개선된 SARM을 이용한 객체지향 부품 재사용 시스템

한 정 수† · 송 영 재††

요 약

본 연구는 소프트웨어 부품 재사용을 위해 구문분석을 통하여 재사용 가능한 부품으로 만들고, 부품의 이해를 위한 뷰어(Viewer)를 설계하였으며, 검색시간을 단축시킨 개선된 SARM(E-SARM)을 이용하여 유사한 부품들까지 검색할 수 있도록 하였다. 또한 검색을 위해 부품과 질의어 관계를 정의할 수 있는 인터페이스를 구성하였으며, 검색된 부품들을 객체 다이어그램으로 표현하여 부품의 이해성을 높이도록 하였고, 다이어그램 상에서 직접 부품의 재사용이 가능하도록 하였다. 그리고 다이어그램 상에서의 재사용은 코드의 수정이 필요하기 때문에 부품 수정을 위한 편집기 기능을 제공하여 그 효율성을 높였다. 따라서 본 연구에서는 구문분석, 뷰어(Viewer), 질의어와 부품관계 인터페이스, 부품 검색(E-SARM), 다이어그램 표현, 부품의 재사용, 편집기로 구성된 재사용 시스템을 구축하였다.

Object-Oriented Components Reuse System using Enhanced SARM

Jung-Soo Han† · Young-Jae Song††

ABSTRACT

In this paper, we made software components reusable through syntax-analysis method, designed a Viewer for understanding component information, and retrieved similar components by using Enhanced SARM. Because SARM requires a lot of computation time, it was enhanced by reducing unnecessary activation value. Also GUI was designed for component-query relationship and Viewer represents hierarchy diagram of a retrieved component. This system supports facilities which can insert and delete components on diagram. For a component modification, this system supports a editor to rebuild class inheritance relationship. In this paper, SCRS (software components reuse system) is consisted of syntax-analysis method, component-query relationship interface, retrieval (Enhanced SARM), diagram viewer, reuse on diagram, and a editor.

1. 서 론

소프트웨어 재사용(Software Reuse)은 많은 종류의 시스템 개발시 공통적으로 발견되는 소프트웨어 라이브러리를 새로운 소프트웨어 개발시 재사용함으로써 개발시간과 비용을 감소시키고, 품질과 생산성을 향상시키는 기술이다. 또한 시스템에 대한 정보와 개발방법을 공유함으로써 개발자에게 개발노력의 감소와 소프트웨어의 확장성, 유지보수에 관심을 집중시켜 소프트웨어의 신뢰성, 사용성, 효율성을 높일 수 있도록 해 준다[1]. 재사용 기술은 객체지향 프로그래밍(OOP ; Object-Oriented Programming), OMT(Object Modeling Technique), OOA/OOD등의 객체지향 방법론으로 발전함에 따라 객체의 효율적 관리와 재사용 방법에 집중되고 있다[2]. 그러나 재사용을 위해서는 객체들을 효율적으로 저장하고 관리, 검색하는 방법 등의 기술이 요구되고 있다. 소프트웨어 부품(Software Components)은 기존의 소프트웨어

† 정 회 원 : 경희대학교 대학원 전자계산공학과
†† 종신회원 : 경희대학교 전자계산공학과 교수
논문접수 2000년 1월 21일, 심사완료 2000년 4월 8일

를 분석하여 재사용 가능한 부품단위와 정보 즉, 코드, 클래스, 상속관계, 다이이그램 정보 등을 추출, 분류하여 정보저장소(Repository)에 저장한 후 다양한 검색방법에 의하여 재사용된다[3, 4].

기존의 재사용 시스템은 객체의 재사용을 위한 부품검색 부분과 문서화 정보에 의존하고 실제 재사용에 대한 제한적 경험을 바탕으로 이루어져 왔다. 재사용을 위한 검색방법인 RSL[5]은 자연어 형식의 질의어를 이용하지만 검색할 때 사용되는 키워드의 의미와 질의어 재구성, 브라우징 기능이 없다 또한 Diaz 분류 메커니즘[6]은 모듈을 패싯 분류하고 영역 모델을 설정하여 질의어로 검색한다. 그러나 한번 분류가 설계되면 고정적인 제한성을 가진다. MT-View[7]는 질의+인덱스+브라우징 검색방법을 사용하지만 부품의 재사용과정이 요구된다. 본 연구에서는 객체지향 재사용을 위한 정보저장소의 구성을 소프트웨어 부품에 대한 정보 즉, 클래스이름(classname), 멤버함수(method), 속성(attribute), 상속관계(inheritance) 정보로 구성하였다. 검색 방법으로는 Spreading Activation[8] 알고리즘을 사용하였는데, 이 알고리즘은 부품요소간의 연관성에 대한 연결강도(connection relaxation)를 기초로 한 방법이다. 이 기술은 검색어에 연관이 있지만 직접 연결되지 않은 유사한 부품도 검색하도록 해준다. 그러나 검색할 때 활성값을 이용하여 유사도를 측정하기 때문에 시간이 많이 걸리는 단점이 있다 따라서 본 연구는 획장 검색된 결과를 그대로 유지하면서 선별계산 방법을 적용하고 활성값의 재산 회수를 줄여 검색 시간을 단축시켰다. 검색 과정에서는 검색어와 소프트웨어 부품 사이의 관계를 지정할 수 있도록 인터페이스를 설계하여 부품과 검색어의 생성, 삭제, 재구성이 가능하도록 하였다 검색아를 입력하면 개선된 SARM(Enhanced Spreading Activation Retrieval Method) 알고리즘에 의한 부품과 그 유사 부품들은 활성값 순으로 검색되고, 검색된 부품은 뷰어(Viewer) 기능을 통하여 클래스 계층도로 나타낼 수 있도록 하였다. 그리고 계층도 상에서 직접 클래스를 선택하여 다른 계층도로 삽입이 가능하도록 구현하였다 부품의 이동은 계층도의 재구성이 가능해야 하기 때문에 상속에 대한 클래스명은 자동으로 변환되고, 부품에서의 코드의 수정이나 새로운 기능의 추가는 편집기를 이용하여 수정할 수 있도록 하였다.

