# 도메인 분석정보의 재사용을 위한 처리기의 설계 및 구현

김 지 홍[†] 송 영 재[††]


## 요 약

도메인 분석은 새로운 응용시 손쉽게 재사용할 수 있도록 유사 시스템의 공통점을 식별하는 활동이다. 대부분의 도메인 분석의 결과는 통일된 방식이 없이 도식적으로 표현되고 수작업으로 재사용되고 있어서 분석명세의 재사용성이 작다. 도메인 분석정보도 수작업 대신에 잘 정의된 구문을 갖는 도메인 분석명세 언어로 표현하여 명세의 적절성 검사와 재사용의 연산을 통해 도메인 분석명세 재사용의 자동화가 가능하다. 본 연구에서는 도메인 분석의 결과를 도메인 분석명세 언어로 표현하여 재사용할 수 있도록 도메인 분석명세 재사용 처리기를 설계 구현하고 대여 도메인에 적용하여 새로운 명세의 인스탄스를 얻을 수 있었다. 아울러 종전의 데이타 흐름도 지원 도구를 통한 재사용 방식과 분석정보 처리기를 통한 재사용 시간을 비교하였으며 두 가지 방식은 서로의 단점을 보완하여 재사용성을 향상시킬 수 있음을 발견하였다.


# Design and Implementation of a Processor for the Reuse of Domain Analysis Information

Ji-Hong Kim[†] Yong-Jae Song[††]


## ABSTRACT

Domain Analysis is an activity to identify commonalities and variabilities within similar application areas in order to reuse analyzed information easily in new software construction. Most of domain analysis output is represented by various diagrams without common standard, and its manual reuses result in low reusability. Domain analysis language can be used to represent domain analysis information and make it possible to automate reuse and test the specifications. In this paper we designed and implemented a processor to reuse domain analysis information represented by domain analysis language and applied our approach to a rental domain and got new specification instances. In addition, we compared reuse of a data flow diagramming tool with reuse of a domain information processor and found combining of each approach can increase the reusability of both.


## 1. 서 론

도메인 분석은 비슷한 소프트웨어들 간의 공통점과 차이점을 식별하여 관련된 응용의 개발시 기존의 분석명세를 손쉽게 재사용하려는 접근 방식으로 체계적 재사용의 중요한 기술적 요소로 인식되고 있다[3, 4, 12, 13]. 그 동안 도메인 분석에 관한 연구는 주로 분석의 절차에 집중되어 많은 발전을 가져 왔으나 분석된 정보는 통일된 표준이 없이 다양한 도식적 방법으로 표현되어 왔고 주로 수작업을 통해 이해되고 재사용되어 재사용의 자동화가 부족하였다. 현재까지 연구 및 시도된 여러 도메인 분석 방식 가운데 명세에 관련된 것으로는 Neighbors연구, Prieto-Diaz방식, 특성 중심의 도메인 분석 방식, 도메인 공학 방식 등이 있다. Neighbors[5, 7, 8, 9]는 Draco 프로젝트의 연구를 통해 소프트웨어 시스템은 재사용 컴포넌트를 통합하여 생성할 수 있음을 입증하고 도메인 분석이란 용어를 처음 사용하였다.


† 정 회 원 : 경원대학교 전자계산학과 조교수
†† 정 회 원 : 경희대학교 전자계산공학과 교수

논문접수 : 1995년 6월 9일, 심사완료 : 1995년 7월 21일.

Draco에서는 컴포넌트의 변환 및 정련을 통해 실행 가능한 목적 시스템을 얻기 위해 도메인 분석과 도메인 언어의 사용 및 이를 위한 도메인 파서, 도메인 프리티프린터, 원시 프로그램의 변환, 도메인 컴포넌트를 소개하였다. 이 연구에서는 SIMAL이라는 도메인 언어를 포함하여 도메인 언어를 통한 분석의 표현을 소개하였으나 언어의 구조에 관한 구체적인 언급은 없었다. Prieto-Diaz[9, 10, 11]는 재사용 라이브러리를 갖기 위한 3가지 활동으로 1) 도메인 분석 이전 활동, 2) 도메인 분석 활동, 3) 도메인 분석 이후 활동으로 구분하고 도메인 분석의 3단계를 제안하였다. Prieto-Diaz의 연구는 종래의 도메인 분석 경험을 중심으로한 도메인 분석 절차의 설명과 달리 데이타 흐름도를 이용하여 도메인 분석의 절차를 체계적으로 설명하고 발전시켰다. 도메인 분석명세의 표현에 관한 구체적인 언급은 없었으나 도메인 모델을 표현하는데 프레임, 의미망, 모듈 연결 언어의 가능성을 소개하였다. 특성 중심 도메인 분석(FODA)[6]은 1) 배경 분석, 2) 도메인 모델링, 3) 구조 모델링의 3단계로 나누었다. FODA에서는 이전의 다른 도메인 분석 연구와 달리 각 단계의 모델링을 위해 특성, 엔티티, 속성, 관계의 표현을 위해 간단한 언어 형태를 갖는 5가지의 양식(FORM)을 지원하였다. 양식은 구체적으로 통일된 표현 방법을 간편하게 제공할 수 있는 장점을 갖으나 제약 사항의 표현이 용이하지 않으며 부분적 자연어의 표현으로 전산 처리성이 떨어진다. 도메인 공학[1, 2]은 공학적 관점에서 1) 도메인 분석, 2) 도메인 모델링, 3) 하부구조의 구축으로 구분하였으며, 요구 모델 언어(RML)를 바탕으로 하여 BNF (Backus Naur Form) 형식으로 MoDL명세 언어를 제안하였다[2]. MoDL은 모델과 객체, 그리고 이들간의 관계 표현은 잘 정의되어 있으나 명세의 검사, 재사용 연산의 시설은 지원이 안된다.

