 표현 방법에 대해 간략히 소개하고자 한다.

4.1 METASOFT에서의 정보 시스템 설계 정보 표현

METASOFT에서는 정보 시스템의 프로그램과 데이터베이스를 표현하기 위해 객체 지향적 표현 방법을 채택하고 있다. 여기서 객체가 의미하는 바는 프레임(Lee and Song, 1994; Minsky, 1985; Sull and Kashyap, 1992)과 동일한 개념을 가리킨다. 유지 보수 지원을 위한 정보 시스템에 대한 설계 정보 표현은 사용자의 유지 보수 요구와 연계될 수 있는 추상성(Abstraction Level)과 동시에 유지 보수 대상 인식을 위한 구체성도 가져야 됨은 물론이고 시스템에 대한 설계 자료가 거의 없는 상태에서 원문으로부터 역공학 기법(Aiken *et al.*, 1994; Biggerstaff, 1989; Burson *et al.*, 1990; Housier and Pieszkoch, 1990; Kozaczynski *et al.*, 1991; Markosian *et al.*, 1994; Ning *et al.*, 1994; Premerlani and Blaha, 1994)을 이용하여 추출 가능하여야 한다. METASOFT는 COBOL 프로그램과 IMS 데이터베이스에 대해 이러한 특성을 만족하는 시스템 설계 지식베이스를 자동 구축할 수 있도록 구현되어 있다.

4.1.1 정보 프로그램 및 데이터베이스 설계 지식 표현 방법

METASOFT에서는 COBOL 프로그램의 프로그램 단위, 섹션(Section), 명령문 등에 대한 시스템 설계 지식들을 Lee등(1997)의 연구 결과에 의거하여 표현하고 있다. 여기에서는 자세한 시스템 설계 지식 표현 양식에 대한 설명은 생략하고 (그림 2)의 프로그램 원문에 대한 몇 가지 설계 지식 표현 예를 이용하여 그 설명을 대신하고자 한다.

다음은 (그림 2)의 COBOL 프로그램으로부터 자동 추출된 설계 지식에 대한 객체 지향적 정보 표현 예이다. 아래에서 프로그램 CAM30JOJ는 'CAM30JOJ'라는 이름을 갖는 객체를 이용하여 표현하고 있으며 이와 함께 CAM30JOJ-2000-GET-021800, CAM30JOJ-2000-GET-021900 객체 등에 의한 명령절에 대한 정보 표현 예도 보여주고 있다. 특히 여기서 각 프로그램 설계 지식 객체의 이름들은 METASOFT에 의해 프로그램

명, 섹션 명, 라인 번호를 합성하여 자동 생성되고 있다.

```
{{ CAM30JOJ
    IS-A :                  PROGRAM
    RELATED-SYSTEM :        CHARGING-SYSTEM
    TITLE :                 Charging Main Program
    LANGUAGE :              COBOL
    TYPE :                  BATCH
    MAIN-SUB :              MAIN
    VARIABLES :             CA00-MST-CD CAJOJ-ED-KWH-FARE
                            CA-USAGE MST-CNT
    INPUT-DB :              CAPMLOW1
    OUTPUT-DB :             CAPMLOW1
    INPUT-FILE :            MAST-F DATE-F
    OUTPUT-FILE :           DATE-F MAST-F
    CALL :                  CAJ92201 CAJ94101 CAS30EXT
    FIRST-SECTION-NAME :    CAM30JOJ_0000-START-PROCESS  }}

{{ CAM30JOJ_2000-GET_021800
    IS-A :                  IF
    IN-SECTION :            CAM30JOJ_2000-GET
    CONDITIONAL-VARIABLES : CA00-MST-CD
    THEN-CLAUSES :          CAM30JOJ_2000-GET_021900
                            CAM30JOJ_2000-GET_022000
    ELSE-CLAUSES :                                        }}

{{ CAM30JOJ_2000-GET_021900
    IS-A :                  ASSIGNMENT
    IN-SECTION :            CAM30JOJ_2000-GET
    LHS-VARIABLE :          CAJOJ-ED-KWH-FARE
    RHS-VARIABLES :         CA-USAGE                      }}
```

한편 METASOFT에서는 IMS 데이터베이스에 대한 설계 지식 역시 프로그램 설계 지식 표현 방법과 유사한 방법을 이용하여 표현하고 있으며 여기에서는 이에 대한 상세한 내용은 생략하고자 한다(보다 자세한 내용은 Lee등(1997)의 연구 결과에서 참조 할 수 있다).

4.1.2 키워드와 용어 사전(Key Words Dictionary)

METASOFT에서 사용하고 있는 '키워드'라는 용어는 프로그램에 존재하는 변수나 데이터베이스의 객체와 속성에 대한 사용자 관점에서 이해 가능한 용어들을 의미한다. 데이터베이스의 속성이나 객체들에 대한 키워드들은 직접 데이터베이스 설명 파일로부터 자동 추출이 가능한 경우가 있기도 하지만, 대부분의 정보 시스템의 프로그램이나 데이터베이스 원문에 존재하는 변수나 객체의 의미를 자동적으로 파악하여 키워드를 할당하는 것은 거의 불가능하다. 따라서 이러한 문제를 해결하기 위한 METASOFT의 접근 방법은 먼저 해당 기업의 규정집(Moulin and Rousseau, 1992), 업무 지침, 각종 보고서, 입출력 화면 설계 자료 등으로부터 잠재적 키워드들을 조사하여 용어 사전을 구축하고 이를 바탕으로 역공학을 이용한 프로그램 및 데이터베이스 원문 분석을 통하여 유지 보수 담당자로 하여금 최소한의 해당 변수들을 앞에서 준비된 용어 사전의 특정 키워드에 연결시킬 수 있도록 지원함으로써 최종적인 용어 사전을 완성하고 있다. 이

렇게 얻어진 키워드들은 특정 유지 보수 요구나 유지 보수 사례 표현에서도 표준적 용어로서 사용되게 되며, 이는 유지 보수 요구 및 사례와 원문간의 상호 이해의 출발점이 되고 있다.

특히 키워드 표현 정보는 규정 집에서의 정의나 프로그램과 데이터베이스에서의 관계 분석을 통해 키워드들간의 동의어 관계는 물론이고 상하위 관계 정보까지 함께 포함하게 된다. 아래의 키워드 'Monthly-Fare'에 대한 표현 예는 이러한 관계 정보가 어떻게 표현될 수 있는지를 보여 주고 있다. 먼저 아래의 예는 키워드 '*Monthly-Fare*'가 부모 키워드로서 '*Fare*'를 가지고 있으며 동시에 자식 동의어로서 '*Monthly-Over-Fare*'를 가지고 있다는 것을 말하고 있고 CAPMLOW1라는 데이터베이스의 CA00ROOT 객체 속성인 CA00-ED-KWH-FARE와 연계되고 있다. 한편 프로그램과 관련해서는 CAM30JOJ 프로그램의 지역 변수인 CAJOJ-ED-KWH-FARE와도 연결되어 있는 상태이다((그림 2)의 라인 번호 21900 참조).

{{ **Monthly-Fare**
IS-A : KEYWORD
EXISTING-DATABASE : (CAPMLOW1 CA00ROOT CA00-ED-KWH-FARE)
EXISTING-PROGRAM : (CAM30JOJ CAJOJ-ED-KWH-FARE)
PARENT : Fare
CHILD : Monthly-Over-Fare
SYNONYM : }}

4.2 유지 보수 요구의 표현

이제 우리는 METASOFT의 정보 시스템 설계 지식 표현 방법을 바탕으로 유지 보수 사례의 문제 기술 영역에 대한 표현 방법으로도 사용될 유지 보수 요구 표현 방법을 제안하고자 한다. 유지 보수 요구 표현 방법은 먼저 유지 보수 담당자가 사용하기에 충분히 간편하여야 하며 동시에 유지 보수 대상 인식을 위해 정보 시스템 설계 지식과도 일관성을 유지하여야 한다. 이러한 표현 방법에 대한 요건들은 곧 유지 보수 요구 표현에 있어서 앞 절에서 소개한 키워드와 단위 명령 절의 속성 정보가 기본적으로 포함되어야 한다는 것을 의미한다.

그러나 3절에서의 COBOL 프로그램의 각 명령 절들에 대한 여덟 가지 속성 분류 체계하에서는 유지 보수 담당자가 유지 보수 요구를 표현할 때 미리 그 속성을 정확히 분류하기 매우 어렵기 때문에 이를 위해 여덟 가지 분류 체계를 유지 보수 담당자의 입장에서 네 가지로 축소 조정하였다. 또한 이에 덧붙여 데이터베이스 자체에 대한 유지 보수 요구에 대한 표현 편의를 위해 'DATABASE' 분류를 추가하였다. 따라서 유지 보수 요구 표현을 위한 명령 절 분류 속성들은 최종적으로 다음의 다섯 가지로 제시되었다.

(1) ASSIGNMENT
(2) CONDITIONAL : <표 1>의 IF와 LOOP 속성들을 포함한다.
(3) DATA-INTERFACE : <표 1>의 DB-INTERFACE와 FILE-INTERFACE를 포함한다.
(4) I/O
(5) DATABASE : 데이터베이스 자체 조정을 위해 추가되었다.

유지 보수 요구 표현 방법이 갖추어야 하는 또

다른 요건은 유지 보수 요구 사항의 실질적 내용을 충분히 기술할 수 있어야 한다는 점이며 키워드와 명령 절의 속성 분류 외에도 유지 보수 요구의 성격을 묘사하는 중요한 특징 중의 하나는 유지 보수 요구서(Maintenance Requirement Statement)에서 요구 사항을 설명하는 문장 내의 각각의 절들이 새로운 내용의 추가를 의미하는지 아니면 기존의 내용에 대한 보완이나 삭제 또는 참조를 의미하는지에 따른 구분이다. 이에 따라 우리는 이를 유지 보수 요구 표현에 반영하기 위하여 유지 보수 유형(Maintenance Mode)이라는 특성을 정의하고, 유지 보수 담당자가 각각의 관련 절들에 대한 성격 기술에 추가(Addition), 삭제(Deletion), 보완(Modification), 참조(Reference)의 분류 체계를 사용할 수 있도록 하였다.