본 연구는 질의어를 통한 부품뿐만 아니라 유사한 부품들을 E-SARM 알고리즘으로 검색할 수 있도록 하였다. 또한 검색된 부품을 객체지향 다이어그램으로 표현하여 계층도 상에서 직접 부품 재사용이 가능하도록 구현하였다. 추출 뷰어(Viewer)와 검색 인터페이스, E-SARM 알고리즘, 계층도 표현과 재사용 기술, 편집기 기능은 Visual C++ 6.0 MFC Library를 이용하여 구현하였다

본 논문의 구성은 2장에서 기존의 관련 연구들을 고찰하였으며, 3장에서 부품 검색 시스템의 구조와 개선된 SARM 알고리즘을 기술하였으며, 4장에서는 구현한 부품 재사용 도구를 이용하여 검색된 부품의 재사용 방법을 기술하였고, 5장에서는 개선된 SARM의 성능 평가와 다른 시스템과 비교하였으며, 끝으로 6장에서는 결론 및 향후 연구 과제에 대하여 서술하였다

2. 관련 연구

본 절에서는 기존에 연구된 부품 검색 시스템의 장단점과 부품 재사용 방법 그리고 SARM 알고리즘에 대하여 살펴본다.

2.1 RSL 시스템

RSL[5]은 재사용 부품 속성으로 재사용 주석을 정의하고 속성이름과 기능설명을 가진 레이블을 할당하여 PDL과 소스코드로부터 재사용에 필요한 정보를 추출한다 검색 메커니즘은 입력된 질의어의 키워드 집합과 저장되어 있는 키워드를 비교하여 일치되는 부품들을 검색한다 검색된 부품들은 품질 평가항목의 중요도에 따라 재평가된 후 가장 적합한 부품을 제공한다 그러나 이 시스템은 부품 저장과 검색에 사용되는 키워드에 따라 시스템의 효율이 결정된다는 특징이 있다. 즉, 검색할 때 사용되는 키워드의 의미를 파악하기 어렵기 때문에 질의어 재구성이나 브라우징을 통한 부품관리가 어려운 단점을 가지고 있다. 또한 순수한 정보 검색 방법으로 질의에 사용된 특별한 키워드의 의미를 추론하기가 어렵다.

2.2 Diaz 분류 메커니즘

Diaz 분류 메커니즘[6]은 모듈들이 갖는 열거 속성들을 각각 패싯 분류하여 패싯에 해당하는 원소들을 조합시켜서 원하는 기능을 정의할 수 있고, 분류 대상에 따라 적절히 적응하기 쉬울 뿐 아니라 지속적으로 모듈의 집합이 확장되는 경우에도 계층적 방법보다 쉽

게 대처할 수 있다. 검색과 탐색을 지원하는 방법으로 개발된 프로토타입 시스템이 질의어 구성, 검색, 순서를 결정하는 시스템으로 구성되어 있고, 원하는 부품에 대한 질의는 각 패싯에서 항을 선택하여 부품 기술서(descriptor)를 형성한다. 질의는 수정도 가능하며 검색에 실패할 경우 기능이 비슷한 부품의 추출을 위해 동의어 관리 방법을 이용하여 새롭게 질의가 작성된다. Diaz에 의해 제안된 분류와 검색 방법은 영역 모델이 설정되어 질의의 구성과 재구성에 사용된다. 그러나 한번 분류가 설계되면 고정적이 된다는 제한성을 갖는다.

2.3 CATALOG 시스템

CATALOG[9]는 비구조화된 자료를 처리할 수 있으며 구축된 데이터베이스 내의 각 모듈이 헤더 기술서로부터 인덱싱에 필요한 정보를 얻어낸다. 검색은 메뉴 인터페이스와 전문가를 위한 명령어 중심의 탐색 인터페이스를 제공하고, 부울리언 질의와 부분적인 스트링 매칭기법을 허용한다. 또한 의미 없는 단어들이 정지리스트(stoplist)에 의해 처리되지만 라이브러리의 확장성, 부품의 이해, 부품의 선택과 수정을 통한 통합 등의 문제를 해결하지 못하였다.

2.4 MT-View 시스템

MT-View[7] 시스템은 소프트웨어 재사용의 실체화를 목적으로 부품 검색과 수정을 통하여 조립할 수 있는 다중템플리트(Multiple-Template Views) 시스템이다. 이는 혼합형(Hybrid) 검색 방법에 따른 유사성 평가를 통하여 검색된 부품의 이해를 위한 시스템이다. 검색방법은 템플리트 유도나 제시된 패싯 항목을 선택한다. 또한 검색한 부품의 수정을 통하여 새로운 컴포넌트로 생성할 수 있는 편집기를 제공한다. 그러나 MT-View 시스템에서는 유사부품 검색을 위한 보다 효율적인 검색 알고리즘과 라이브러리에 대한 지식 기반정보를 활용하는 재사용 프레임워크를 필요로 한다.

2.5 SARM

검색방법 중 하나인 키워드 접근법은 각 키워드와 직접 연결된 부품들을 검색하기 때문에 질의어와 유사한 부품 검색에는 어려움이 있다. 그러나 SARM (Spreading Activation Retrieval Method)[8]은 부품과 질의어 사이에 질의어 기능을 포함하는 유사한 부품들을 검색하여 보다 더 정확하고 넓은 범위의 부품들을 찾을 수 있는 방법이다. 재사용을 위한 부품들은 각 질의어와 부품들 사이에 연결된 링크에 따라 활성값을 계산하여 검색된다. 검색과정은 질의어에 기본 활성값 1.0을 지정하고 계산과정에서 양수의 활성값은 연결 강도에 누적되며, 음수나 0의 활성값은 전달되지 않는다. 예를 들어, 두 노드가 연결강도 0.5인 연결에 의해 전달되면 활성값을 받는 노드는 1/2인 0.25를 받게된다. 각 노드는 자신에게 입력되는 활성값을 누적시킨다. 입력된 활성값은 입력 데이터(fan-in), 단계별 감소비율(decay rate) 그리고 파라미터에 의해 조절된다. 즉, 시작 질의어에 기본 활성값을 부여하고 질의어에 연결된 각 부품들에 활성값을 전달한다. 계산과정이 순환되면서 활성값이 거의 변화가 없는 안정(stable)된 값을 가지며 직접 연결된 부품과 비슷한 값을 갖는 부품들이 검색된다[10].

SARM 검색 시스템은 활성값이 안정되거나, 사용자가 설정한 최대 사이클 수만큼 실행한다. 그러나 SARM은 활성값을 이용한 반복 계산으로 유사도를 측정하기 때문에 검색시간을 지연시키는 단점이 있다. 이는 부품들이 증가할수록 활성값의 계산회수가 지수적으로 증가하기 때문에 부품이 많을수록 검색에는 어려움이 있다. 이처럼 SARM은 직접 인덱싱되어 있지 않은 부품들까지 검색할 수 있는 효율적인 검색 방법이며 정보저장소에 부품들을 구축할 때 각 항목들을 일일이 인덱싱하지 않아도 되기 때문에 많은 비용이 절감된다. 따라서 SARM 검색 시스템 향상을 위해서는 SARM의 성능은 최대한 유지하면서 검색 속도를 향상시킬 수 있는 방법이 요구된다.