즉, 그 동안의 도메인 분석에 관한 연구는 주로 절차에 집중되어 많은 발전을 가져왔으며, 도메인 분석의 표현은 프레임, 의미망, 상태 전이도, 데이터 흐름도, 엔티티 관계도와 같은 도식적인 방식이 주로 사용되었다. 도식적인 방식은 표현성은 좋으나 분석정보의 직접 재사용은 어렵고, 전산 처리성이 부족하여 수작업을 통한 재사용으로 행하여 졌다. FORM과 MoDL은 도메인 분석정보의 언어적 표현을 제시하였으나 명세 표현에만 국한되었으며 완전성, 중복성, 일치성과 같은 오류 검사의 기능과 합성, 추출과 같은 재사용 연산의 지원에 관한 고려는 부족하였다. 체계적인 도메인 분석의 절차도 중요하나 분석된 결과를 수작업으로 재사용하는 대신에 자동적 처리를 통해 도메인 분석정보의 재사용성 향상도 필요하다.

본 논문에서는 이러한 문제의 개선을 위해 잘 정의된 구문과 연산을 갖는 분석명세 언어로 도메인 분석 결과를 표현하여 이들 명세의 오류를 검사하고 재사용 처리하여 새로운 명세의 인스탄스를 생성할 수 있는 도메인 분석정보 처리기를 구현하여 도메인 분석명세 재사용의 자동화를 향상시키고자 한다.

## 2. 관련 개념

### 2.1 도메인 분석정보의 재사용

도메인 분석정보의 재사용을 위해서는 1) 재사용을 위한 개발의 단계와 2) 재사용을 통한 개발의 단계로 구분된다. 재사용을 위한 개발의 단계에서는 도메인 분석정보를 재사용할 수 있도록 전산 처리가 용이한 도메인 분석명세 언어로 분석명세를 작성한다. 이 단계에서는 작성된 분석명세가 재사용하기에 적절한 구문으로 표현되었는가를 검사하고 올바로 표현된 도메인 분석명세는 나중에 재사용되기 위해 라이브러리에 저장된다. 재사용을 통한 개발 단계에서는 새로운 소프트웨어의 개발 요구시 매번 처음부터 새로운 요구 분석을 하지 않고 라이브러리에서 비슷한 도메인 분석명세를 찾아 기존의 분석명세를 조금 바꾸어 재사용하여 새로운 분석명세를 얻는다. 도메인 분석명세 언어를 통한 도메인 분석정보의 재사용을 위해서는 다음과 같은 6단계가 필요하다.

1) 도메인 분석 단계, 2) 명세 표현 단계, 3) 명세 검사 단계, 4) 재사용 명세 검색 단계, 5) 재사용 표현 단계, 6) 명세의 인스탄스 생성 단계

제1단계에서는 여러 도메인 분석 방식에 의해 분석을 하여 해당 도메인에서 공통점과 차이점을 식별한다. 제2단계에서는 도메인 분석정보를 전산 처리 가능한 도메인 분석명세 언어로 표현하며 3단계에서는 재사용 분석명세 자산의 생성을 위해 도메인 분석명세 언어로 표현된 명세 프로그램을 검사한다. 4단계에서는 질의어나 facet 분류 방식을 통해 새로운 응용에 비슷한 도메인 분석명세를 검색한다. 5단계에서는 해당 도메인 분석명세를 바탕으로 하여 새로운 요구에 맞도록 재사용 명령의 프로그램을 작성하며 6단계에서는 재사용 프로그램을 번역, 실행하여 새로운 응용에 맞는 분석명세의 인스탄스를 생성한다.

### 2.2 도메인 분석명세 언어

재사용 도메인 분석명세 언어는 명세의 표현을 위한 단계 2와 재사용의 표현을 위한 단계 5를 위하여 기존의 프로그래밍 언어와 데이터 플로우, 엔티티 관계, 상태 전이의 표기를 기반으로 설계된 형식 언어이다[14].

코딩을 위해 프로그램, 서브루틴, 모듈 등의 단위가 있듯이 도메인 명세 언어에서는 도메인, 파트, 객체의 3가지 단위를 통해 명세와 재사용을 위한 모듈화를 지원한다. 도메인은 특정 응용 분야를 의미하는 개념적 단위로서 고유한 이름을 갖고 최상위의 명세 재사용을 지원한다. 두 번째 단위인 파트는 논리적 또는 기능적으로 비슷한 여러 객체들로 구성된 합성적 객체나 클래스와 같이 소규모적 재사용 명세의 단위이다. 세 번째 단위인 객체는 더이상 나누어질 수 없는 최소의 재사용 명세 단위이다. 도메인 단위의 명세는 대규모의 재사용을, 객체 단위의 명세는 소규모의 재사용을, 파트 단위의 정보는 중간의 실질적 재사용을 지원한다.