이상에서의 요인들을 고려하면서 우리는 (그림 4)과 같이 유지 보수 요구 표현 방법을 고안하였다. (그림 4)에서 볼 수 있듯이 하나의 유지 보수 업무(Maintenance-Task)는 여러 개의 유지 보수 요구(Requirement) 단위들을 가질 수 있으며 다시 이들 각각의 유지 보수 요구는 하나 또는 복수의 유지 보수 요구 요소(Requirement-Component)들로 구성되게 되나 이때는 유지 보수 업무 표현과 달리 복수의 유지 보수 요구 요소들로 구성될 경우 이들의 순서는 각 요구 요소들의 수행 순서를 의미하게 된다. 한편 유지 보수 요구 요소는 단순 요소(Simple-Component) 양식이나 복합 요소(Complex-Component) 양식 중 선택적으로 표현 가능하며 복합 요소의 경우 **IF, AND, OR, THEN, ELSE** 등의 논리적 연산자들을 이용하여 논리 표현도 가능하게 된다. 마지막으로 최하위의 유지 보수 요구 요소는 앞에서 정의된 바 있는 유지 보수 요구 속성, 유지 보수 요구 유형 및 관련 키워드들을 이용하여 표현되고 있다.

```
<Maintenance-Task>          ::= (<Requirement>) {(<Requirement>)}
<Requirement>               ::= <Requirement-Component> {<Requirement-Component>}
<Requirement-Component>     ::= <Simple-Component> | <Complex-Component>
<Simple-Component>          ::= REQ(<Maintenance-Mode>, <Maintenance-Property>,
                                <Keyword> {<Keyword>})
<Complex-Component>         ::= ((IF <Premise-Part>) (THEN <Requirement))
                                | ((IF <Premise-Part>) (THEN <Requirement >)
                                      (ELSE <Requirement>))
<Premise-Part>              ::= (<operator> <Premise-Part> {<Premise-Part>})
                                | <Simple-Component>
                                | <Premise-Part>
<operator>                  ::= AND | OR
<Maintenance-Mode>          ::= Addition | Modification | Deletion | Reference
<Maintenance-Property>      ::= IO | Conditional | Assignment | Data-Interface | Database
```

::= : *consists of*
{x} : *zero or more occurrence of x*
| : *OR relationship*

그림 4. 유지 보수 요구 표현 방법에 대한 BNF

이상에서 제안된 유지 보수 요구 표현 방법에 대한 이해를 돕기 위해 한국 전력에서의 유지 보수 요구에 대한 실례를 들고 그 표현 결과를 보이고자 한다. 다음은 한국 전력의 정보 시스템 관리 부서에 접수 되었던 유지 보수 요구서 MR190.05-2754의 요구 내역을 보여주고 있다.

만약 이와 같은 유지 보수 요구가 신청된 당시의 규정의 내용은 계약 종별이 산업인 경우, 전력사용량 500시간을 초과할 때 요금의 30%를 추가 적용했었다면 위의 유지 보수 요구는 (그림 4)에서 제안된 표현 방법에 따라 다음과 같이 표현될 수 있다.

> **유지보수요구:**
> **계약 종별(Contact Cateory)이 산업이고 전력사용량이 450시간을 초과할 경우 초과된 전략사용량에 대한 요금의 50%를 추가 적용한다.**

MR190.05-2754 ::= (*"초과 사용량에 대한 추가 요금"*)
"초과 사용량에 대한 추가 요금" ::=
((**IF** (**AND** *"계약 종별이 산업인 조건"*
"전력사용량이 450시간 초과인 조건"))
(**THEN** *"초과 사용량에 대한 요금의 50% 추가"*))
"계약 종별이 산업인 조건" ::= **REQ** (Reference, Conditional, Contract-Category)
"전력사용량이 450시간 초과인 조건" ::= **REQ** (Modification, Conditional, Monthly-Consumption)
"초과 사용량에 대한 요금의 50% 추가" := **REQ** (Modification, Assignment, Monthly-Fare)

한편 METASOFT는 사용자가 이러한 유지 보수 요구 표현 정보 입력을 쉽게 수행할 수 있도록 GUI 환경을 지원하고 있으며 시스템 내부적으로 앞의 예와 같은 형태의 객체 지향적 정보 형태로 전환 저장하게 된다. 이제 다음 절에서는 이상에서 살펴본 정보 시스템 설계 지식 표현과 유지 보수 요구 표현 방법을 기초로 하여 유지 보수 사례 표현 방법을 제안하고자 한다.

4.3 유지 보수 사례의 표현

사례 표현 방법은 사례 기반 추론에 있어 그 효과와 효율성을 결정하는 매우 중요한 요인이다. 한편 앞에서 언급한 바와 같이 유지 보수 사례 표현은 문제 기술 영역 표현과 해법 기술 영역 표현 부분의 두 가지 부분으로 나뉘어진다. 본 연구에서는 용어의 혼돈을 피하기 위해 문제 기술 영역과 해법 기술 영역이라는 용어를 유지 보수 분야에 적합한 유지 보수 요구 부분과 이에 따른 유지 보수 결과 부분이라 부르고자 하며 이들 두 부분에 대한 표현 방법을 결정하는 것이 곧 유지 보수 사례 전체 표현 방법의 결정을 의미한다.

따라서 먼저 유지 보수 요구 부분에 대한 표현 방법이 갖추어야 될 요건을 살펴보기로 하자. 유지 보수 사례를 구성하는 유지 보수 요구 부분에 대한 표현 방법 역시 유지 보수 담당자에 의해 기록 가능하며 동시에 이해가 용이해야 한다. 또한 각각의 유지 보수 요구의 특성들을 반영할 수 있도록 충분

한 표현력도 갖추어야만 한다. 따라서 본 연구에서는 유지 보수 사례의 유지 보수 요구 부분의 표현을 위해 이들 요건에 대해 충분한 분석을 거친 유지 보수 요구 표현 방법을 채택하여 이용하고자 한다.

한편 우리는 이미 본 장의 앞 부분에서 정보 시스템 원문 자체가 유지 보수 결과 표현하기에는 충분치 않은 몇 가지 이유들을 살펴 보았다. 이외에도 사례를 바탕으로 유지 보수 업무를 지원하고자 하는 본 연구의 목적 관점에서 유지 보수 업무의 효율을 위해 새로운 유지 보수 사례의 습득은 자동화되어야 한다. 그러나 유지 보수 결과를 수정 이전의 원문과 이후의 원문 간의 비교 수준에서는 의미 있는 결과 파악이 극히 어려운 실정이다. 따라서 유지 보수 결과의 표현은 유지 보수 후 보수 결과 추적이 용이한 형태로 이루어져야 하며 이는 유지 보수의 결과를 시스템 설계 수준의 정보로 관리함으로써 가능해 진다. 이들 설계 수준의 정보 표현을 갖는 유지 보수 결과는 물론 시스템 원문과의 연계 정보도 유지하여야 만 정확한 유지 보수 대상 인식이 가능할 수 있다.

따라서 우리는 이상의 유지 보수 결과 표현에 대한 요건을 만족시키기 위해 METASOFT에서의 시스템 설계 정보 표현 방법을 유지 보수 결과 부분을 표현하기 위한 기본 표현 틀로 채택하고 이제까지의 논의들을 바탕으로 (그림 5)의 유지 보수 사례 표현 방법을 설계 제안하고자 한다.

```
<Maintenance-Case>          ::= <Case-Component> {<Case-Component>}
<Case-Component>            ::= <Simple-Case-Component> | <Complex-Case-Component>
<Simple-Case-Component>     ::= (REQ(<Maintenance-Mode>, <Maintenance-Property>,
                                <Keyword> {<Keyword>}), <Case-Result-Set>)
<Complex-Case-Component>::= ((IF <Premise-Part>) (THEN <Maintenance-Case>))
                            | ((IF <Premise-Part>) (THEN <Maintenance-Case>)
                                  (ELSE <Maintenance-Case>))
<Premise-Part>              ::= (<operator> <Premise-Part> {<Premise-Part>})
                            | <Simple-Case-Component>
                            | <Premise-Part>
<Case-Results-Set>          ::= <Case-Results-in-Program> {<Case-Results-Set>}
<Case-Results-in-Program>   ::= <Case-Result> {<Case-Results-in-Program>}
<Case-Result>               ::= (<Maintenance-Mode> <Modified Clause>) {<Case-Result>}
```

그림 5. 유지 보수 사례의 표현 방식에 대한 BNF

(그림 5)에서 볼 수 있듯이 하나의 유지 보수 사례(Maintenance-Case)는 (그림 4)의 유지 보수 요구(Requirement)에 해당되며 이는 하나의 유지 보수 업무에 대해 복수의 유지 보수 사례가 발생할 수 있다는 점을 의미한다. 한편 특이한 점은 유지 보수 사례(Maintenance-Case) 표현은 유지 보수 요구(Requirement) 표현과 동일한 구조를 가지므로 단순 유지 보수 사례 요소(Simple-Case-Component)가 유지 보수 요구 표현에서의 단순 유지 보수 요구 요소(Simple-Requirement-Component)에 해당되게 되는데 (그림 5)와 같이 단순 유지 보수 요구 요소(Simple-Requirement-Component) 표현과 함께 그에 따른 유지 보수 결과 집합(Case Results Set)의 짝으로써 구성되고 있다. 이는 유지 보수 요구 구조에 속하는 단순 유지 보수 요구 요소별로 그 유지 보수 결과를 관리

함으로써 사례 표현에 유지 보수 요구의 구조 특성이 유지 보수 결과에 반영되도록 하기 위한 한 방법이며 이로 인해 5장에서 제안될 유지 보수 대상 인식 방법론이 보다 정교한 유지 보수 대상 인식을 할 수 있는 바탕이 된다.

이에 덧붙여 (그림 5)의 유지 보수 사례 표현 방법을 좀 더 설명하자면 먼저 단순 유지 보수 사례 요소를 구성하고 있는 유지 보수 결과 집합(Case-Result-Ste)은 하나 또는 복수의 관련 프로그램 사례(Case-Results-in-Program) 단위로 나뉘어지게 되는데, 이는 각 프로그램에 대한 이력 관리 및 사례 관리의 용이성을 위해 색인 정보로서 활용되게 된다. 이에 따라 관련 프로그램 사례들은 다시 하나 또는 복수의 유지 보수 결과(Case-Result) 표현들로 구성되는데 이들 각 유지 보수 결과는 유지 보수로 인해 변화된 정보 시스템 원문 부분들을 METASOFT의 설계 정보 표현 방법을 이용하여 표현함으로써 완성된다.