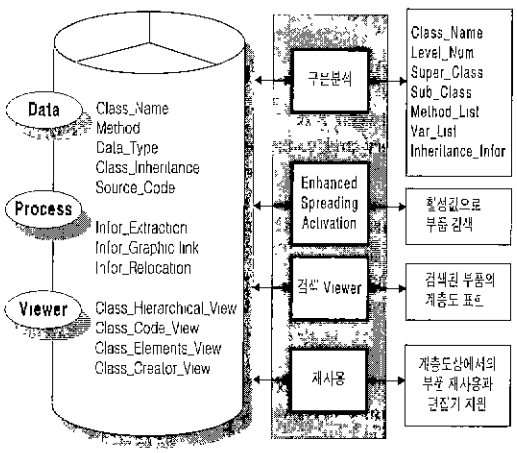
3. 부품 검색 시스템

개선된 SARM(E-SARM)으로 검색된 소프트웨어 부품들은 상속관계를 알 수 있는 계층도로 표현되고, 계층도상에서 직접 부품이동이 가능하며, 부품 수정을 위한 편집기를 지원하여 부품의 재사용이 가능하도록 하였다.

3.1 재사용 시스템 구조

소프트웨어 부품 재사용 시스템은 (그림 1)에서처럼

부품정보를 얻기 위하여 구문분석기를 통하여 정보저장소에 저장하였다. 정보저장소는 데이터(Data), 프로세스(Process), 뷰어(Viewer)로 구성하였고, 검색 뷰어를 통하여 질의어를 입력하면 E-SARM에 의하여 활성값이 계산된 후 활성값 순으로 부품들이 목록으로 나타난다. 이때 부품을 재사용하기 위해서는 계층도상에서 부품을 선택하여 복사, 붙여넣기 기능을 이용하면 부품정보, 원시코드 정보, 그래픽정보가 자동으로 이동된다. 그러나 계층도상에서의 부품이동은 새로운 클래스의 서브클래스가 됨을 의미하기 때문에 이동된 클래스의 원시 코드는 상속관계 정보가 변형되어야한다. 즉, 클래스명에 상속을 의미하는 상위 클래스명이 바뀌어야 한다. 이를 위하여 상속관계에 있는 상위 클래스명을 참조하여 하위 클래스명의 상속정보가 변경된다 부품이 계층도상에서 이동되었어도 재사용 부품은 새로운 소프트웨어 구축시 각 기능에 맞도록 수정이 필요하다. 따라서 본 시스템은 편집기를 지원함으로서 부품의 수정이 가능하도록 하였다 또한 편집기가 부품 수정에 알맞도록 멤버함수, 변수 선언 등의 기능을 지원하도록 하였다. 클래스 계층도상에서 수정할 클래스를 선택하면 편집기가 원시코드와 함께 지원되고 코드를 수정한 후 저장하면 정보저장소에는 코드정보가 갱신되며 부품의 재사용이 이루어진다.



(그림 1) 소프트웨어 부품 재사용 시스템

3.2 부품 분석

재사용을 위한 부품들은 구문분석을 통하여 클래스명, 멤버함수, 속성, 변수, 수퍼클래스와 서브클래스의 상속관계 등의 정보를 추출하여 정보저장소를 구축하였다[11, 12] 또한 각 부품에 대한 계층도 표현을 위하여 클래스 추출정보, 상속정보를 바탕으로 한 링크정보 그리고 각 클래스의 위치정보를 연결시켜 계층도 표현에 이용할 수 있도록 구축하였다. 위치정보는 각 클래스들이 계층도로 나타나기 위해서는 수퍼클래스를 중심으로 심호 균등하게 표현되어야 하기 때문에 계층도의 각 레벨 단위로 정보를 추출하였다[11].

3.3 부품 뷰어

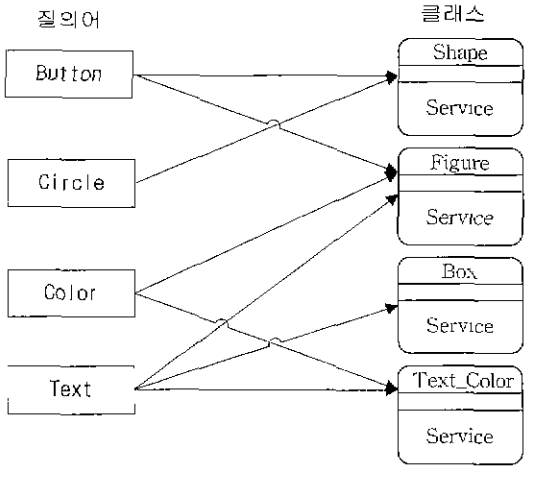
뷰어는 부품들을 분석한 정보를 이용하여 시각화하기 위한 기능으로서 부품을 구성하고 있는 최상위 클래스를 중심으로 각 클래스와 상속관계(inheritance relationship)를 갖는 서브클래스의 수 등을 이용하여 객체지향 다이어그램으로 표현하였다. 스키마 정보는 객체의 깊이와 너비를 계산하고 클래스와 서브클래스 수, 각 레벨에서의 클래스 수를 이용하여 객체(object)와 상속관계를 설정한 후 다이어그램으로 표현된다. 이때 각 클래스의 위치와 함께 클래스명이 표시된다. 또한 다이이그램 상에서 각 클래스에 대한 원시 코드를 볼 수 있도록 코드에 대한 정보가 링크되어 부품의 이해도를 높이도록 하였다. 따라서 뷰어(viewer)는 정보저장소에 저장된 부품 정보를 이용하여 다이어그램, 클래스 상속 관계, 각 클래스에 대한 코드정보 등을 볼 수 있도록 하였다. 뷰어에서 중요한 기능은 부품 전체나 일부분의 클래스들을 삽입, 삭제할 때 각 레벨에서 증가, 감소된 클래스들의 수를 기준으로 각 클래스의 위치를 설정하여 다이어그램으로 재배치해야한다.