재사용 도메인 분석명세 언어는 검사와 표현의 간편성을 위해 여러 형을 제공하고 있는데 대표적인 형으로는 도메인형, 파트형, 객체형이 있다. 도메인형은 context, feature, architecture, 사용자 정의형으로 세분될 수 있으며, 파트형은 std, dfd, erd형으로, 객체형은 process, state, object형으로 세분될 수 있다. 각각의 재사용 명세 단위는 머리, 선언, 몸체의 3 부분으로 나누어져 표현되는데 명세 프로그램의 머리 부분은 명세의 이름과 재사용 단위 및 인터페이스 정보를 갖고 외부적 식별과 인터페이스 정보를 제공한다. 선언 부분은 명세에 필요한 각종 변수 및 형의 정보를 나타내어 몸체 부분에서의 각종 형검사를 지원한다. 명세 프로그램의 몸체 부분은 명세의 구체적인 내용을 기술하는데, 내용은 형태에 따라 데이터 흐름문, 상태 전이문, 엔티티 관계문, 조건문 등으로 표현된다. 모든 문장은 문장 분리자 ';'로 끝나는데, 상태 전이문은 X = Y / z; 형태로, X, Y는 상태 변수이며, z 는 데이터 변수로서 상태 Y에서 입력 또는 사건 데이터가 z일 때 다음 상태 X로 전이를 나타낸다. 데이터 흐름문도 X = Y / z; 형태로 상태 전이문과 비슷한데 X, Y는 프로세스 변수, z 는 데이터 변수로서 프로세스 Y에서 흐름 또는 사건을 유발시키는 데이터가 z일 때 다음 프로세스 X로의 흐름을 나타낸다. 엔티티 관계문은 R = E1(m) : E2(n); 형태로 표현되는데 E1, E2는 엔티티이며 R은 이들간의 관계를 나타내고 m, n은 엔티티간 대응의 차수를 나타낸다.

집단화문은 X = Y + Z; 형태로 표현되는데 X, Y, Z는 내용에 따라 도메인, 파트, 객체형의 변수로서 X 는 Y와 Z로 구성됨을 의미한다. 이외에 순차, 조건, 반복 및 치환의 기본문과 재사용 명세 단위의 선후 조건문을 사용할 수 있다.

### 2.3 도메인 분석정보의 재사용 연산 및 처리기

기존의 분석된 도메인 분석명세는 응용에 따라 1) 그대로, 2) 삭제, 3) 추가, 4) 계승, 5) 실체화, 6) 추출의 6가지 연산을 통하여 재사용을 할 수 있다. 그대로의 연산은 응용의 요구와 분석명세가 같거나 기존 도메인 분석 내용에 맞추어 개발을 하는 이른바 변경없이 분석된 대로 재사용하는 경우이다. 삭제 연산은 기존의 도메인 분석명세에서 불필요한 항목을 제거하는 연산으로 특정 형 또는 상태문, 엔티티 관계, 프로세스문 등의 삭제가 가능하다. 삭제의 연산은 해당 명세 항목과 관련된 항목의 삭제도 수반될 수 있다. 추가는 기존의 재사용 분석명세에서 특정 변수나

문장을 추가하여 새로운 도메인 분석명세의 인스탄스를 생성하는 연산을 말한다. 계승의 연산은 새로운 응용을 위해 기존의 오브젝트를 기반으로 하여 파생 인스탄스의 생성을 의미하며, 실체화 연산은 분석명세에 특정형이나 변수를 매개변수로 전달하여 명세의 고객화를 통한 재사용을 의미한다. 특성의 추출 연산은 해당 도메인 명세에서 최소로 필요한 특성의 추출을 의미한다. 도메인 분석명세 언어에서 추가와 계승은 +, 삭제는 -, 추출은 #연산자를 제공한다.

이러한 재사용 연산의 실행을 위해서는 재사용 처리기가 필요하다. 도메인 분석명세 언어를 바탕으로한 재사용 처리기는 명세 및 재사용 명령의 적절성 검사와 재사용 연산의 지원을 기본으로 다음과 같은 특성을 갖는다.

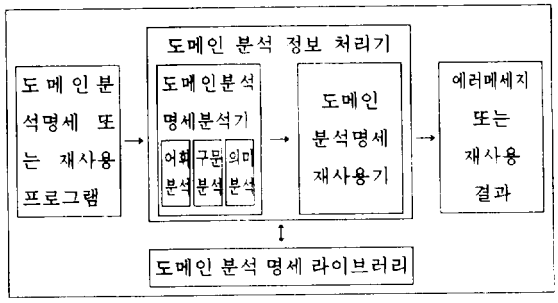
1) 명세의 적절성 검사, 2) 재사용 명령의 적절성 검사, 3) 재사용 명령의 실행

첫 번째 특성인 명세의 적절성 검사는 재사용 가능한 분석명세 자산을 만들기 위한 검사로서 1) 미 정의 항목의 참조, 2) 연관성의 정의 누락, 3) 중복 표현, 4) 입력상 철자의 오류, 5) 명세의 의미적 검사를 실행한다. 재사용 명령의 적절성 검사는 도메인 분석명세의 재사용 연산을 기술한 재사용 프로그램의 검사로서 1) 미 정의 항목의 재사용, 2) 미 정의 항목의 추가, 3) 미 정의 항목의 삭제, 4) 미 정의 연산의 사용을 검사한다. 재사용 명령 실행의 특성은 명세의 적절성 검사가 끝난 도메인 분석명세를 재사용 하기 위해 작성하고 재사용 명령의 적절성 검사가 끝난 재사용 프로그램을 기술된 대로 6가지의 재사용 연산을 해석하고 실행하여 손쉽게 새로운 명세의 인스탄스를 생성해 준다.