이제 이상에서 제안된 유지 보수 사례 표현 방법을 이용하여 앞 절에서 사용했던 MR190.05-2754의 유지 보수 요구와 이에 따른 한국 전력의 전기 요금 관리 시스템에서의 실제 유지 보수 결과를 하나의 유지 보수 사례로 표현해 보고자 한다. (그림 6)은 유지 보수 업무 MR190.05-2754의 유지 보수 요구 표현에 나타난 유지 보수 요구 요소인 "*계약 종별이 산업인 조건*"을 시스템에 반영하기 위한 유지 보수 작업에 관련되었던 프로그램 CAM30JOJ와 CAJ94101에 대한 원문 부분 예를 보여 주고 있다.

```
PROGRAM CAM30JOJ
.....
015600      MOVE CAJOJ-ED-CATEG (I) TO CATEG-OVER-X.
015610*     CATEG-OVER-OK is a redefined variable of CATEG-OVER-OK
015700      IF NOT CATEG-OVER-OK
015800           MOVE OVER-CHK TO OVER-CHK-SKIP(I)
.....
018410*     OVER-CHK-SKIP-ITEM is a redefined variable of OVER-CHK-SKIP
018500      IF TEMP-FARE-ILSU-TOT = W-NOW-YEOK-ILSU AND
018600           OVER-CHK-SKIP-ITEM = SPACE AND
018700           CA01-MON-KW > 3
018800           MOVE OVER-SIGN TO CAJOJ-OVER-SIGN.
                   .....

PROGRAM CAJ94101
.....
028500      IF CAJOJ-OVER-SIGN = '*'
028600           PERFORM 3210-OVER-FARE-COMPUTE.
                   .....
```

그림 6. CAM30JOJ와 CAJOJ94101 프로그램에 대한 유지 보수 결과 예

다음은 유지 보수 요구서 MR190.05-2754에서의 유지 보수 요구 사항과 이에 대한 (그림 6)과 같은 유지 보수 결과를 갖는 유지 보수 사례를 (그림 5)의 표현 방법을 따라 작성된 예를 보여주

고 있다. 먼저 'MR190.05-2754-CASE#1/1' 객체의 객체 명칭은 이 사례가 유지 보수 업무 요청, MR190.05-2754의 유지 보수 요구에 대한 유지 보수 사례이고 또한 이 업무 요청 시점에 유일하게 발생한 유지 보수 사례임을 의미하고 있다. 특히 MR190.05-2754-CASE#1/1의 *CASE-DETAILS* 항목은 (그림 5)의 표현 방법에 따라 표현된 유지 보수 사례(Maintenance-Case) 내역을 보여주고 있는데 이에 따르면 두 개의 조건과 하나의 할당 관련 단순 유지 보수 요구 요소가 발생하였고, 이들 각각의 유지 보수 요구 요소들이 각각 MR190.05-2754-CASE#1/1-REQ#1, MR190.05-2754-CASE#1/1-REQ#2, MR19 0.05-2754-CASE#1/1-REQ#3 등의 유지 보수 결과에 의해 처리되었음을 알 수 있다. 이러한 유지 보수 결과들은 다시 복수의 하부 프로그램별 유지 보수 결과로 나뉘어지게 되는데 MR190.05-2754-CASE#1/1-REQ#1를 예로 든다면 *CASE-RESULTS-IN-PRGOGRAM*라는 항목에 소속 프로그램별 유지 보수 결과를 가지게 되는데 여기에서는 MR190.05-2754-CASE#1/1-REQ#1-CAM30JOJ, MR190.05-2754-CASE#1/1-REQ#1-CAM85JOJ, MR190.05-2754-CASE#1/1-REQ#1-CAJ94101등이 이에 해당되게 되는데 이는 곧 MR190.05-2754-CASE#1/1의 첫번째 유지 보수 요구 요소인 REQ(*Modification, Conditional, Contract-Category*)을 반영하기 위해 CAM30JOJ, CAM85JOJ, CAJ94101등의 프로그램이 유지 보수 되었다는 것을 의미한다.

한편 MR190.05-2754-CASE#1/1-REQ#1-CAM30JOJ은 CAM30JOJ의 프로그램 내에서 구체적으로 변화된 정보 시스템 설계 정보를 저장하게 되는데 CASE-RESULTS 항목의 내용 중 '(*Reference* CAM30JOJ-3210-BUHA-CHNG-ITEM-MOV E-015600)'는 CAM30JOJ-3210-BUHA-CHNG-ITEM-MOVE-015600라는 절을 참조하여 유지 보수가 이루어졌다는 사실을 나타내고 있다. 또한 아래에는 이들 각각의 유지 보수 대상이 되었던 절에 대한 관련 정보 시스템 설계 정보의 객체 지향적 표현도 예시되고 있는데 이들은 (그림 8)에서 굵은 글씨로 나타낸 실제 유지 보수 대상이 되었던 원문과 일치하게 된다. 이외에도 유지 보수 사례 표현에 있어 사례간의 파생 관계도 표현할 수 있도록 하고 있는데 MR190.05-2754-CASE#1/1의 예에서 볼 수 있듯이 *DERIVED-CASE-NAME*항목이 해당 사례가 파생 되어진 과거 사례를 가리키게 되며 이러한 파생 관계 정보 역시 다음 장에서 제안될 사례 기반 유지 보수 대상 인식 방법론의 유사 사례 선정 과정에서 사용하게 된다.

;; MR190.05-2754-CASE#1/1
TITLE : "조과 사용량에 대한 추가 요금"
DATE-OF-ENFORCEMENT : 96/04/05
ACCEPT-DEPARTMENT : "정보시스템실"
REQUEST-DEPARTMENT : "영업운영실"
REFERENCE-DEPARTMENT : "중앙전자계산소"
DERIVED-CASE-NAME :
CASE-DETAILS :
((IF (AND (REQ(*Reference, Conditional, Contract-Category*) **MR190.05-2754-CASE#1/1-REQ#1**)
(REQ(*Modification, Conditional, Monthly-Consumption*) **MR190.05-2754-CASE#1/1-REQ#2**)))
(THEN REQ(*Modification, Assignment, Monthly-Fare*) **MR190.05-2754-CASE#1/1-REQ#3**))

```
        }}

;; MR190.05-2754-CASE#1/1-REQ#1
        IS-A :                          CASE-RESULTS-SET
        PART-OF :                       MR190.05-2754-CASE#1/1
        PROGRAM-NAME :                  CAM30JOJ
        CASE-RESULTS-IN-PROGRAM :       MR190.05-2754-CASE#1/1-REQ#1-CAM30JOJ
                                        MR190.05-2754-CASE#1/1-REQ#1-CAM85JOJ
                                        MR190.05-2754-CASE#1/1-REQ#1-CAJ94101
                                            ...
        }}

;; MR190.05-2754-CASE#1/1-REQ#1-CAM30JOJ
        IS-A :                  CASE-RESULTS-IN-PROGRAM
        PART-OF :               MR190.05-2754-CASE#1/1-REQ#1
        CASE-RESULTS :          (Reference CAM30JOJ_3210-BUHA-CHNG-ITEM-MOVE_015600)
                                (Reference CAM30JOJ_3210-BUHA-CHNG-ITEM-MOVE_015700)
                                (Reference CAM30JOJ_3210-BUHA-CHNG-ITEM-MOVE_015800)
                                (Reference CAM30JOJ_3200-KWH-MOVE-RTN_018500)
                                (Reference CAM30JOJ_3200-KWH-MOVE-RTN_018800)
                                }}

;; MR190.05-2754-CASE#1/1-REQ#1-CAJ94101
        IS-A :                  CASE-RESULTS-IN-PROGRAM
        PART-OF :               MR190.05-2754-CASE#1/1-REQ#1
        CASE-RESULTS :          (Reference CAJ94101_3200-KWH-FARE-COMPUTE_028500)
                                (Reference CAJ94101_3200-KWH-FARE-COMPUTE_028600)
                                }}


;; CAM30JOJ_3210-BUHA-CHNG-ITEM-MOVE_015600
        IS-A :                          ASSIGNMENT
        IN-SECTION :                    CAM30JOJ_3210-BUHA-CHNG-ITEM-MOVE
        LHS-VARIABLE :                  CATEG-OVER-X
        RHS-VARIABLE :                  CAJOJ-ED-CATEG
        }}

;; CAM30JOJ_3210-BUHA-CHNG-ITEM-MOVE_015700
        IS-A :                          IF
        IN-SECTION :                    CAM30JOJ_3210-BUHA-CHNG-ITEM-MOVE
        CONDITIONAL-VARIABLES :         CATEG-OVER-OK
        THEN-CLAUSES :                  CAM30JOJ_3210-BUHA-CHNG-ITEM-MOVE_015800
        ELSE-CLAUSES :
        }}
        ...

;; CAJ94101_3200-KWH-FARE-COMPUTE_028500
        IS-A :                          IF
        IN-SECTION :                    CAJ94101_3200-KWH-FARE-COMPUTE
        CONDITIONAL-VARIABLES :         CAJOJ-OVER-SIGN
        THEN-CLAUSES :                  CAJ94101_3200-KWH-FARE-COMPUTE_028600
        ELSE-CLAUSES :
        }}
        ...
```

이상에서의 유지 보수 사례 정보는 METASOFT의 GUI 환경을 이용하여 위와 같은 객체 지향적 표현으로 사용자가 용이하게 입력할 수 있으나, 만약 전반적 유지 보수 업무가 METASOFT를 이용하여 이루어질 경우 이러한 유지 보수 사례의 수집과 객체 지향형 정보 표현 및 저장이 완전 자동화될 수 있다. 이제 우리는 이렇게 수집 표현된 유지 보수 사례들을 이용하여 향후 새로이 발생하는 유지 보수 요구에 대해 유지 보수 대상을 인식하여 제공할 수 있는 방법론을 제안하고 또 그 유지 보수 지원 성과와 타당성을 분석해 보고자 한다.

V. 사례 기반 유지 보수 대상 인식

본 장에서는 먼저 유지 보수 요구에 대해 유지 보수 되어야 하는 프로그램의 명령 절 들이나 데이터베이스의 항목들을 인식하여 유지 보수 담당자에게 제공하기 위한 목적으로 사례 기반 유지 보수 대상 인식(Case Based Maintenance Point Identification) 방법론을 제안하고자 한다. 이 방법론은 크게 다음의 네 가지 연속적 단계들로서 구성되어 있는데 이들 각각의 단계들은 다음과 같다.

(1) 유지 보수 요구 표현 및 입력 단계: (그림 1)에서 보듯이 새로운 유지 보수 요구가 유지 보수 담당자에 의해 작성되고 이는 자동적으로 유지 보수 요구 표현 양식에 따라 저장되게 된다.

(2) 유사 유지 보수 사례 추출 단계: 전 단계에서 작성된 유지 보수 요구 표현을 이용해 가장 의미 있는 과거 유지 보수 사례를 사례 베이스(Case Base)로부터 추출하는 단계이다.

(3) 유지 보수 사례 결과 검증 및 조정 단계: 앞의 제 (2)단계에서 추출된 유지 보수 사례의 유지 보수 결과가 새로이 작성된 유지 보수 요구에 대해 정확한 유지 보수 대상인지를 검증하고 그 미비점을 보완하는 단계이다. 이 단계는 범용적 유지 보수 대상 탐색 기법인 MPI 알고리즘(*Lee et al.*, 1997)을 이용하여 수행된다.

(4) 유지 보수 사례 자동 습득 단계: 마지막 단계로서 제안된 유지 보수 대상을 바탕으로 유지 보수 담당자에 의해 수행된 유지 보수 결과를 수정된 원문을 이용하여 자동적으로 분석하고 (1)단계에서 작성되었던 유지 보수 요구와 함께 새로운 유지 보수 사례를 자동 생성하여 사례 베이스에 저장하게 된다.