3.4 E-SARM

SARM에서 활성값 계산을 위한 순환을 반복하는 것은 정확한 활성값으로 인한 유사 부품을 검색하기 위한 목적이다. 이 방법으로는 각 질의어와 부품들의 활성값이 다른 질의어와 부품들의 활성값 계산에 영향을 주기 때문에 시간이 많이 걸리는 단점이 있었다.본 연구에서는 SARM 단점을 해결하기 위하여 인덱싱이 적은 부품들의 연결정보를 제거함으로써 활성값 계산회수를 줄여 검색시간을 단축시키는 방법으로 개선하였다. 즉, E-SARM (Enhanced Spreading Activation Retrieval Method)은 순환과정이 일정 수준 반복된 후 기준에 미치지 못하는

질의어나 부품들의 연결정보를 제거하여 연산에서 제외시킴으로써 질의어의 확장범위를 줄여 보다 관계가 깊은 부품들만을 검색하도록 하였다.

(그림 2)는 질의어와 클래스의 연결관계를 보여주고 있다. 클래스는 소프트웨어 부품의 근노드(root node)에 해당되는 최상위 클래스로 정의하였다. 질의어 *Button*을 입력하면 3개의 클래스가 검색된다 여기서 질의어 *Button*은 클래스 *Shape*와 *Figure*에 직접 연결되어 있지만 클래스 *Text_Color*와 *Box*에는 연결되어 있지않다 그러나 *Button*(질의어)→*Figure*(클래스)→*Text*(질의어)→*Text_Color*(클래스)를 통하여 연결되고, *Button*(질의어)→*Figure*(클래스)→*Text*(질의어)→*Box*(클래스)를 통하여 2개의 클래스(*Box, Text_Color*)가 연결됨을 알 수 있다 하지만 *Box*는 검색과정에서 적게 참조되므로 연결이 제거된다. 이처럼 각 질의어와 클래스 부품은 서로 연결되어 있는 노드를 참조해 가면서 활성값을 계산하게 된다. 또한 계산결과 *Shape*가 *Figure*클래스 보다 높게 나타나는데 그 이유는 *Shape*가 *Figure*보다 참조 회수가 더 많기 때문이다. 기존의 SARM과 E-SARM의 평균 부품참조 회수 시뮬레이션 결과 활성값 계산회수가 37.8% 감소됨을 알 수 있었다. <표 1>은 E-SARM에서 사용하는 활성값 계산 변수들에 대한 정의를 보여주고 있고, 알고리즘에서 질의어와 부품들은 메트릭(metrics)을 이용하여 활성값을 계산하였다. 순환이 반복될수록 활성값은 안정되며 참조회수가 기준에 미달되는 부분은 자동으로 제거되어 계산과정이 종료된다.



(그림 2) 클래스와 질의어 관계

Enhanced Spreading Activation 알고리즘

```
while
 Get_Level = Get_Pos[0] / End1;
            /* position of row matrix */
 Get_Col = Get_Pos[0] % End1;
            /* position of column matrix */
 if(exist component)
   for(all component number)
       Array[] = each component value(0 or 1)
 else
   for(all query number)
       Array[]= each query value(0 or 1)
 for(query or component number)
   if(exist relationship and index is Not last_index)
       push Current_index in Stack
 query_visit_count ++
 component_visit_count ++
 if(first query)
   initialize query_act_value=1.0
 else if(a query)
       D_i(t+1) =
       δ_Di(t) + φ_Di(t)(M − D_i(t))  if  φ_Di(t) > 0
       δ_Di(t) + φ_Di(t)(D_i(t) − m)  if  φ_Di(t) ≤ 0
       δ_Di = (1 − θ_Di) D_i(t)
     else if(a component)
           T_i (t-1) =
           δ_Ti(t) + φ_Ti(t)(M − T_i(t))   if  φ_Ti(t) > 0
           δ_Ti(t) + φ_Ti(t)(T_i(t) − m)   if  φ_Ti(t) ≤ 0
           δ_Ti = (1 − θ_Ti) D_i(t)
 else Err("Not Calculate Activation Value")
 pop(Get_Pos);
 if( Not MAX_CYCLE )
 { cycle++;
  if(cycle > MAX_CYCLE) /*MAX_CYCLE=3~5*/
          break;
  if(cycle is between 2 & 3)
    for( all query and components)
      if(query_visit_count or component_visit_count
        = current_cycle_count)
      {  AvgVisit += VisitNum[];
         cnt++; }
    if(cnt==0) return 0;
    else AvgVisit /= cnt;
  }
  if(visit_count exist)
   for( all query and component)
     if(VisitNum <= AvgVisit)
     {
       for(j =0 ; j < End1; j++)
          level0_1[j][i%End1]=0;
       Cut_component_value = -9990;
       // 제거된 노드의 초기화
     }
endwhile
```

〈표 1〉 E-SARM 파라미터

기 호	성 의
$\delta_{D_i}(t)$	질의어 감소비율로 감소된 이전의 활성값
$\phi_D(t)$	현재(t) 클래스 i에게 들어오는 입력값
M	최대 활성값 = 1.0
$D_i(t)$	이전 단계의 누적 클래스 활성값
θ_D	클래스 감소 비율 = 0.1
$\delta_{T_i}(t)$	클래스 감소비율로 감소된 이전의 활성값
$\phi_{T_i}(t)$	현재(t) 질의어 i에게 들어오는 입력값
m	최소 활성값 = -0.2
$T_i(t)$	이전 단계의 누적 질의어 활성값
θ_T	질의어 감소 비율 = -0.2