## 3. 시스템 설계 및 구현

### 3.1 시스템 구성

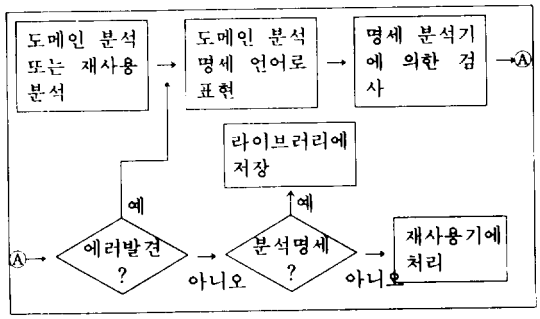
본 장에서는 도메인 분석명세 언어를 통한 재사용 단계 3과 6을 위해 처리기를 설계 구현한다. 본 논문에서 설계한 도메인 분석정보 처리기는 (그림 1)과 같이 도메인 분석명세 분석기와 도메인 분석명세 재사용기로 구성된다. 명세 분석기는 도메인 분석정보가 재사용하기에 적절한 구문으로 표현되었는가를 검사해주며, 재사용기는 분석명세의 재사용 연산을 해석, 실행하여 새로운 명세의 인스탄스를 생성한다.




(그림 1) 도메인 분석정보 처리기 구조
(Fig. 1) Structure of Domain Analysis Information Processor

### 3.2 도메인 분석명세 분석기

도메인 분석명세 분석기는 재사용 도메인 분석명세의 자산을 만들기 위해 분석가가 도메인 분석명세 언어로 작성한 분석명세 또는 이들의 재사용 처리를 기술한 재사용 프로그램을 입력으로 받아 재 사용에 용이하도록 도메인 분석명세 언어로 표현되었는가를 검사하여 오류가 발견되면 메시지를 출력한다. 올바로 표현된 분석명세는 재사용 가능한 자산으로 간주하여 라이브러리에 저장하고 올바로 표현된 재사용 프로그램은 재사용기로 보낸다. 명세 분석기의 배경은 (그림 2)와 같다.




(그림 2) 명세 분석기의 배경
(Fig. 2) Context of Specification Analyzer

도메인 분석명세는 2.3절에 소개된 일관성, 완전성, 중복성 검사를 통해 명세에 오류가 없어야 재사용할 수 있다. 이를 위해서는 구문상, 의미상 오류 검사가 필요하므로, 도메인 분석명세 분석기는 (그림 3)과 같은 구조를 갖는 심볼 테이블에 토큰을 저장, 참조하면서 어휘 분석, 구문 분석, 의미 분석의 3단계를 통해 처리된다.

```
typedef struct symbol table
{
    char token[MAXSIZE];
    int  token type;
    int  attribute;
    struct symbol table *next;
} symbol table;
```

(그림 3) 심볼 테이블의 주요 자료 구조
(Fig. 3) Major Data Structure of Symbol Table

어휘 분석 단계에서는 입력 스트림에서 공백 문자를 제거하고, 연산자, 구두점을 중심으로 나누어 재사용 명세의 구문 분석을 위해 구문적 기본 단위인 토큰을 생성한다. 특히 키워드와 사용자 정의 변수는 심볼 테이블에 저장하고, 필요할 때 검색한다.

구문 분석 단계에서는 명세의 재사용이 가능하도록 정해진 문법에 따라 작성되었는지 여부를 결정하기 위해 명세의 3부분을 차례로 검사한다. 머리 부분은 구문에 따라 형정보로 시작해서 프로그램 이름이 ':'로 끝나는가를 검사하며, 선언 부분은 키워드 'declare'로 시작해서 변수의 형, 속성, 이름에 뒤이어 문장 분리자 ';'로 끝나는가를 검사하여 구문 분석 단계에 등록된 심볼 테이블에 있는 해당 변수의 형, 속성 필드를 갱신한다. 몸체 부분은 주어진 명세의 내용에 따라 도메인 특성문, 데이터 흐름문, 상태 전이문, 엔티티 관계문 등이 적절히 표현되었는가를 검사한다.