이상의 절차 중 유지 보수 요구 표현 단계에 대해서는 이미 앞에서 충분히 다루었으며, 마지막 유지 보수 사례 자동 습득 단계는 나머지 단계들에 비해 상대적으로 단순하고 기계적 절차이므로 본 장에서 그 설명을 생략하고자 하며, (*Lee et al.*, 1995)에서 보다 자세한 내용을 참조할 수 있다. 이제 이하의 절들에서는 2 단계와 3 단계를 수행하기 위한 이론적 배경과 그 수행 과정을 좀 더 자세히 논의하고자 한다.

5.1 유지 보수 사례간의 유사성 정의

새로운 유지 보수 요구에 대해 가장 의미 있는 유지 보수 결과를 갖는 과거 유지 보수 사례를 추출하기 위해서는 무엇보다도 먼저 이에 따른 유지 보수 사례의 유지 보수 요구 부분 표현간의 유사성(Similarity)에 대한 적절한 정의가 필요하다. 그러나 유지 보수 요구가 일반적인 사례 추론 방법론

에서 채택하고 있는 방식(Kolodner 1993)처럼 여러 개의 평행적 수준에 존재하는 속성들을 가지고 있지도 않으며 동시에 이들 속성간의 상대적 중요성이 하나의 함수 형태로 수치적 비중을 이용하여 평가할 수도 없는 실정이다. 이 뿐만 아니라 하나의 속성에 대해서도 그 속성이 가지는 값이나 형태들간의 차이 조차 계량적으로 측정하기가 매우 곤란하다. 이러한 어려움은 (그림 5)의 사례 표현 방법에서 볼 수 있듯이 각각의 유지 보수 요구 부분 표현이 여러 개의 같은 수준의 속성들의 값으로 표현되는 것이 아니라 논리적 구조와 구조 내의 속성 값들을 이용하여 이루어짐에 기인한다.

한 예로 유지 보수 요구 요소를 구성하는 속성 중 하나인 유지 보수 속성(Maintenance Property)에 대한 속성 값으로 '*IF*'를 다른 사례의 해당 속성 값은 '*ASSIGNMENT*'이거나 혹은 '*I/O*'를 가지게 될 경우, '*IF*'의 값이 어느 경우에 더 가까운지 측정한다는 것은 거의 불가능하다. 유지 보수 요구 표현에 대해 기존의 사례 추론 방법론을 적용하기 어렵게 하는 또 하나의 이유는 실질적인 유지 보수 사례간의 유사성이 각 유지 보수 요구 요소간의 유사성보다는 유지 보수 요구의 논리적 구조 유사성에 더 깊이 관련되어 있다는 사실이다. (그림 7)의 예는 이러한 유지 보수 사례간의 유사성 평가를 수행할 때 유지 보수 요구의 구조의 중요성을 쉽게 이해할 수 있도록 해주고 있다. 만일 (그림 7)에서와 같은 구조의 신규 유지 보수 요구가 입력되었을 때, 유지 보수 요구 요소 집합간의 일치되는 정도를 이용하여 유사성을 평가한다면 사례 2가 사례 1에 비해 신규 요구에 더 유사하다고 볼 수 있으나 실질적인 유지 보수 업무 수행 관점에서 볼 때 신규 요구는 사례 1에 해당되는 원문의 THEN 절에 'B-에 해당되는 사항을 추가함으로써 사례 2의 경우보다는 더 용이하게 유지 보수할 수 있다. 따라서 신규 요구에 대한 사례 1과 사례 2의 유사성을 평가했을 때, 사례 1이 더 높은 유사성을 나타낼 수 있는 유사성 평가 기준이 제시되어야 의미 있는 사례 기반 유지 보수 대상 인식이 가능해질 수 있다.

신규요구	사례 1	사례 2
((IF A) (THEN B))	((IF A) (THEN C))	((IF B) (THEN A))

그림 7. 사례간의 구조 유사성 예 (A, B, C는 임의의 서로 다른 유지 보수 요구 요소임.)

이러한 유지 보수 사례간의 유사성 평가에 있어서의 문제점들을 해결하기 위해 우선 이상에서 논의된 관점에서 유지 보수 사례들을 구별할 수 있는 유지 보수 사례의 특성 요인들을 파악하여야 한다. 본 연구에서는 이를 위해 유지 보수 요구 표현의 근간을 이루는 다음의 네 가지 특성 요인들을 유지 보수 사례간의 유사성 평가 요인으로서 이용하고자 한다.

(1) 사례 구조 요인(Case Structure Factor): IF, AND, THEN, ELSE 등의 논리 및 조건 연산자들을 이용한 논리 구조와 각 연산자와 연계된 절(Clause) 단위에서의 단순 사례 요소(*Simple-Case-Component*)의 순서와 집합 구성에 관련된 유지 보수 사례 특성 요인이다.

(2) 유지 보수 유형 요인(Maintenance Mode Factor): 단순 사례 요소 내부의 유지 보수 유형들간의 차이를 이용한 특성 요인이다.

(3) 키워드 유사성 요인(Keyword Similarity Factor): 각 단순 사례 요소 내부의 키워드들에 관련된 특성 요인으로서 키워드간의

유사성을 측정하고자 한다.

(4) 유지 보수 속성 요인(Maintenance Property Factor): 단순 사례 요소 내부의 유지 보수 속성 관련 특성 요인으로서 각 단순 사례 요소의 유지 보수 속성(Maintenance-Property)을 나타내며 이들 속성간의 차이가 곧 사례의 유사성에 영향을 미치게 된다.

이제 우리는 이상의 네 가지 사례 특성 요인들을 이용하여 유지 보수 사례간의 유사성을 정의하고자 한다. 먼저 앞에서의 네 가지 특성 요인 중 사례 구조 요인이 유사성 평가에 있어서 가장 중요하다는 사실은 이미 (그림 7)의 예를 통해 확인된 바 있으며 따라서 사례 구조 요인이 다른 나머지 세가지 요인에 대해 유사성 평가에 있어 우선권을 갖게 된다. (그림 9)의 예를 이용해 다시 설명하자면 앞에서의 네 가지 요인에 따라 이들 세 가지 사례를 비교해 보자. 먼저 '신규 사례-가 사례 2와는 사례 구조 요인만을 제외하고 나머지 세 가지 요인에 있어 완벽하게 일치하고 있다. 반면 사례 1은 사례 구조 요인 관점에서는 '신규 사례-와 일치하고 있으나, 나머지 세가지 관점에서는 완벽하게 일치하고 있지는 않다. 이러한 사실은 이들 네 가지 요인 중에서 사례 구조 요인이 최우선적 고려 항목이라는 것을 뒷받침하고 있다. 한편 이와 유사한 이유로 인해 나머지 특성 요인들간의 우선 순위는 앞에서의 언급된 순서에 준하게 된다.

이렇게 네 가지 요인들간의 유지 보수 사례 유사성 평가에 있어서의 우선 순위가 결정되었다 해서 유지 보수 사례간의 유사성 정의가 완전히 끝난 것은 아니며 이제 이들 네 가지 요인 각각에 관련된 유사성의 평가를 수행할 수 있는 기준이 마련되어야 한다. 예를 들어 설명하자면 임의의 사례 A, B, C에 대해 A와 B의 차이와 A와 C와의 차이가 모두 같은 우선 순위를 갖는 특성 요인에 의한 차이라면 이에 대해서도 각각의 차이에 대한 유사성 정도를 구분할 수 있는 기준이 있어야 된다는 것을 의미한다. 따라서 이후에서는 이들 네 가지 특성 요인들 각각에 대한 유사성 평가 방법에 대해 다루고자 한다.

(1) 사례 구조 요인(Case Structure Factor)의 평가 방법

사례 구조 요인의 유사성을 평가하기 위해 본 연구에서는 두 사례 구조 요인의 속성간의 유사 정도를 평가하기 보다는 이들간의 차이를 평가함으로써 궁극적 유사성을 평가하는 접근 방법을 취하고 있다. 이는 하나의 논리 구조가 다른 논리 구조 형태를 취하기 위해 뒤에서 정의하게 될 논리 구조 변화 연산을 통해 어느 정도의 형태 변환(Form Transformation) 과정을 수행해야 되는 지를 평가하기가 용이하기 때문이다. 따라서 만일 동일한 논리 구조를 갖는 두 사례의 차이는 0으로서 정의되며, 하나의 구조 요소 차이가 발생할 때마다 1점씩 유사성 정도가 감소하게 된다. 다음의 예는 이러한 사례 구조 요인의 평가 예를 보여주고 있으며 예에서의 '*'는 임의의 단순 사례 요소를 의미하고 있다.

가) **((IF *A*) (THEN *B*))**
나) **((IF (AND *A* *)) (THEN *B*))**
다) **((IF *A*) (THEN *B* *))**
라) **((IF *A*) (THEN *B*)) * or * ((IF *A*) (THEN *B*))**
마) *A* or *B* :: 각각 **((IF *A*) (THEN))** 또는 **((IF) (THEN *B*))**와 동일
바) **((IF (AND *A* *)) (THEN *B* *))**

위의 예에서 가) 사례에 대한 나머지 사례의 비유사성 정도를 측정하기 위해 우리는 먼저 사례 구조에 대한 구조 변화(Structure Change) 연산 방법으로 다음과 같이 여섯 가지를 정의하고자 한다.

ㄱ) 조건부 추가 연산(Premise Addition Operation)

ㄴ) 조건부 삭제 연산(Premise Deletion Operation)

ㄷ) 결론부 추가 연산(Consequent Addition Operation)

ㄹ) 결론부 삭제 연산(Consequent Deletion Operation)

ㅁ) 절차 추가 연산(Sequence Addition Operation)

ㅂ) 절차 삭제 연산(Sequence Deletion Operation)

이상의 사례 구조 변화 연산들은 모두 유지 보수 사례에 적용되어 그 결과로서 원래의 유지 보수 사례 구조가 변화되게 된다. 예를 들어 가)의 사례에 조건부 추가 연산을 적용하면 나)의 형태로 유지 보수 사례가 변화하게 되고 결론부 추가 연산을 적용하면 다)의 형태로 변화하게 된다. 이제 이러한 구조 변환 연산을 이용하여 다음과 같이 유지 보수 사례의 구조에 대한 유사성을 정의하고자 한다.

사례 구조 유사성 $(B, A) = -1 \times$ *Number of Elements of Structure Change Operations Set on B*
where A and B are case representations
and *Structure Change Operations Set* = $\{Op_1, Op_2, ..., Op_n\}$
and Op_n is n-th operation which was applied to B_{n-1}
and B_{n-1} is a case structure form which was transformed by applying $\{Op_1, Op_2, ... Op_{n-1}\}$ on it sequentially
and A and B_n have same structure.