3.5 소프트웨어 부품 정보와 질의어

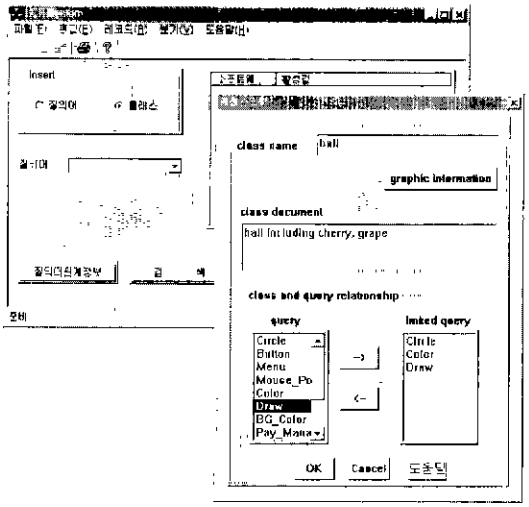
소프트웨어 부품 정보는 클래스명, 멤버함수, 속성, 슈퍼클래스와 서브클래스의 관계, 소스코드, 부품과 질의어 관계로 구성된다[13] 각 부품들은 클래스들로 구성되어 있기 때문에 부품명을 최상위 클래스명으로 정의하였다 <표 2>는 E-SARM 검색시스템에서 사용하는 데이터들을 정의한 것이다. 데이터는 부품명, 상위 클래스, 서브클래스, 관계(relationship), 질의어 등으로 구성하였다. 또한 질의어는 각 클래스의 기능을 포함할 수 있도록 정의하여 각 클래스와의 관계를 지정해 주었으며 기본 클래스는 질의어와 연결되는 부품의 최상위 클래스이다. 또한 subclass1, subclass2,... 등은 다이어그램 표현을 위하여 정의하였고, 질의어 관계는 질의어와 기본 클래스 즉, 부품과의 연결관계를 나타낸다. 이와 같이 재사용 가능한 부품들을 질의어와 연결하여 부품화시키면 질의어에 의해 활성값이 계산된다. 질의어는 인터페이스 상에서 리스트 형식으로 보여줌으로서 질의어 사용에 편리함을 제공하였고 질의어를 통한 검색은 질의어와 직접 연결된 부품과 간접 연결된 부품들이 활성값 순으로 검색된다.

(그림 3)은 클래스와 질의어를 데이터베이스에 추가, 삭제할 때 관계정보를 입력하는 인터페이스이다. 삽입(insert) 부분의 클래스 입력부분을 선택하면 부품 추가 대화상자가 나타난다. 클래스명(*ball*)과 클래스 도큐먼트를 입력하고 클래스와 질의어 사이에 연결 정보를 선택한다. 이때 연결설정에 주어지는 리스트들은 데이터베이스 내에 있는 질의어의 집합이고 부품에 관련된 질의어를 선택함으로서 연결정보가 구성된다. 그리고 질의어를 리스트화하여 잘못된 연결을 방지하고 질의어에 대한 정보와 이해를 높일 수 있도록 하였다. 또한 질의어 삽입 역시 질의어 생성 인터페이스가 제공되어 새로운 질의어와 부품 사이의 연결 정보를 설정할 수 있도록 하였다

〈표 2〉 데이터 정의 테이블

my_file : 테이블

필드 이름	데이터 형식
Index	숫자
활수기능(질의어)	문자열
기본 클래스	문자열
구성 클래스의 갯수	숫자
ClassName	문자열
ClassID	숫자
Class_ParentCount	숫자
Class_Level	숫자
Class_Postion	숫자
▶ Class_left	숫자
Class_Top	숫자
Class_Right	숫자
Class_Bottom	숫자
SubClass1	문자열
SubClass2	문자열
SubClass3	문자열
SubClass4	문자열
SubClass5	문자열
FunctionList	메모
VarList	메모
클래스DOC	메모
질의어 관계1	숫자
질의어 관계2	숫자
질의어 관계3	숫자



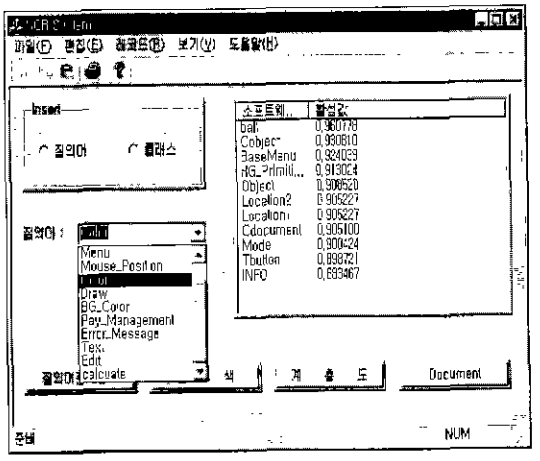
(그림 3) 클래스와 질의어 생성

4. 소프트웨어 부품 재사용

E-SARM 방식으로 검색된 부품은 활성값 계산 과정에서 인덱싱이 적은 것을 제거하여 질의어에 직접 연결된 부품과 유사한 부품들을 활성값 순으로 검색할 수 있도록 하였고, 계층도상에서 직접 부품 재사용이 가능하도록 하였다.

4.1 부품 검색

정보저장소에 저장되어 있는 부품 정보와 연결정보를 이용하여 질의어와 직접 연결된 부품과 간접 연결된 유사 부품들이 Enhanced Spreading Activation 알고리즘을 통하여 검색될 수 있도록 하였다. 모든 데이터는 정보저장소에 저장되고, 질의어에 연결된 활성값이 계산되어 안정적이 될 때까지 반복된다. 검색 결과는 기준이 되는 활성값 범위에 해당하는 모든 클래스 부품의 최상위 클래스인 기반 클래스(Basc Class)가 검색된다. 활성값은 참조가 많을수록 값이 커지며 유사한 클래스 부품 역시 참조 기능으로써 검색된다. 이와 같이 검색된 부품들은 여러 개의 부품을 제공하기 때문에 재사용을 위한 부품을 선택할 수 있는 폭이 넓어지며, 질의어에 대한 부품의 이해도를 높일 수 있었다. 본 시스템은 부품 또는 질의어 입력 인터페이스, 질의어 관계정보, 검색, 도큐먼트, 부품 다이어그램 표현, 부품 재사용 기능으로 설계하였다.