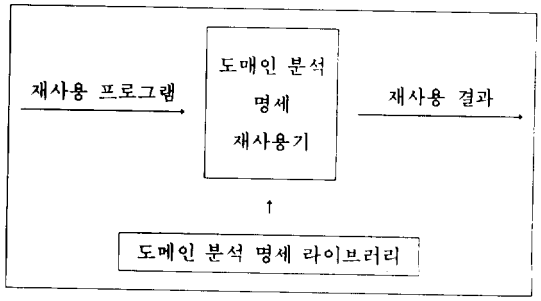
의미 분석 단계에서는 문장의 의미상 오류를 검사하는데, 어휘 분석 단계와 구문 분석 단계에 구축된 심볼 테이블의 정보를 바탕으로 주어진 연산자에 적절한 형의 피연산자가 사용되었는가의 여부를 검사한다. 구문적으로 올바른 문장이라도 변수의 위치에 따라 적절한 형을 가져야 한다. 예를 들어 도메인 특성 문장은 lhs = rhs1 + rhs2; 형태의 문장으로 표현되는데 명세 프로그램에서 특성 명세는 집단화를 의미하므로 lhs의 재사용 단위는 rhs1 이나 rhs2의 단위 보다 커야 하며, 데이터 흐름문은 lhs = rhs1/rhs2의 형태로 표현되는데 프로세스 rhs1에서 데이터 rhs2에 의해 프로세스 lhs로의 흐름을 의미하므로 lhs1과 rhs1은 반드시 프로세스형이어야 하며 lhs2는 데이터형 변수이어야 한다. 데이타 흐름문의 의미 분석을 위한 주요 알고리즘은 (그림 4)와 같다.

```
// lhs1 = rhs1 operator rhs2
get left hand side variable lhs1;
get right hand side variable rhs1, rhs2
      and operator;
if (operator = '/') then begin
    if (lhs1.type and rhs1.type is not same)
        return(error);
    if (rhs2.type is not data) return(error);
end
```

(그림 4) 의미 분석 알고리즘 (데이타 흐름문)
(Fig. 4) Algorithm of Semantic Analysis (data flow statement)

### 3.3 도메인 분석명세 재사용기

도메인 분석명세 재사용기는 명세 언어를 바탕으로한 재사용의 마지막 단계를 위해 (그림 5)와 같이 재사용 프로그램을 입력으로 받아 라이브러리에 있는 기존의 분석명세의 내용을 합성, 추출, 변경하여 새로운 응용 요구에 맞는 도메인 분석명세 인스탄스를 생성하는 인터프리터이다.




(그림 5) 도메인 분석명세 재사용기
(Fig. 5) Reuse Processor for Domain Analysis Specification

재사용기는 사용자가 작성한 재사용 프로그램의 첫 번째 키워드가 재사용을 의미하는 reuse인가의 확인을 포함하여 구문 검사를 먼저 하고, 에러가 없으면 재사용 문장을 번역 실행한다. 재사용 프로그램의 머리 부분 처리에서는 재사용 단위와 프로그램 이름을 식별하고, 선언 부분에서는 재사용될 명세와 새로 추가, 삭제, 생성될 항목의 이름과 형을 식별하며 연산에 필요한 명세 화일을 개방한다. 실질적인 번역은 몸체 부분에서 이루어지는데 도메인 특성문, 데이터 흐름문, 상태 전이문, 엔티티 관계문 등의 문장에 대하여 선언 부분에 언급된 정보를 기반으로 삽입, 삭제, 변경의 재사용 연산을 행한다. 몸체 부분의 해석을 위한 주요 알고리즘은 (그림 6)과 같다.

```
interpret body:
    determine and read reuse item;
    determine add item;
    determine delete item;
    new instance = reuse item +
          add item - delete item;
end
```

(그림 6) 몸체 부분 해석의 알고리즘
(Fig. 6) Algorithm of interpreting body section

### 3.4 도메인 분석명세 재사용기의 예

도메인 수준의 명세는 구조, 특성, 배경의 관점으로 표현할 수 있는데, 특성의 관점에서 표현된 도메인 분석명세를 재사용하여 새로운 인스탄스를 만들 수 있다. 예를 들어 무료로 책을 빌려주는 학교 도서관의 대출자 관리 프로그램은 기존의 대여 도메인 특성 rental manager에서 금전 부분을 삭제하여 무료 도서 대여를 위한 명세의 인스탄스를 얻을 수 있다. 재사용 프로그램 reuse rental.dal은 (그림 7)과 같다.

```
reuse feature reuse rental:
declare
  feature rental manager;
  feature new new rental;
  part moneytary;
body
  new rental  =  rental manager  -
        moneytary;
end
```

(그림 7) 특성 명세를 위한 재사용 프로그램
(Fig. 7)Reuse Program for Feature Specification

재사용기는 먼저 프로그램의 구문을 분석하고 구문상 오류가 없으면 재사용을 위한 해석을 실행한다. 먼저 프로그램 첫 줄인 reuse feature reuse rental;의 구문 검사를 통해 특성(feature) 도메인의 재사용(reuse) 프로그램 reuse rental을 식별하고 해석을 시작한다. 선언 부분의 feature rental manager;을 통해 재사용되는 도메인 특성 명세는 rental manager임을 식별하고 라이브러리에 있는 도메인 분석명세 프로그램 가운데 rental manager.dal을 재사용하기위해 개방한다. 선언의 두번째줄 feature new new rental;은 new라는 속성의 선언을 통해 새로 생성될 인스탄스를 식별하고, new rental.dal을 개방한다. 마지막 선언 part moneytary;를 읽고 파트 moneytary가 재사용 연산에 관련된 항목인가를 결정하고, 재사용 명세 rental manager.dal에 moneytary의 선언 여부를 검색한다. 만일 moneytary가 도메인 특성 rental manager에 존재하지 않으면 없는 항목에 관한 재사용 선언으로서 에러로 취급된다.

몸체 부분의 new rental=rental manager-moneytary;를 분석하고 기존의 도메인 특성 명세 rental manager에서 파트 moneytary와 이에 종속된 항목을 함께 삭제하여 새로운 인스탄스 new rental을 생성한다.