이러한 사례 구조 유사성 정의에 따르면 가)와 나) 사례의 경우, 가) 사례에 대해 조건부 추가 연산을 적용함으로써 서로 같은 논리 구조를 가지게 되며 따라서 두 사례 간의 유사성 정도는 -1로서 평가 된다. 역시 같은 방법으로 다), 라), 마)의 사례에 대해서도 각각 가) 사례에 결론부 추가 연산, 절차 추가 연산, 조건 또는 결론부 삭제 연산을 적용하면 동일한 논리 구조를 가지게 되는데 따라서 이들 세 경우에 모두 -1의 유사성을 보이게 된다. 만약 다른 나머지 세가지 사례 특성 요인에서의 유사성도 같다면 이들 나), 다), 라), 마)의 사례는 모두 가) 사례에 대해 동등한 유사성을 가지는 것으로 평가하게 된다. 한편 바)의 사례는 가)의 사례에 대해 조건부 추가 연산과 결론부 추가 연산을 연속적으로 적용해야 비로소 동일한 구조를 가질 수 있으므로 이 때의 유사성 정도는 정의에 따라 -2로 평가되며 이는 바) 사례가 나머지 사례들에 비해 상대적으로 가) 사례와의 유사성이 떨어진다는 것을 의미한다.

앞의 예에서 만일 또 다른 사례 '((IF A) (THEN B))'가 사례 베이스에 존재한다면 가)의 사례에 대해 0의 유사성 정도를 가지게 되며 따라서 당연히 다른 모든 사례들보다 유사 사례 추출 과정에서 우선하게 된다. 예외적으로 단순 사례 요소의 유지 보수 유형(Maintenance Mode)이 추가(Addition)인 경우에는 해당 단순 사례 요소를 제거하고 위와 같은 유사성 평가 작업을 수행하게 된다.

(2) 유지 보수 유형 요인(Maintenance Mode Factor)의 평가 방법

앞에서 정의한 사례 특성 요인간의 우선 순위에 따라 사례 구조간의 유사성 평가가 이루어진 후에는 사례를 구성하는 각 단순 사례 요소의 유지 보수 유형 요인 관점에서 추가적인 유사성 평가가 진행되게 되는데 다음의 네 가지 사례 표현 예를 이용하여 이를 설명하고자 한다(여기서 'A'는 임의의 관련 키워드를 나타낸다).

가) ((IF REQ(*Modification, Conditional, A*)) (THEN *))

나) ((IF REQ(Reference, Conditional, A)) (THEN *))

다) ((IF REQ(*Addition, Conditional, A*)) (THEN *))

라) ((IF REQ(*Deletion, Conditional, A*)) (THEN *))

이 예에서 우리는 가) 사례에 대해 나머지 모든 사례가 사례 구조 상으로는 같은 유사성을 가지고 있지만 가) 사례에 대한 유지 보수 작업 관점에서 볼 때 나)와 다)의 경우는 'A-라는 키워드에 관련하여 조건적 원문이 현재 프로그램에 존재하는 경우이고, 라)의 경우는 그러한 원문이 이미 삭제된 경우이기 때문에 나)와 다)의 경우가 사실상 라)의 경우에 비해 더 유지 보수하기에 용이하다는 것을 알 수 있다. 또한 이러한 측면에서 볼 때, 나)와 다)는 그 유사성을 구별하기가 매우 곤란하다. 이러한 각 단순 사례 요소의 유지 보수 유형 요인에 따른 유사성 측정 기준이 아래와 같은 <표 2>에 정리되어 있다.

표 2. 유지 보수 요구에서의 유지 보수 유형에 대한 기존 사례의 유사성 정도 측정 기준

유지 보수 요구에서의 단순 사례 요소의 유지 보수 유형	사례 베이스의 유지 보수 유형			
	Addition	Deletion	Modification	Reference
Deletion, Modification, Reference	0	1	0	0

위의 <표 2>에서 유지 보수 요구의 유지 보수 유형 항목 중 추가(Addition)의 경우가 빠진 이유는 추가(Addition)의 유지 보수 유형을 갖는 단순 사례 요소들은 선행 사례 구조 비교 단계에서 이미 제거되었기 때문에 상대적으로 하위 비교 단계인 유지 보수 유형 요인 비교시에는 제외된 것이다. 이상의 정의를 바탕으로 특정 유지 보수 요구와 사례 베이스의 기존 사례와의 유지 보수 유형 요인에 따른 유사성 평가 방법은 다음과 같이 정형화될 수 있다.

유지 보수 유형의 유사성 $(B, A) = -1 \times \sum_{i=0}^{n} SP(SCC^i_A, SCC^i_B)$

where A and B are case representations which have the same structure to each other and SCC^i_A and SCC^i_B are i-th simple case component of A and B, respectively and $SP(SCC^i_A, SCC^i_B)$ is a corresponding similarity point between SCC^i_A and SCC^i_B in Table 2 and n is the number of simple case components of A (or B)

(3) 키워드 유사성 요인(Keyword Similarity Factor)의 평가 방법

키워드간의 유사성 평가는 4.1.2절에서 논의된 키워드간의 동의어 및 상하위 관계 지식을 바탕으

로 이루어지게 되는데 <표 3>은 이러한 키워드 간의 유사성 평가가 허용되는 경우와 그 때 적용되는 유사성 평가 방법들을 보여주고 있다.

특히 <표 3>에서 검게 표시된 부분이 유사성 평가가 허용되지 않는 경우를 의미하며 이는 곧 아무런 관계도 성립되지 않는 키워드간의 유사성과 동일하게 취급하게 된다는 것을 의미한다. 유지 보수 사례 간의 키워드 유사성은 <표 3>에 근거하여 다음과 같이 정의될 수 있다.

$$\text{키워드 유사성 } (B, A) = -1 \times \sum_{i=1}^{n} \sum_{(s,t)\in MS_i(A,B)} SP(KW^s_{i,A}, KW^t_{i,B})$$

where A and B are case representations which have the same structure to each other
and $KW^s_{i,A}$ and $KW^t_{i,B}$ are s-th and t-th keywords of
simple i-th case component of A and B, respectively
and (s, t) is a member of $MS_i(A, B)$
and $MS_i(A, B)$ is a set of best matching pairs $\{..(s, t)..\}$
in which each pair satisfy $SP(KW^s_A, KW^t_B) = \text{Min. } SP(KW^s_A, KW^k_B)$ for all available k
and s has unique appearance in this set
and set of all s which appears in $MS_i(A, B)$ is equal to set of keywords in SCC^i_A.

표 3. 키워드 간의 유사성 요인 평가 기준과 방법

유지 보수 유형	유사성 평가 방법				
	완전 일치	동의어	상위 탐색	하위 탐색	상하위탐색
Addition	0	0	*length of path*		
Deletion	0	0		*length of path*	
Modification	0	0	*length of path*	*length of path*	*length of path*
Reference	0	0	*length of path*	*length of path*	*length of path*

위의 방법에 따라 다음의 세 가지 예를 들어 유지 보수 사례간의 키워드 유사성을 계산해 봄으로써 그 특성을 살펴보고자 한다. 먼저 가)와 나)의 사례간의 키워드 유사성을 계산하기 위해서는 각각의 유지 보수 요구 요소들 간의 키워드 유사성을 먼저 계산해야 하는데 우선 {*Contract-Category, Category*}은 상호 상하위 관계에 있으므로 키워드 유사성이 1이 되고 {*Monthly-Consumption, Yearly-Consumption*}의 경우는 두 키워드가 모두 *Consumption* 키워드를 상위 키워드로 하고 있으므로 키워드 유사성은 2가 된다. 마지막으로 {*Monthly-Fare, Monthly-Over-Fare*}의 경우도 첫번째 경우와 마찬가지로 서로 상하위 관계에 있으므로 유사성 정도가 1이 된다. 이에 따라 이들 두 사례간의 키워드 유사성 정도는 이들 각각의 키워드 유사성들의 합계에 대한 음수인 -4가 된다. 특수한 형태의 예로서 다)의 경우를 살펴보면 마지막 단순 사례 요소의 유지 보수 유형이 삭제(Deletion)인 경우에 해당되는데 따라서 <표 3>의 규칙에 따라 상위 탐색이 허용되지 않게 된다. 이는 곧 다) 사

례가 나) 사례와는 유사성 평가가 가능하나 가) 사례와의 키워드 유사성 평가가 허용되지 않기 때문에 궁극적으로 가) 사례가 다) 사례에 대한 유사 사례 추출 대상에서 제외됨을 의미한다.

가) **((IF(AND REQ(***Reference, Conditional, Contract-Category***)**
REQ(*Modification, Conditional, Monthly-Consumption***)))**
(THEN REQ(*Modification, Assignment, Monthly-Fare***)))**
나) **((IF(AND REQ(***Reference, Conditional, Category***)**
REQ(*Modification, Conditional, Yearly-Consumption***)))**
(THEN REQ(*Modification, Assignment, Monthly-Over-Fare***)))**
다) **((IF(AND REQ(***Reference, Conditional, Category***)**
REQ(*Modification, Conditional, Yearly-Consumption***)))**
(THEN REQ(*Deletion, Assignment, Monthly-Over-Fare***)))**

(4) 유지 보수 속성 요인(Maintenance Property Factor)의 평가 방법

이제 마지막 유지 보수 사례 간의 유사성 평가 요인인 유지 보수 속성 요인에 대한 평가 방법에 대해 설명하고자 한다. 앞에서도 언급한 바와 같이 유지 보수 속성 요인간의 차이가 사례간의 실질적 유사성을 의미하지 않는다. 따라서 앞의 세 가지 유사성 요인들을 이용하여 사례간의 유사성 비교를 진행할 때, 유지 보수 속성 요인에 대해서는 완전히 일치되는 경우를 제외하고는 유사 사례 추출 과정에서 고려되지 않는다.

5.2 유사 유지 보수 사례 추출 방법론

이제 사례 기반 유지 보수 대상 인식 방법론의2 단계에 해당하는 유사 유지 보수 사례 추출 방법론을 앞에서의 유지 보수 사례간의 유사성에 대한 정의와 평가 방법을 기반으로 설계 제안하고자 한다. 유사 유지 보수 사례 추출 방법은 크게 유사 유지 보수 사례 추출 단계와 최적 유사 유지 보수 사례 선정 단계의 두 가지 단계로 나뉘어 지는데 (그림 8)은 이중 첫 단계인 유사 유지 보수 사례 추출 단계를 순서도 형식으로 보여주고 있으며 각각의 단계에서 이루어져야 하는 작업들에 대해 설명하고자 한다.

유사 유지 보수 사례 추출 방법론의 각 단계들을 설명하기에 앞서 먼저 이하에서 사용하게 될 완전 매칭(Perfect Matching)과 유사 매칭(Similar Matching)이란 용어들에 대해 정의하고자 한다. 완전 매칭이란 유사성 평가 방법에 의해 0점의 유사성을 가지게 될 경우를 의미하며 유사 매칭이란 매칭은 허용하나 유사성 점수가 0이하의 음수 값을 가지게 되는 경우를 의미한다.

Step 1) 유사 유지 보수 추출 단계

Step 1-1) 매칭 리스트(Matching List)와 잠재 리스트(Potential List)를 각각 NULL과 사례 베이스의 전체 사례 집합으로 초기화 한다.