(그림 4) 부품 검색 결과

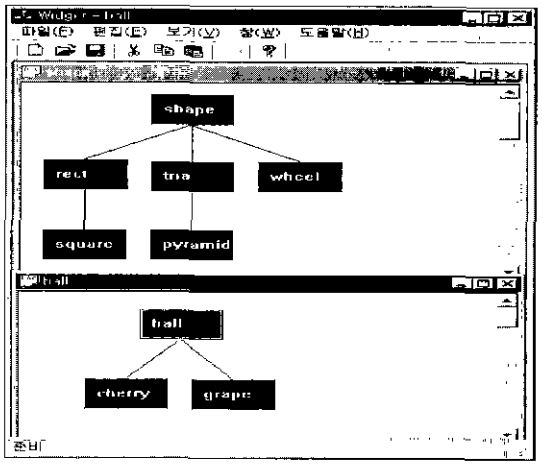
부품 입력은 검색할 부품의 확장성을 고려하여 현재 데이터베이스에 있는 부품들에 추가로 삽입하여 검색이 가능하도록 하였다. 입력 인터페이스에서는 새로운 부품과 질의어와의 관계를 규정하였고, 질의와 부품과의 관계를 나타낼 때는 리스트에 허용 가능한 질의어를 보여줌으로써 오류를 방지할 수 있도록 하였다. (그림 4)는 질의어 *Color*를 입력하여 이와 연결된 부품들 중 활성값이 큰 순서로 부품들이 검색된 결과를 보여준다. 질의어 *Color*입력에 의한 부품 검색결과는 참조 회수가 가장 많은 부품 *ball*을 중심으로 11개의 부품이 검색되었다. *Color*와 직접 연결된 부품들은 상위 5개인 부품 *Object*까지 직접 연결된 부품이고 나머지는 간접 연결된 유사한 부품들이다. 검색된 부품들은 질의어 *Color*의 기능을 갖는 클래스들을 포함하고 있다. 또한 각 부품에 대한 *Document* 기능을 이용하여 부품에 대한 상세 기능 정보를 얻을 수 있도록 하였다. 그리고 각 부품의 클래스 구성 정보를 알기 위해서는 부품을 선택한 후 계층도 기능을 이용하면 부품에 대한 다이어그램이 표현된다. 따라서 검색된 부품의 이해를 위한 정보를 얻을 수 있었고, 재사용을 위한 부품이 선정되면 다이어그램 상에서 직접 재사용이 가능하도록 하였다.

4.2 부품 이동

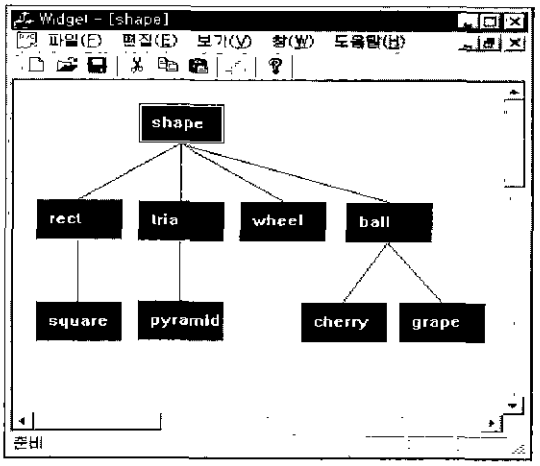
E-SARM 알고리즘으로 부품이 검색되었을 때 부품을 객체 다이어그램으로 표현해 준다. 이때 선택한 부품을 재사용하기 위해서는 다이어그램 상에서 직접 부품의 이동이 가능하도록 하였다 (그림 5)는 *shape*라는 부품에 (그림 3)에서처럼 질의어 *color*를 이용하여 검색된 *ball*이라는 부품을 찾아 재사용하기 위한 다이어그램을 보여주고 있다. *ball*의 부품을 최상위 클래스인 *shape*에 삽입하기 위하여 마우스 오른쪽 버튼의 복사와 붙여넣기 기능을 이용하면 된다. 이때 부품 전체를 사용할 때는 최상위 클래스만 클릭하면되고, 일부분만을 사용하기 위해서는 shift 키를 이용하여 선택한 클래스들을 이동시킬 수 있다.

(그림 6)은 부품 *ball*이 *shape*에 삽입된 결과를 보여준다. 삽입 알고리즘은 소프트웨어 부품을 윈도우상에 다이어그램으로 나타냄과 동시에 각 클래스의 현재 위치정보, 클래스정보, 관계정보를 인식한다. 마우스로 클릭했을 때 선택된 포인터가 클래스 내부에 있는지를 검사하고 선택표시를 해준다. 선택된 클래스를 이동하기 위해서는 클래스의 *id, color, size*등을 저장하고 또 다른 MDI 윈도우로 클릭을 통하여 이동시킨다. 이때의 클래스 이동은 클립보드(clipboard)를 이용하였다. 클립보드에 있는 부품 정보를 삽입하기 위해서는 부품의 상위클래스(*shape*)를 클릭하고 붙여넣기 기능을 이용하면 된다. 이때 객체의 표현은 상위클래스(*shape*)가

서브클래스(*rect, tria, wheel*)를 갖고 있을 때 하위클래스를 검사하면서 클래스 *ball*의 위치는 *wheel* 다음으로 정해진다. 또한 클래스 *ball*의 서브클래스 역시 각 레벨의 클래스 개수와 통합되어 그 위치가 결정된다.



(그림 5) 소프트웨어 부품 다이어그램



(그림 6) 부품의 삽입

부품 이동 알고리즘은 다이어그램 상에서 직접 부품 전체 또는 일부분을 이동시킬 수 있으며, 동시에 각 클래스의 정보와 연결되어 있는 원시코드 정보도 자동으로 이동된다. 이때 상속되는 클래스명은 자동으로 수정된다. 이 알고리즘은 재사용을 위하여 검색된 부품의 이해를 위한 시각적 표현과 직접 부품 이동을 통한 코드의 이동이 이루어지는 효율성을 갖고 있다.

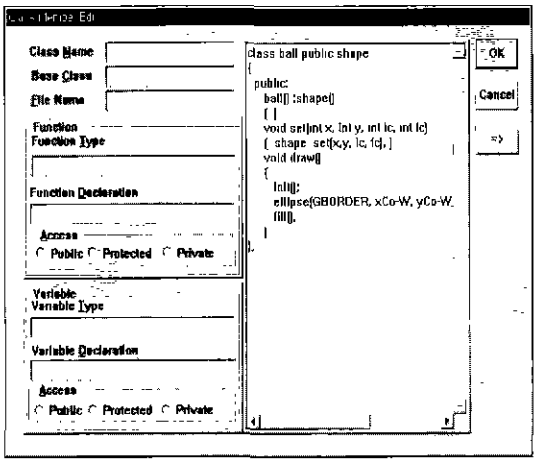
다이어그램 상에서의 부품 이동 알고리즘

```
for(all class_count) /* open a diagram */
  get class_pointer and rectangle coordinate
  draw rectangle and write class name
endfor
if(exist parent_class and
  subclass_name equal to class_name)
  draw line between two classes
endif
for(all class) /* when mouse click */
  get class_box area inform
  decide that click_pointer is in class_box area
endfor
if(no class was clicked)
  change the selection to NULL and exit
  select classes clicked and refresh a diagram
if(exist selected classes) /* copy class */
  define clipboard and get id, color, size of class
  get the active MDI child window
  get the active view attached to the active
        MDI child window
  place classes on the clipboard
if(multiple classes) /* paste class */
for(Max_subclass_num)
  if(sunclass of parent_class is NULL)
    place root class_name that was copied
    move subclass of copied class to created class
    try{
      get level, column, row, position,
          class_id, class_color
      create classes
      append created object to class_list
    }
    catch(Memory Exception Error)
      error("Out of memory")
  else /* single class */
    if(sunclass of parent_class is NULL)
      place copied class_name and create a class
  endif
endfor
endif
```