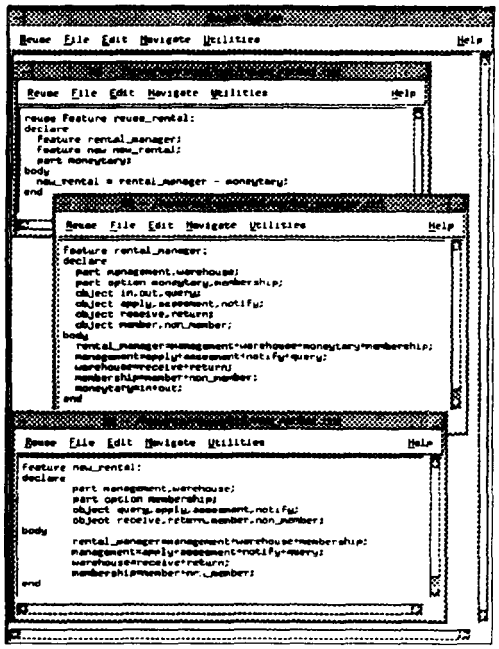
## 4. 적용 결과 및 평가

설계 구현된 도메인 분석명세 언어의 분석기와 재사용기의 타당성 시험을 위해 SPARC Station20의 Solaris 1.x에서 OSF/Motif를 이용하여 프로토타입을 구현하였다. 프로토타입은 재사용 도메인 분석명세의 재사용 단위인 도메인, 파트, 객체의 재사용성을 실험하였다.

### 4.1 도메인 특성의 재사용 특성의 재사용

실험을 위해 3.4절에 소개된 특성 명세 재사용 프로그램 reuse rental을 입력 화일로 사용하였

으며 (그림 8)의 상단부에 나타나 있다. 재사용의 자산이 되는 명세 프로그램 rental manager는 (그림 8)의 중간 부분에 나타나 있다. 재사용기의 실행 결과로 생성된 new rental은 (그림 8)의 하단부에 나타나 있다.



(그림 8) 특성 명세의 재사용
(Fig. 8) Reuse of Feature Specification

재사용 연산에 따라 기존 도메인 특성 명세에서 moneytary의 삭제는 이와 관련된 in과 out도 함께 생략되어 새로운 도메인 특성 인스탄스 new rental을 얻을 수 있었다.

### 4.2 비교 테스트

설계된 도메인 분석정보 처리기에 의한 재사용 방식과 종전의 도식적 방식인 데이타 흐름 지원 도구에 의한 재사용의 비교를 위해 동일한 문제를 각각의 방식으로 실험하였다. 비교 테스트는 기존의 분석명세를 1) 추가 연산을 통한 재사용과 2) 삭제 연산을 통한 재사용 처리로 나누었다.

#### 4.2.1 추가를 통한 재사용 시간의 비교

실험은 두개의 그룹으로 나누어 추가의 연산을 통한 재사용 방식으로 새로운 도메인 분석명세 인스탄스의 생성을 위해 표현과 검사, 처리 시간을 문제의 크기에 따라 비교하였다. 그룹 I은 종전의 수작업 방식에 의한 재사용을 대변하는 그룹으로 Teamwork/SA에서는 수작업으로 데이타 흐름도의 도식적 방식으로 재사용 처리하고, 명세의 오류를 검사하여 새로운 인스탄스를 생성하였다. 그룹 II는 동일한 문제를 재사용 처리기에 명세 언어로 재사용 표현하고 오류 검사를 행한 후 새로운 인스탄스를 자동 생성하였다. 오류 검사의 기능은 도식적인 경우 Teamwork/SA에 내장된 도식적 표현의 검사의 기능을 사용하였고 명세 언어의 경우는 재사용 처리기에 내장된 구문 및 의미 검사기의 기능을 이용하였다. 추가 연산을 통한 재사용 처리 시간 비교 실험의 결과는 〈표 1〉에 나타나 있으며 Teamwork/SA에 의한 결과와 비교한 도메인 분석정보 처리기의 재사용 처리 시간 향상율은 (그림 9)의 향상율 그래프에 나타나 있다.

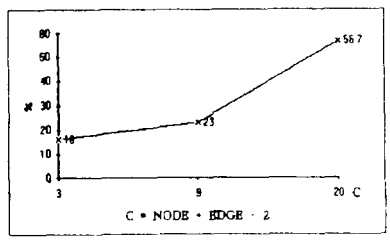
추가를 통한 도메인 분석정보의 재사용시에는 그룹 II의 도메인 분석정보 처리기 방식이 그룹 I의 도식적 방식의 재사용 처리 시간보다 평균 약 32% 빨랐으며 문제의 복잡도(C)와 비례적으로 증가하였다.