Step 1-2) 유지 보수 요구에 대해 사례 구조 요인 비교를 통해 완전 매칭이 성립되는 사례를 검색한다. 검색이 성공하면 Step 1-3을 실패할 경우 Step 1-7을 수행하게 된다.

Step 1-3) 사례 구조 요인 관점에서 이미 완전 매칭된 사례에 대해 해당 유지 보수 요구와 유지 보수 유형 요인 관점에서 완전 매칭 여부를 검증한다. 만일 완전 매칭이 성립되었다면 Step 1-4를 반대의 경우에는 Step

1-8을 수행하게 된다.

Step 1-4) 사례 구조 요인 관점에서 완전 매칭이 성립되었고 유지 보수 유형 요인 관점에서는 완전 혹은 유사 매칭된 사례에 대해 해당 유지 보수 요구와 키워드 요인 관점에서 완전 매칭을 검증하게 된다. 완전 매칭이 성립되는 경우 Step 1-5를 수행하고 반대의 경우에는 Step 1-9를 수행한다.

Step 1-5) 세 가지 유사성 평가 단계를 통과한 사례에 대해 유지 보수 속성 관점에서 완전 매칭 여부를 검사하며, 완전 매칭이 성립되면 Step 1-6을 수행하고, 반대의 경우에 Step 1-7을 수행한다.

Step 1-6) 추출된 사례를 매칭 리스트에 첨가하고 잠재 리스트로부터 제거한다.

Step 1-7) 잠재 리스트로부터 현재의 사례 구조 유사성 매칭 수준 이상의 결과를 가질 수 있는 사례가 존재하는지 검사한다. 존재할 경우 Step 1-2를 수행하며 반대의 경우에 Step 1-10을 수행한다.

Step 1-8) 유지 보수 유형 요인에 대한 완전 매칭이 실패한 사례에 대해 유사 매칭을 수행한다. 유사 매칭에 성공할 경우, Step 1-4로 진행하며 실패할 경우 Step 1-7을 수행하게 된다.

Step 1-9) 키워드 요인에 대한 완전 매칭이 실패한 사례에 대해 해당 유지 보수 요구의 키워드들에 대한 키워드 유사 매칭을 실시한다. 유사 매칭이 성립되면 Step 1-5를 수행시키고 반대의 경우 Step 1-7을 수행한다.

Step 1-10) 현재 매칭 리스트에 확보된 추출 사례가 존재하지 않을 경우 사례 구조 변화를 실시하기 위해 Step 1-11을 수행하며 반대의 경우 Step 1-13을 수행한다.

Step 1-11) 현재의 사례 구조 상태가 추가적 사례 구조 변화 연산을 통해 잠재 리스트에 남아 있는 사례들과 매칭 가능한지 검사한다. 가능할 경우 Step 1-12을 수행하고 불가능한 경우에는 Step 1-13을 수행한다.

Step 1-12) 현재의 사례 구조에 대한 사례 구조 변화 연산을 통해 구조 변화를 수행하여 새로운 유지 보수 요구의 논리 구조를 생성한다.

Step 1-13) 유사 유지 보수 사례 추출 과정을 종료한다.

Step 2) 최적 유사 유지 보수 사례의 선정

유사 사례 추출 과정을 통해 얻어진 매칭 리스트의 유사 사례들 중에서 최적의 유사성을 나타낸 사례를 선정하게 되는데 이 때의 매칭 리스트에 존재하는 사례들은 추출 과정에서 같은 횟수의 구조 변화 및 유사 매칭 작업이 적용된 사례들만 수집되도록 Step 1에서 통제되고 있다. 다시 말해서 추출된 사례들이 모두 두 번의 구조 변화 후 추출되었거나 아니면 한 번의 키워드 유사 매칭이 적용된 후 추출된 경우는 가능하나 반면 두 번의 구조 변화를 갖는 사례와 한번의 키워드 유사 매칭을 갖는 사례가 동시에 수집되는 경우는 발생하지 않는다는 것을 의미한다. 한편 이와 같이 수집된 사례들에 대한 유사성 평가가 본 단계에서 다시 수행되는데 구조 변화만을 통한 매칭 리스트가 구성되었을 경우 구조 변화 횟수만이 유사성을 측정하는 유일한 지표이기 때문에 해당 유지 보수 요구에 대해 모두 동일한 수준의 유사성을 가지게 되고 따라서 상위 단계에서 전달된 매칭 리스트의 사례들이 그대로 다음 과정인 유지 보수 대상 인식 과정으로 제공되게 된다. 한편 유지 보수 유형 유사 매칭이나 키워

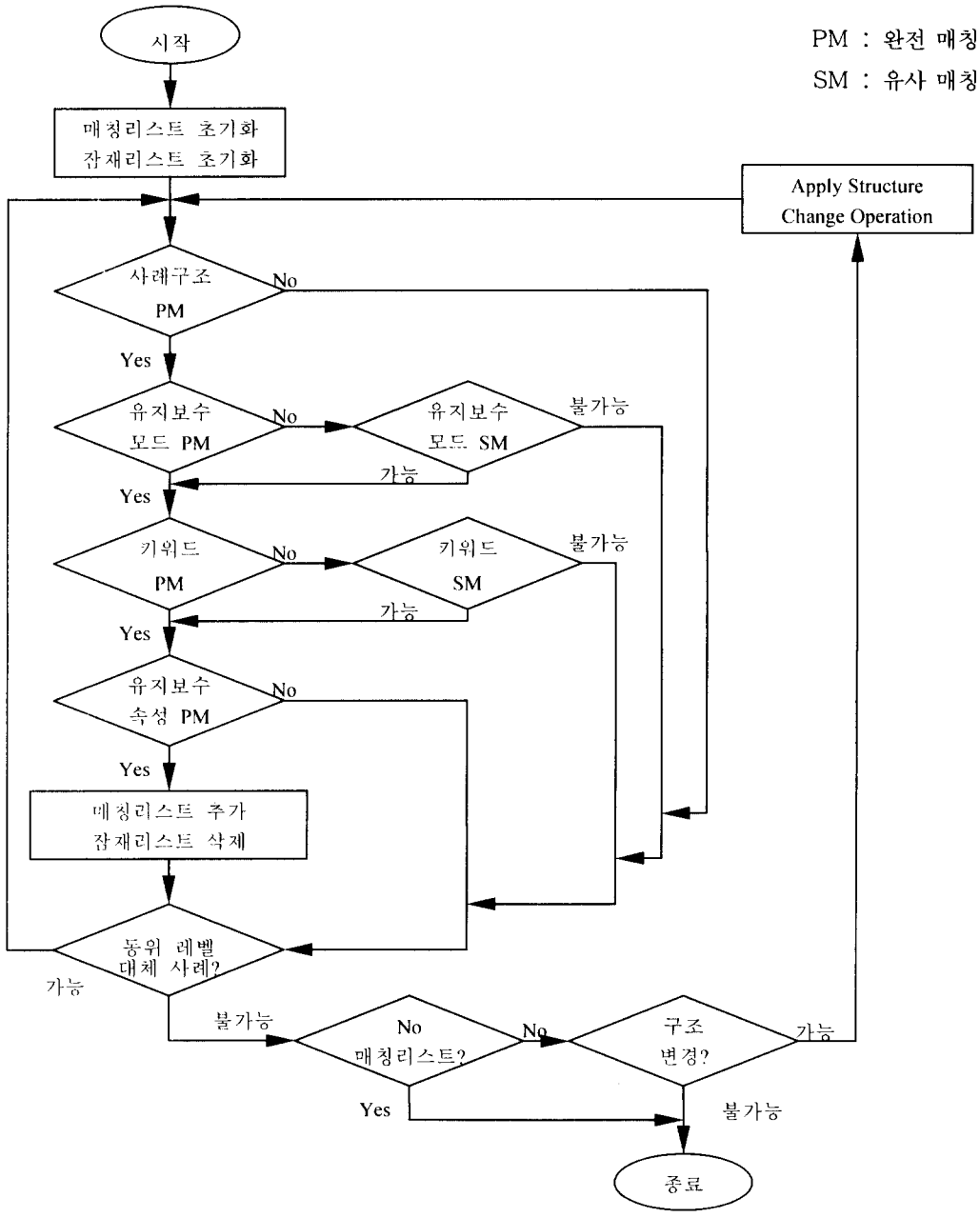

그림 8. 유사 유지 보수 사례 추출 방법

드 유사 매칭이 적용된 경우에는 유지 보수 유형 유사 매칭, 키워드 유사 매칭 순서로 앞에서의 평가 방법을 이용하여 계량화 되고 이 수치들을 비교하여 최고의 유사성을 갖는 사례를 선정하여 다음

단계인 사례 결과 조정 단계로 전달하게 된다. 만약 유지 보수 유형 유사 매칭과 키워드 유사 매칭이 동시에 수행된 경우에는 키워드 유사 매칭의 유사성이 아무리 크더라도 유지 보수 유형 유사 매칭에 의한 유사성의 크기가 유사 사례 선정에 우선적으로 고려되게 된다. 또한 이상의 선정 과정을 통해 복수의 사례가 선정되었을 경우 사례들간의 파생 관계를 조사하여 파생된 사례를 우선적으로 제시하고 파생의 근거가 되었던 사례는 매칭 리스트에서 제거하게 된다.

5.3 사례 조정(Adaptation) 및 유지 보수 대상 인식 방법

본 절에서의 사례 조정 및 유지 보수 대상 인식 방법은 앞 절에서 수행된 결과로서 추출된 유지 보수 사례들을 바탕으로 해당 유지 보수 요구에 대한 유지 보수 대상을 인식하고 최종적으로 유지 보수 담당자에게 유지 보수 대상을 제안하는 다음의 세 단계의 과정으로 구성되어 있다.

Step 1) 유사 유지 보수 사례 추출 과정에서 얻어진 매칭 리스트의 사례로부터 해당 유지 보수 결과를 추출한다. 만일 복수의 사례가 매칭 리스트에 포함된 경우 각각의 유지 보수 결과의 합집합을 유지 보수 대상으로서 제시하게 된다.

Step 2) 선행 단계에서 제시된 초기 유지 보수 대상들에 대해 METASOFT의 기본 유지 보수 대상 탐색 방법인 MPI알고리즘을 이용하여 해당 유지 보수 요구에 대한 완전성을 검증하고, 또한 불완전한 부분이 발생할 경우 이를 보완하게 된다. 이러한 과정에서 극단적인 경우에는 유사 유지 보수 추출 단계에서 유사 사례를 전혀 추출할 수 없는 경우도 발생할 수 있지만 MPI알고리즘은 완전 탐색(Full Search)을 통하여 관련 유지 보수 대상을 인식할 수 있기 때문에 탐색에 있어서의 효율성은 감소되기는 하지만 유지 보수 대상 인식의 완전성은 보장할 수 있다.