4.3 부품 수정

소프트웨어 부품 다이어그램 상에서 부품을 이동시켰을 때 상위 클래스에 대한 하위 클래스의 부품 정보가 저장되지만, 원시코드의 상속관계 또는 새로운 기능의 추가는 편집을 통한 수정이 필요하다. 따라서 본 연구에서는 부품 수정을 위한 편집기능을 지원하였다 (그림 7)은 클래스 *shape*에 *ball*을 삽입하고 상속관계인 *ball*의 원시코드를 수정한 내용을 보여주고 있다. 코드수정은 자동으로 되지 못하기 때문에 편집기를 통한 부분 수정이 필요하다 또한 편집기에서는 단순한 코드의 수정 즉, 상속관계 수정뿐만이 아니라 멤버함

수나 클래스도 추가할 수 있도록 하였다. 클래스를 추가할 때에는 보다 편리하게 작성할 수 있도록 일반 편집기능 이외에 클래스명과 멤버함수, 액세스 모드, 변수 등의 프로토타입을 지원해 준다. 따라서 각 필드에 입력하면 코드와 직접 연결되어 클래스가 구성되고 나머지 코드의 입력은 직접 편집기에서 가능하도록 하였다.



(그림 7) 부품 편집기

5. 성능 평가

5.1. E-SARM 효율

본 연구는 소프트웨어 부품을 클래스정보, 클래스사이의 관계정보, 멤버함수, 다이어그램정보 등으로 분석 저장한 후 E-SARM 알고리즘을 통하여 검색하고, 부품을 다이어그램 상에서 직접 재사용이 가능하도록 하였다. 이를 위하여 부품들을 패싯 분류하였고 질의어와의 관계를 정의하였으며 검색된 부품들을 활성값 순으로 열거하여 보다 정확한 부품 검색과 각 부품 이해를 위한 계층도, 문서화, 코드 편집기, 계층도 표현 기능 등을 지원할 수 있도록 하였다. <표 3>은 부품 개수에 대한 활성값 계산회수의 시뮬레이션 결과이다. 시뮬레이션 환경은 n × n 크기의 배열을 이용하여 행은 질의어, 열은 부품을 나타내고, 각 행, 열의 초기값은 질의어와 부품의 연결 상태를 나타낸다 연결관계는 무작위로 희소행렬이 되도록 설정하였다. 이 결과는 부품의 증가에 따라 검색시간이 많이 소요됨을 보여준다 따라서 이 알고리즘에서 참조회수가 적은 부분을 제거시킴으로서 검색 시간을 단축시킬 수 있었다. <표 4>는 <표 3>과 같은 기준으로 실제 검색되는 결과의 개수를 비교한 것이다. 이 결과는 제거 기준이 커짐에 따라 활성값 계산회수가 적어지고 부품 참조회수 또한 감소됨을 알 수 있다. 기존의 SARM 알고리즘과 비교할 때 E-SARM 알고리즘의 평균 부품참조회수가 시뮬레이션 결과 약 37.8%가 단축되었음을 알 수 있었다

또한 검색 시스템의 재현율과 정확성을 측정하기 위해서 질의어에 대한 유사 부품 검색 결과를 실험하였다 재현율은 질의 부품 중 적절한 부품의 수에 대한 검색된 적절한 부품의 수, 정확도는 검색된 전체 부품 수에 대한 검색된 적절한 부품의 수를 이용하여 측정하였다[7]. 질의어는 임의로 10개를 선정하고 각각의 질의어에 대한 부품 검색 개수의 평균으로 측정하였다. <표 5>는 각 질의어에 대한 검색된 평균 부품 수를 나타낸다. <표 6>은 이 결과에 대한 정확도와 재현율을 측정한 것이다. 이 실험에서 E-SARM은 SARM의 검색 결과를 최대한 유지하며, 검색 속도가 빠르기 때문에 효율성이 높음을 알 수 있었다.

〈표 3〉 부품에 대한 활성값 계산회수

부품 크기	SARM	E-SARM
200	22085.41	1710890
400	48637.57	3396471
600	82268.17	5662729

〈표 4〉 부품에 대한 평균참조

부품 크기	SARM	E-SARM
200	6.3	4.75
400	7.88	5.48
600	11.2833	7.4

〈표 5〉 검색된 부품 수

질의어 (10)	검색된 적절한 부품	검색된 부적절한 부품	검색되지 않은 적절한 부품	검색되지 않은 부적절한 부품
임의선정	11.5	1.4	1.3	24.5

〈표 6〉 검색 효율

검색방법	평균 측정율	
	재현율	정확도
E-SARM	89.8	89.1

5.2 기존 시스템과 비교

<표 7>은 부품 재사용을 위한 부품 검색, 뷰, 재사용 방법, 수정을 위한 편집 기능 등의 특성을 기준으로 기존의 연구와 비교 분석하였다. RSL[5]와 Diaz[6]

〈표 7〉 기존의 시스템과 비교

항목 / 시스템	부품 성격	분류 모델	검색 모델	재사용방법
RSL	함수	패싯	자연어질의	검색
Diaz	함수	패싯	자연어질의	검색
CATALOG	함수	열거	부울리언+스트링매칭	검색
MT-View	클래스	열거+패싯	질의+인덱스+브라우징	검색+편집
CARS[14]	클래스	패싯	질의+브라우징	검색
본 논문의 제시방법	클래스	열거+패싯	D-SARM+질의+인덱스+브라우싱	검색+계층도에서 재사용+편집

은 부품을 함수 단위로 패싯분류하여 자연어 형식의 질의를 이용하지만 질의 개구성과 브라우징 기능이 없어 부품관리와 객체의 이해도가 부족하다. 또한 MT-View [7]는 객체지향 소프트웨어의 재사용을 위한 시스템으로서 패싯분류 방법을 사용하고 수정을 위한 편집기능을 지원해주는 시스템이지만 부품의 이해를 위한 계층화 표현방식이 없고 유사부품 검색을 위한 방법이 부족하다. 따라서 본 연구는 객체지향 소프트웨어 부품들을 패싯분류하고 질의어와의 관계를 정의함으로써 부품과 질의어들을 추가할 수 있도록 하였고, 검색에서는 활성값과 부품 참조회수를 이용하여 유사한 부품들까지 검색할 수 있는 객체지향 부품 재사용 시스템을 구축하였다. 그리고 검색된 유사 부품들을 계층도로 표현하여 이해성을 높였으며, 보다 효율적인 부품의 재사용을 위하여 계층도상에서 직접 부품의 이동이 가능하도록 하였다. 또한 삽입된 부품의 수정과 추가를 위한 편집기를 제공함으로써 재사용의 편리함과 효율성을 높이도록 노력하였다.