〈표 1〉 재사용 처리 시간의 결과(추가)
〈Table 1〉 Result of reuse process time(Insertion)

단위: 초(sec)

| 복잡도 | GROUP I | GROUP II |
|---|---|---|
| 3 | 108.4 | 93 |
| 9 | 315.8 | 256.6 |
| 20 | 721.25 | 460 |

복잡도 : NODE-EDGE+2




(그림 9)재사용 처리기의 향상율 그래프 (추가)
(Fig. 9) Improvement rate graph of Reuse Processor (Insertion)

#### 4.2.2 삭제를 통한 재사용 시간의 비교

새로운 도메인 분석명세를 얻기 위하여 삭제를 통한 두 가지 방식의 재사용을 비교 실험하였다. 그룹 I은 Teamwork/SA에서 수작업으로 분석정보의 재사용 처리를 표현하고 명세의 오류를 검사 한 후 도식적 형태의 새로운 인스탄스를 생성하였다. 그룹 II는 동일한 문제를 도메인 분석명세 언어로 재사용 처리를 표현하고 처리기로 오류 검사를 한후에 자동으로 새로운 도메인 분석명세의 인스탄스를 생성하였다. 삭제를 통한 재사용의 경우 Teamwork/SA는 도식적인 방식이므로 삭제하고자 하는 객체들의 그룹화 여부와 위치에 따라 실행 결과에 차이가 있으므로, 연산의 평균 시간 측정을 위해 모든 정보 항목을 전체적으로 한번에 삭제하는 최적 조건과 모든 정보 항목을 각각 삭제하는 최저 조건으로 나누어 실험을 행하였다. 삭제 항목의 위치 변경에 따른 결과는 〈표 2〉에 나타나 있다.

〈표 2〉 Teamwork에서 재사용 처리 시간 결과(삭제)
〈Table 2〉 Result of reuse process time in Teamwork/SA(Deletion)

단위 : 초(sec)

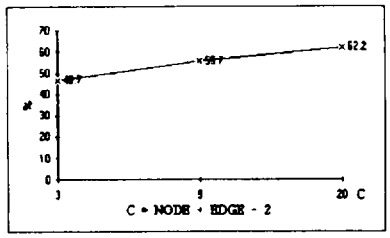
| 복잡도 | 전체 삭제 | 각각 삭제 | 평 균 |
|---|---|---|---|
| 3 | 17 | 32 | 24.5 |
| 9 | 23.5 | 39 | 31 |
| 20 | 31 | 65.5 | 48.2 |

도식적 방식에서 삭제를 통한 재사용의 방식은 항목의 그룹화와 위치에 따라 평균 21.5초의 차이가 나타났다. 아울러 도식적 방식과 언어적 방식에서 삭제 연산을 통한 재사용의 시간 비교 실험의 결과는 〈표 3〉에 나타나 있으며 분석정보 처리기에 의한 결과와 비교한 Teamwork/SA의 재사용 처리 시간 향상율은 (그림 10)의 그래프에 나타나 있다.

〈표 3〉 재사용 처리 시간의 결과(삭제)
〈Table 3〉 Result of reuse process time(deletion)

단위 : 초(sec)

| 복잡도 | GROUP I | GROUP II |
|---|---|---|
| 3 | 24.5 | 46 |
| 9 | 31 | 70 |
| 20 | 48.2 | 127.5 |




(그림 10) Teamwork의 향상율 그래프 (삭제)
(Fig. 10) Improvement rate graph of Teamwork (Deletion)

삭제를 통한 도메인 분석정보의 재사용에서는 그룹 I의 도식적 방식이 그룹 II의 도메인 분석정보 처리기의 처리 시간보다 평균 54.8 % 빨랐으며 복잡도(C)에 비례적으로 증가하였다.

#### 4.2.3 검 토

위의 적용의 결과 분석된 정보는 도메인 분석명세 언어로 표현할 수 있었으며 도메인, 파트, 객체 단위의 명세에 추가, 삭제의 재사용 연산을 통해 새로운 도메인 분석명세 인스탄스의 생성이 가능하였다. 분석정보 재사용 처리의 비교에서 도식적 방식인 Teamwork/SA에의한 추가 연산은 객체의 선택, 위치 선정, 크기 조절과 같은 절차가 필요하므로 시간이 더 필요했으나 삭제의 경우 도식적 방식은 삭제 항목의 그룹화 정도와 위치 그리고 실험자의 경험에 따라 결과에 많은 차이를 가져왔다. 이에 비해 분석명세 언어에 의한 방식은 도식적 방식보다 추가와 삭제에 있어 위치 및 경험 유무의 영향이 적었다. 한편 전체를 이해하는데는 도식적 방식이 유리하였으며 불일치성, 삭제의 오류 검사 기능은 형 선언을 먼저하고 다시 몸체 부분에서 명령을 표현하는 분석명세 언어의 방법이 유리하였다.

## 5. 결 론

소프트웨어의 재사용성 향상을 위해 유사 시스템간의 공통점을 추출하여 비슷한 분야의 응용에 손쉽게 재사용하고자 하는 도메인 분석의 중요성

은 커지고 있는데 현재의 도메인 분석은 분석의 절차에 치중되어 통일된 방식이 없이 도식적으로 표현하고 수작업으로 재사용함으로서 체계적인 도메인 분석정보의 재사용이 용이하지 않았다.

본 연구에서는 잘 정의된 구문과 연산을 갖는 분석명세 언어로 도메인 분석 결과를 표현하여 이들 명세의 오류를 검사하고 재사용 처리하여 새로운 명세의 인스탄스를 생성할 수 있는 도메인 분석정보 처리기를 개발 하였다.