Step 3) 선행 단계들의 결과를 토대로 유지 보수 담당자에게 해당 유지 보수 요구에 대한 유지 보수 결과를 제시하게 된다.

이상에서 우리는 새로운 유지 보수 요구에 대해 사례 베이스로부터 유사한 유지 보수 사례를 추출하는 방법과 추출된 사례를 이용하여 유지 보수 대상을 제시하는 방법론을 설계 제안하였다. 이제 우리는 몇 가지 실제 사례를 이용하여 본 연구에서 제안된 방법론의 적용 과정을 예시하고 이와 함께 그 실질적 타당성과 그 성과를 검증하고자 한다.

5.4 사례 기반 유지 보수 대상 인식 과정 예시

4.3절에서 사용한 바 있는 초과 사용량 관련 사례를 이용하여 사례 기반 유지 보수 대상 인식 방법론이 어떻게 적용될 수 있는지를 예시하고자 한다. 4.3절에서 예시되었던 사례 외에도 다음과 같은 추가적인 사례 MR193.09-292-CASE#2/2가 이미 발생하여 사례 베이스에 저장되어 있는 상황을 가정하자.

{{ **MR193.09-292-CASE#2/2**
TITLE : "초과 사용량에 대한 추가 요금 관련 조정 사항"
DATE-OF-ENFORCEMENT : 96/05/10
DERIVED-CASE-NAME : **MR190.05-2754-CASE#1/1**

CASE-DETAILS :
((**IF** (**AND** (**REQ**(*Reference*, *Conditional*, *Contract-Category*)
MR193.09-292-CASE#2/2-REQ#1)
(**REQ**(*Reference*, *Conditional*, *Monthly-Consumption*)
MR193.09-292-CASE#2/2-REQ#2)
(**REQ**(*Addition*, *Conditional*, *Exceptional-Code*, *Complex-APT*)
MR193.09-292-CASE#2/2-REQ#3)))
(**THEN** (**REQ**(*Reference*, *Assignment*, *Monthly-Fare*)
MR193.09-292-CASE#2/2-REQ#4)))
}}

만약 새로운 규정 변경으로 인해 "450시간 초과 사용 전력에 대한 추가 요금 적용에 있어 적용 제외 대상을 종합 아파트 외에도 추가 제외 대상을 고려할 수 있는 코드를 조정 입력"하여야 하는 유지 보수 요구 MR193.09-986가 발생했다면 이는 다음과 같이 유지 보수 요구 입력 과정을 통하여 아래와 같은 내부 표현으로 표현될 수 있다.

{{ **MR193.09-986**
TITLE : *"초과 사용량에 대한 추가 요금 관련 조정 사항"*
DATE-OF-ENFORCEMENT : 97/04/15
REQUIREMENT-DETAILS :
((**IF** (**AND** **REQ**(*Reference*, *Conditional*, *Contract-Category*)
REQ(*Reference*, *Conditional*, *Monthly-Consumption*)
REQ(*Modification*, *Conditional*, *Exceptional-Code*,
Apartment, *Additional-Exception-Code*)))
(**THEN** **REQ**(*Reference*, *Assignment*, *Monthly-Fare*)))
}}

사례 기반 유지 보수 대상 인식 방법론을 이 예에 적용하면 첫 단계로서 유지 보수 요구 MR193.09-986에 대한 최적 유사 유지 보수 사례를 사례 베이스로부터 추출하여야 한다. 따라서 다음과 같은 유사 유지 보수 사례 추출 과정이 발생하게 된다.

유사 유지 보수 사례 추출 방법 적용 예

Step 1)

Step 1-1) 매칭 리스트가 NULL로 초기화 되며, .잠재 리스트는 {MR193.09-292-CASE#2/2 MR190.05-2754-CASE#1/1, ...}으로 초기화 된다.

Step 1-2) 유지 보수 요구 MR193.09-986의 사례 구조와 완전 매칭이 성립되는 사례를 사례 베이스로부터 탐색한다. 이 경우 탐색의 결과로서 사례 MR193.09-292-CASE#2/2가 추출된다.

Step 1-3) MR193.09-986과 MR193.09-292-CASE#2/2 간의 유지 보수 유형 요인 관점에서의 완전 매칭 여부를 확인한다. 이 경우, 처음 두 개의 REQ문에서는 완전히 일치하게 되고 세 번째 REQ문의 경우에는 각각 유지 보수 유형이 Modification과 Addition으로 다르지만 <표 2>에서의 유지 보수 유형 요인에 대한 유사성 평가 기

준에 따르면 유사성이 0으로 평가되어 완전 매칭이 성공하게 된다.

Step 1-4) MR193.09-292-CASE#2/2를 MR193.09-986에 대해 키워드 완전 매칭 여부를 검사하게 되는데 이 경우 완전 매칭에 실패하게 되며 따라서 Step 1-9를 수행하게 된다.

Step 1-9) MR193.09-292-CASE#2/2를 MR193.09-986에 대해 키워드 유사 매칭을 수행하게 되는데 이 때 -Apartment-와 -Complex-APT-간에 상하위 관계가 존재하므로 유사 매칭이 성립되게 되고 따라서 Step 1-6을 수행한다.

Step 1-6) MR193.09-292-CASE#2/2를 매칭 리스트에 포함시키고 잠재 리스트에서는 제거한다.

Step 1-7) 현재의 유지 보수 요구의 논리 구조에 대해 완전 매칭 가능한 사례가 더 존재하는지 점검하게 되는데 본 예에서는 MR190.05-2754-CASE#1/1가 사례 구조 요인 관점에서 완전 매칭되기 위해서는 MR193.09-986에 대한 구조 변화가 필요하므로 이미 확보된 논리 구조에 대해서는 더 이상의 사례가 존재하지 않는다. 이에 따라 Step 1-10를 수행하게 된다..

Step 1-10) 현재 매칭 리스트에 MR193.09-292-CASE#2/2가 존재하므로 Step 1-13을 수행하게 된다.

Step 1-13) 종료.

Step 2) 현재 매칭 리스트에는 유일하게 MR193.09-292-CASE#2/2밖에 없으므로 최적 유사 사례 선정 과정을 생략되고 이 결과가 사례 조정 및 유지 보수 대상 인식 과정을 전달된다.

이제 이렇게 유사 유지 보수 사례 추출 방법을 통해 추출 선정된 사례인 MR193.09-292-CASE#2/2를 이용하여 유지 보수 요구 MR193.09-986에 대한 유지 보수 대상을 인식하여 제시하고자 한다.

Step 1) 추출된 유사 사례 MR193.09-292-CASE#2/2의 정보로부터 다음의 네 개의 사례 결과 집합(Case-Result-Set)들이 호출되며 이들 각각은 다음과 같다.

{R193.09-292-CASE#2/2-REQ#1,
MR193.09-292-CASE#2/2-REQ#2,
MR193.09-292-CASE#2/2-REQ#3,
MR193.09-292-CASE#2/2-REQ#4}

이러한 사례 결과 집합으로부터 유지 보수 요구 MR193.09-986에 대한 초기 유지 보수 대상들의 집합이 다음과 같이 추출 생성되어 진다.

초기 유지 보수 대상 집합 =
{ CAM30JOJ 3210 BUHA CHNG ITEM MOVE 015600,
CAM30JOJ 3210 BUHA CHNG ITEM MOVE 015700,
CAM30JOJ 3210 BUHA CHNG ITEM MOVE 015800,
....
}

위의 예에서 결과로서 제시된 초기 유지 보수 대상 집합에 속하는 각 명령 절(Clause)의 수는 총 113개이다. 만일 위에 예에 대해 키워드에만 관련된 변수들을 바탕으로 단순 스트링 매칭(String Matching)을 통해 유지 보수 대상 명령 절들을 조사할 경우 관련 프로그램의 수는 약 100여 개에 달하게 되며 295개의 명령 절들이 추출되게 되나 이들 295개의 명령 절은 실질적으로 유지 보수 되어야 하는 명령 절들과는 아무런 직접적 관계를 갖지 못하는 있는 것으로 파악되었다. 이는 곧 유지 보수 담당자들이 본 연구에서 제안된 방법론을 이용한 유지 보수 지원이 없었다면 유지 보수 요구 MR193.09-986에 대한 유지 보수를 위해 초기

295개의 명령 절을 바탕으로 무려 13,649개의 명령 절들에 대한 분석 및 유지 보수 대상 탐색 작업을 수행해야만 한다는 것을 의미한다. 이러한 유지 보수 대상 탐색 과정에 투입되어야 하는 노력의 극적인 감소는 곧 유지 보수 업무의 파격적 효율성 향상을 가능케 할 수 있다.

Step 2) 선행 단계에서 얻어진 초기 유지 보수 대상 집합에 대해 MPI알고리즘을 적용함으로써 유지 보수 요구 MR193.09-986에 대한 완전성을 검증하게 되는데 이 경우 초기 유지 보수 대상만으로 완전성이 검증되게 된다. 만약 이때에도 사례 기반 유사 유지 보수 사례 추출 방법에 의한 초기 유지 보수 대상 정보 없이 시스템 전체에 대해 직접 MPI알고리즘을 적용했다면 1,121개의 초기 관련 명령 절을 바탕으로 13,649개의 명령 절에 대한 완전 탐색을 수행하여야만 하는데 MPI알고리즘 탐색 결과의 완전성과 제안한 유지 보수 대상 중에 실제 유지 보수 되었던 결과의 비중을 평가하는 적중률의 우수성은 경험적으로 입증되었음에도 불구하고 (Lee *et al.*, 1997), 사례 기반 유사 유지 보수 사례 추출 방법에 의한 초기 유지 보수 대상 정보를 이용하는 경우에 대비하여 상대적으로 엄청난 탐색 부담을 안게 된다. 한편 사례 기반 유사 유지 보수 사례 추출의 결과로 얻어진 단지 113개의 유지 보수 대상 집합에 대해서만 MPI알고리즘을 적용할 경우 유지 보수 대상의 완전성은 물론이고 탐색에 소요되는 계산상 또는 시간상의 효율성을 획기적으로 증대시킬 수 있게 된다.

Step 3) 마지막으로 이상의 단계를 통해 얻어진 최종 유지 보수 대상 집합을 유지 보수 담당자에게 제시하게 된다.