6. 결 론

본 연구는 객체지향 부품 검색을 위해 개선된SARM을 이용하여 유사한 부품들을 검색할 수 있도록 하였고, 검색된 부품들을 계층도상에서 직접 부품의 재사용이 가능하도록 객체지향 소프트웨어 부품 재사용 시스템을 구축하였다. 재사용을 위해 부품들을 재사용 가능한 부품들로 구문 분석하였고, 부품들을 패싯 분류하여 질의어와 관계정보를 정의할 수 있는 인터페이스를 설계하였다. 인터페이스는 부품이나 질의어를 삽입, 삭제할 수 있고 각 부품별로 질의어와의 관계정보도 입력할 수 있도록 구성하였다. 또한 검색과정에서 부품의 참조회수가 적은 부분을 제거함으로써 많은 시간이 걸리는 단점을 개선시켜 SARM에 비해 37.8% 단축된 시뮬레이션 결과를 얻었다. 그리고 다이어그램상에서의 부품 삽입은 부품 전체 또는 특정 클래스들을 선택하여 삽입할 수 있도록 구현하였다. 또한 부품 재사용시 반드시 필요한 부품 수정이나 기능 추가를 위하여 편집기를 지원하였다 편집기는 코드설계에 도움이 될 수 있도록 파일명과 클래스명 그리고 멤버함수에 대한 *public, protected, private* 중에 어떤 엑세스 모드에 적용되는지 선택할 수 있고, 변수의 형(Var_Type)과 선언(Var_Declaration)란을 이용하여 보다 효과적으로 코드의 수정이 이루어 질 수 있도록 하였다 따라서 본 연구는 기존의 재사용 시스템과 비교하여 검색방법의 효율성과 객체지향 소프트웨어 부품의 재사용을 위한 효율적인 시스템임을 확인하였다

그러나 본 연구는 부품과 질의어 관계설정에 표준화가 되어있지 않아 임의로 관계를 정의하였기 때문에 다양한 질의어와 그 확장에 어려움이 있었다. 또한 검색결과는 활성값에 의존하였기 때문에 부품들에 대한 정확한 기능 정의에만 한정되어 있어서 더 많은 부품의 이해를 위한 정보가 요구된다. 앞으로의 연구방향은 다양한 부품들에 대한 폭 넓은 질의어와 표준화가 필요하며 CASE의 분석, 설계단계에 연관성이 있도록 확장시키는 방법이 요구된다

참 고 문 헌

[1] Carma McClure, "The Three Rs of Software Automation : Re-Engineering, Repository, Reusability," Prentice Hall, pp.221-230. 1992.

[2] Arango G., "Domain analysis methods. Software Reusability," Ellis Horwood, pp.26-37, 1994.

[3] Rumbaugh, J et al, "Object-Oriented Modeling and Design," Prentice-Hall, 1991.

[4] James Petro and Michael E. Fotta, "Model-Based Reused Repositories-Concepts and Experience," IEEE Computer Society Press-Technical Council on Software Eng., pp.60-69, 1995

[5] B A Burton, R W. Aragon, S A. Bailey, K D

koehler and L.A. Mayers, "The Reusable Software Library," IEEE Software, pp.25-33, July 1987.

[6] R.Prieto-Diaz and P.Freeman, "Classifying Software for Reusability," IEEE Software, Vol.4, No.1, pp.6-16, Jan. 1987.

[7] 김행곤, 차정은,"객체지향 프로토타이핑 지원을 위한 컴포넌트 이해 시스템 개발에 관한 연구", 정보처리논문지, 제4권 제6호, pp.1519-1530, 1997.

[8] Scott Henninger, "Information Access Tools for Software Reuse," System Software, pp.231-247, 1995

[9] W.B.Frakes and B.A. Nejmeh, "An Information System for Software Reuse." *Proceedings of the* Tenth Minnowbrook Workshop on Software Reuse, 1987.

[10] Mozer, M.C., "Inductive Information Retrieval Parallel Distributed Computation," ICS Report 8406, Institute for Cognitive Science, Univ. of California, San Diego, June 1984.

[11] 한정수, 송영재, "클래스 부품의 재사용을 위한 객체의 추출과 이해", 정보처리논문지, 제6권 제4호, pp.941-951, 1999, 4.

[12] 정계동, 권오진, 최영근, "프로그램 이해 지원과 재사용을 위한 객체 지향 클래스 라이브러리 설계 및 구현", 정보처리논문지, 제5권 제6호, pp.1507-1521, 1998.

[13] 김재생, 송영재, "재사용에 기반한 객체들의 정보 분석과 자동화에 관한 연구", 정보처리논문지, 제4권 제2호, pp.384-394, 1997

[14] 최은만, 김진석, "객체지향 재사용과 CASE," KISS, Vol.14, No.10, pp.47-54, 1996

한 정 수

e-mail : jshan@case.kyunghee.ac.kr

1990년 경희대학교 전자계산공학과 (공학사)

1992년 경희대학교 전자계산공학과 (공학석사)

1995년 경희대학교 전자계산공학과 박사과정 수료

1993년~현재 경희대학교 전자계산공학과 박사과정

관심분야 : 소프트웨어공학, S/W 재사용, CASE 도구

송 영 재

e-mail : yjsong@nms.kyunghee.ac.kr

1969년 인하대학교 전기공학과 (공학사)

1976년 일본 Keio University 전산학과(공학석사)

1979년 명지대학교 대학원 졸업 (공학박사)

1971년~1973년 일본 Toyo Seiko 연구원

1982년~1983년 미국 Univ. of Maryland 전산학과 연구교수

1985년~1989년 IEEE Computer Society 한국지회부 회장

1984년~1989년 경희대학교 전자계산소장

1976년~현재 경희대학교 전자계산공학과 교수

1993년~1995년 경희대학교 교무처장

1996년~1998년 경희대학교 공과대학장

1998년~현재 경희대학교 기획조정실장

관심분야 : 소프트웨어공학, OOP/S, CASE 도구, S/W 개발도구론, S/W 재사용