이를 위하여 본 논문에서는 어휘 분석, 구문 분석, 의미 분석의 단계를 갖는 도메인 분석정보 처리기를 설계, 구현하였다. 명세 분석기의 실험을 통해서 분석명세 언어로 표현된 명세의 구문과 의미를 검사할 수 있었고 재사용 가능한 명세 자산을 구분할 수 있었다. 분석명세 재사용기 실험을 통해서는 도메인, 파트, 객체의 단위로 표현된 도메인 분석명세에 추가, 삭제, 계승 및 실체화의 재사용 연산으로 새로운 분석명세 인스탄스의 생성이 가능하였다. 그리고 도메인 분석정보 처리기에 의한 재사용 방식과 데이타 흐름도 방식의 Teamwork/SA에 의한 재사용 처리 시간을 비교하였는데 추가 연산을 통한 재사용에는 도메인 분석정보 처리기의 방식이 평균 32% 빨랐으며, 삭제 연산을 통한 재사용은 Teamwork/SA의 도식적 방식이 평균 54.8% 빨랐다.

이를 통하여 대여 도메인에서 도메인 분석정보 재사용기를 통해 도메인 분석정보의 재사용이 가능하였으며 서로의 단점을 보완하여 보다 높은 도메인 분석정보의 재사용성을 위해서는 도식적인 방식과 분석명세에 의한 방식 두 가지를 함께 지원하는 복합적 방식의 필요성을 발견할 수 있었다. 아울러 도식적 표현을 도메인 분석명세 언어로 변환하고 도메인 분석명세 언어를 도식적으로 표현해 주는 소프트웨어 도구의 필요성이 요구되었다.

앞으로의 연구 과제는 도메인 분석명세 언어를 통한 재사용의 응용에 관한 연구와 CASE의 각 개발 단계에서 연관성있게 도메인 분석명세를 손쉽게 재사용할 수 있도록 사용자 인터페이스와 변환기를 수반한 재사용 도구에 관한 연구가 필요하다.

## 참 고 문 헌


[1] Arango, G., "Domain Analysis: From Art Form to Engineering Discipline", Proc. 5th. Int. Workshop on Software Specification and Design, pp.152, 1989.

[2] Arango, G., Domain Engineering for Software Reuse, Ph.D Dissertation, UCI, 1988.

[3] Frakes, W. at al, "Success Factors of Systematic Reuse", IEEE Software, pp. 17, September 1994.

[4] Frakes, W., "Systematic Software Reuse; a Paradigm shift", Third International Conference on Software Reuse, IEEE, pp.2, 1994,.

[5] Freeman, P., "A Conceptual Analysis of the Draco Approach to construction Software Systems", Tutorial: Software Reusability, IEEE Computer Society, pp. 192, 1987.

[6] Kang, K. at al, FODA Feasibility Study, Technical Report CMU/SEI-90-TR-21, SEI, 1990.

[7] Neighbors, J.,Software Construction Using .Components, Ph.D Dissertation, UCI,pp.1,20,1980.

[8] Neighbors, J., The Darco Approach to constructing Software from Reusable Components, Tutorial: Software Reusability, IEEE Computer Society, pp.181, 1987.

[9] Neighbors, J.,Draco: "A method for engineering reusable software system", pp. 303, Software Reusability, Vol. 1, ACM, 1989.

[10] Prieto-Diaz, R., "Domain Analysis for Reusability", Proc. COMPSAC 1987, pp. 23, 1987.

[11] Prieto-Diaz, R., "Domain Analysis: An Introduction", ACM SIGSOFT SEN Vol. 15 no 2, pp.52, April 1990.

[12] Prieto-Diaz, "Status Report: Software Reusability", IEEE Software, pp.61, May 1993.

[13] Tracz, W., "Reuse State of the Art and State of the Practice Report Card", Third International Conference of Software Reuse, IEEE Computer Society, pp. 194, 1994.

[14] 송영재, 김지홍, 도메인 분석정보의 재사용을 위한 언어의 설계에 관한 연구, 한국정보학회논문지, 투고 중, 1995.

**김 지 홍**

1974년 경희대학교 전자공학과(학사)
1982년 미국 California State University 전자계산학과(석사)
1994년 경희대학교 전자계산공학과 박사 과정 수료
1982년~89년 미국 Interpac Software 연구개발엔지니어
1989년~현재 경원대학교 전자계산학과 조교수
관심분야 : 소프트웨어 공학(소프트웨어 재사용, CASE)

**송 영 재**

1969년 인하대학교 전기공학과 졸업 (공학사)
1976년 일본 Keio University 전산학과(공학석사)
1979년 명지대학교 대학원졸 (공학박사)
1971년~73년 일본 Toyo Seiko 연구원
1980년~82년 공업진흥청 공업표준 심의위원
1982년~83년 미국 Univ. of Maryland 전산학과 연구교수
1986년~88년 대한전자공학회 전자계산연구회 전문위원장
1985년~89년 IEEE Computer Society한국지회 부회장
1984년~89년 전국대학전자계산소장협의회 총무이사, 부회장
1984년~89년 경희대학교 전자계산소장
1990년~91년 일본 게이오대학 전산학과 객원교수
1976년~현재 경희대학교 공과대학 전자계산공학과 교수
1993년~현재 경희대학교 교무처장
관심분야 : 소프트웨어 엔지니어링, Object-oriented Programming & System, CASE도구, S/W개발도구론, S/W재사용