5.5 사례 기반 유지 보수 대상 인식을 통한 유지 보수 업무 지원 성과 분석

사례 기반 유지 보수 대상 인식 방법론의 실제 문제에의 적용에 있어서 그 경험적 타당성과 성과는 이미 한국 전력에서의 실제 정보 시스템에 대한 적용을 통해 입증된 바 있다. 한편 사례 기반 유지 보수 대상 인식 방법의 성과를 정량적으로 측정하기 위해 앞에서 사용한 초과 전력 사용량 요금 예를 포함하여 5개의 대표적인 실제 예들을 이용한 실험을 통하여 계량적 성과 자료가 수집되었다. 성과를 분석하기에 앞서 본 연구에서는 다음의 세 가지 관점에서 사례 기반 유지 보수 대상 인식 방법론을 평가하고자 한다. 첫번째 관점은 실제 유지 보수 되어야 할 대상중 사례 기반 유지 보수 대상 인식 방법론에 의해 제시된 유지 보수 대상에서 제외된 경우가 있는지의 여부이다. 이를 우리는 완전성(Completeness)이라고 정의하고자 한다. 둘째는 제시된 유지 보수 대상 중 실제로 유지 보수된 부분이 차지하고 있는 비율에 관련되며 이를 적중률(Hit Ratio)이라 정의하고자 한다. 마지막으로 완전한 유지 보수 대상을 추출하기 위해 탐색되었던 정보의 량(Volume)에 대한 상대적 효율성이며 이는 곧 계산상의 효율성(Computational Efficiency)을 의미한다. 아래의 <표 4>는 이들 5개의 예제에 대해 완전성, 적중률, 계산상의 효율성 측면에서 실험된 결과를 보여주고 있다.

<표 4>에서의 실험 결과에 따라서 우리는 우선 본 연구에서의 사례 기반 유지 보수 대상 인식 방법론이 이에 따라 제시된 유지 보수 대상을 실제 유지 보수 대상과 비교해 볼 때 완전하다는 것을 알 수 있다. <표 4>에서 완전성 항목이 모든 경우에 1의 값을 보이고 있다는 사실이 이를 경험적으로 확인시켜 주고 있다. 한편 (6)열에서의 적중률 항목은 평균적으로 1.60의 값을 나타내고 있으며 이는 본 연구에서 제안된 방법론을 이용할 경우 유지 보수 담당자가 제시된 유지 보수 대상 중 궁

극적으로 유지 보수해야 할 대상을 확인하는 과정에서 단지 38%의 추가적 탐색 부담만을 요구하게 된다는 것을 의미한다. 이러한 결과는 앞에서도 언급한 바와 같이 현재의 전체 유지 보수 시간 중 대부분의 시간을 유지 보수 대상 탐색에 소모하고 있는 실정에 비교해 볼 때 매우 긍정적인 결과임이 분명하며 유지 보수 담당자들이 대부분의 시간을 유지 보수 자체에만 투입할 수 있다는 것을 의미한다.

마지막으로 계산 효율성 차원에서 평가해 볼 때, 평균적으로 0.35%의 수치를 기록하고 있는데 이 수치의 의미는 사례 기반 유지 보수 대상 인식 방법을 적용하지 않고 순수하게 문자열 매칭이나 MPI알고리즘을 적용했을 경우 필요한 탐색 대상의 수에 대비하여 사례 기반 유지 보수 대상 인식 방법론을 적용할 경우 요구되는 탐색 대상 수의 백분율인데 결과적으로 같은 유지 보수 대상을 찾게 되더라도 이에 소모되는 탐색 작업의 양이 사례 기반 유지 보수 대상 인식 방법론을 이용하여 약 99.65%가 감소될 수 있다는 것을 보여 주고 있다. 이러한 획기적인 계산 효율성은 대상 시스템의 규모가 커지면 커질 수록 매우 큰 의미를 갖게 되며 일반적으로 정보 시스템들은 규모 면에서도 대부분 대형의 시스템이므로 사례 기반 유지 보수 대상 추출 방법의 상대적 우수성을 단적으로 나타내 주고 있다.

표 4. 사례 기반 유지 보수 대상 인식 방법의 성과 분석 결과

(1)	(2)	(3)	(4)	(5)	성과 측정 기준		
					완전성	(6)적중률	(7)계산 효율성
MR-01	28,246	25	22	9	1	2.44	0.09%
MR-02	4,808	28	15	14	1	1.07	0.58%
MR-03	13,649	12	3	3	1	1.00	0.09%
MR-04	3,560	29	18	9	1	2.00	0.81%
MR-05	14,294	28	25	17	1	1.47	0.20%
Average					1	1.60	0.35%

(1) 유지 보수 요구
(2) 완전 탐색 경우 탐색 대상 명령 절(Clause)의 수
(3) 사례 기반 유지 보수 대상 추출에 의한 초기 유지 보수 대상의 수
(4) MPI알고리즘을 초기 유지 보수 대상에 대한 사례 결과 보완 후 제안된 유지 보수 대상의 수
(5) 유지 보수 요구에 대해 실제 유지 보수된 명령 절의 수
(6) 적중률은 (4) ÷ (5)의 결과로 얻어진다.
(7) 계산 효율성은 (3) ÷ (2)의 결과로 얻어진다.

Ⅵ. 결 론

우리는 최근에 들어 매우 심각한 문제로 부각되고 있는 정보 시스템에 대한 총체적 유지 보수 지원 시스템인 METASOFT를 개발하였다(Lee et al., 1997). 이러한 배경 하에 본 연구에서는 METASOFT의 정보 시스템에 대한 접근 방법의

일환으로 과거 유지 보수 사례를 이용하여 특정 유지 보수 요구에 대한 유지 보수 대상 인식을 효과적이면서 동시에 효율적으로 수행할 수 있는 사례 기반 유지 보수 지원 방법론을 제안하였다. 이를 달성하기 위해 우리는 우선 METASOFT에서의 정보 시스템 설계 지식 표현 방법을 바탕으로 유지 보수 사례의 표현 방법을 고안하였고, 둘째 유지 보수 업무의 특성을 분석하여 유지 보수 사례간의 유사성을 정의하고 그 평가 방법을 제시하였다. 마지막으로 사례 기반 유지 보수 대상 인식 방법론 제안하였는데 이는 유지 보수 사례 표현 방법과 유사성 평가 방법 및 MPI알고리즘을 기반으로 설계 개발되었다. 한편 이러한 사례 기반 유지 보수 지원 방법론은 METASOFT에 적용되어 하부 시스템으로 구축되어 본 연구에서 제안된 방법론들을 사용자가 체계적으로 이용할 수 있도록 지원되고 있다.

특히 본 연구에서 제안된 사례 기반 유지 보수 대상 인식 방법론은 이론적으로나 실험적으로도 그 완전성 및 성과를 검증할 수 있었으며, 한국 전력의 COBOL과 IMS 데이터베이스를 사용하는 정보 시스템에 적용되어 유지 보수 실무자들에 의해 그 경험적 타당성이 검증되었다. METASOFT와 함께 본 연구가 갖는 의의는 먼저 기존의 정보 시스템을 완전히 재개발하지 않고도 효과적으로 유지 보수할 수 있는 한 접근 방법을 제시한데 있다. 또한 이러한 방법을 통해 본 연구에서 관찰된 유지 보수 지원에의 성과가 다른 정보 시스템이나 보다 복합적 구조를 갖는 시스템에 대해서도 성공적 적용이 가능할 것으로 기대되며 이로 인해 유지 보수 업무에 있어서의 업무량 감소와 유지 보수의 효율성을 동시에 제고시킬 수 있다고 사료된다. 따라서 이는 현재 많은 어려움을 겪고 있는 많은 정보 시스템들에 대한 유지 보수 문제에 대한 하나의 해결책으로서 공헌할 수 있게 될 것이다.

Acknowledgement: 이 연구는 METASOFT 연구를 수행한 대우정보시스템의 이병엽씨의 도움을 받아, 한국전력과 전북대학교 신진교수지원연구비에 의해 수행되었습니다.

References

1. Aiken, P., A. Muntz, and R. Richards, "DoD Legacy Systems: Reverse Engineering Data Requirements," *Communication of ACM*, Vol. 37, No. 5 (1994), 27-41.
2. Bennett, K., "Legacy Systems: Coping With Success," *IEEE Software, Jan.* (1995), 19-23.
3. Biggerstaff, T. J., "Design Recovery for Maintenance and Reuse," *IEEE Computer*, July (1989), 36-49.
4. Burson, S., G. B. Kotik, and L. Z. Markosian, "A Program Transformation Approach to Automating Software Reengineering," *Proc. COMSAC*, IEEE Society Press, Los Alamitos, Calif., (1990), 314-322.
5. Fjeldstrad, R. K. and W. T. Hamlen, "Application Program Maintenance Study - A Report to Our Respondents," in G. Parikh and N. Zvegintzov (Eds.), *Tutorial on Software Maintenance*, IEEE, (1983).
6. Housier, P. A. and M. G. Pieszkoch, "Using Function Abstraction to Understand Program Behavior," *IEEE Software*,

Jan. (1990), 55-63.

7. Kolodner, J. L., *Case Based Reasoning*, Morgan Kaufmann Publishers, Inc., 1993.
8. Kozaczynski, W., S. Letovsky, and J. Q. Ning, "A Knowledge-Based Approach to Software System Understanding," *Proc. 6th Annual Knowledge-Based Software Conference*, (1991), 162-170.
9. Lee, B. Y., W. Kim and J. K. Lee, "A Knowledge Based Maintenance of Legacy Systems: METASOFT," *Expert Systems with Applications*, Vol. 12, No. 4 (1997), 483-496.
10. Lee, J. K. and Y. U. Song, *UNIK User Manual*, Intelligent Information System Laboratory in Korea Advanced Institute of Science and Technology, 1994.
11. Lee, J. K., W. Kim and B. Y. Lee, *Software Development Supporting Expert System*, Intelligent Information System Laboratory in Korea Advanced Institute of Science and Technology, 1995.
12. Lientz, B. and E. Swanson, *Software Maintenance Management*, Addition-Wesley, 1980.
13. Lirov, Y., "Computer Aided Software Engineering of Expert Systems," *Expert Systems with Applications*, Vol. 2 (1991), 333-343.
14. Markosian, L., P. Newcomb, R. Brand, S. Burson and T. Kitzmiller, "Using an Enabling Technology to Reengineer Legacy Systems," *Communication of ACM*, Vol. 37, No. 5, May (1994), 58-70.
15. Minsky, M., "A Framework for Representing Knowledge," in R. J. Brachman and H. J. Levesque, Eds., *Reading in Knowledge Representation*, (1985).
16. Moulin, B. and D. Rousseau, "Automated Knowledge Acquisition from Regulatory Texts," *IEEE Expert*, October (1992), 27-35.
17. Ning, J. Q., A. Engberts and W. Kozaczynski, "Automated Support for Legacy Code Understanding," *Communication of ACM*, Vol. 37, No. 5, May (1994), 50-57.
18. Premerlani, W. J. and M. R. Blaha, "An Approach for Reverse Engineering of Relational Databases," *Communication of ACM*, Vol. 37, No. 5, May (1994), 42-49.
19. Sull, W. and R. L. Kashyap, "A Self-organizing Knowledge Representation Scheme for Extensible Heterogeneous Information Environment," *IEEE tran. on Kowledge and Data engineering*, Vol. 4, No. 2, April (1992).
20. Williamson, M., "Software Engineering for Redevelopment," *CASE Strategy*, Vol. 3, No. 9, Aug. (1991).