컴퍼넌트 재사용에 의한 소프트웨어 아키텍쳐 생성 프레임워크

황 하 진*, 손 이 경**, 김 행 곤**

* 대구효성가톨릭대학교 경영학과

** 대구효성가톨릭대학교 전자정보공학부

요약

소프트웨어 위기를 극복하기 위한 많은 대안들 중 소프트웨어 재사용만이 생산성과 품질면에서 효과를 얻을 수 있는 실질적인 접근법으로 제시되었으며 많은 가능성을 제공한다. 특히, 컴퍼넌트 수준에서 소프트웨어 생산을 취급하게 하고 소프트웨어 부품에 대한 임의의 추상적 서술을 강조함으로써, 개발자가 최소의 구현 명세를 고려하도록 허용하였다. 그러나 일반적인 소프트웨어 재사용과 개발 지원 방법론, 툴등의 결여는 과학적 또는 공학적 과정에서 볼 때, 소프트웨어의 재사용을 방해하는 요소가 될 수 있다.

따라서, 본 논문에서는 소프트웨어 재사용에 관한 기본 이론들과, 객체지향 개발에 초점을 맞추어 보다 효과적으로 재사용 가능한 소프트웨어 컴퍼넌트의 검색을 가능하게 해주는 소프트웨어 아키텍쳐 생성 프레임워크인 FSA(Framework for Software Architecture)를 제안한다.

1. 서 론

소프트웨어의 생산성은 정보 산업의 핵심기술로써 수십년간의 집중적인 연구 결과 많은 향상을 보여왔다. 그럼에도 불구하고 기반 소프트웨어 기술의 미흡, 전문 인력 부족 및 고비용에 의해 증가하는 수요를 충족시키지 못하고 있다. 이러한 차이를 줄이기 위해 많은 대안들이 제시되어 왔지만 그중 소프트웨어의 재사용만이 생산성과 품질면에서 효과를 얻을 수 있는 실질적인 접근법으로 제시되었으며 많은 가능성을 제공하고 있다. 먼저, 소프트웨어의 개발시 컴퍼넌트 단위의 개발을 지원하고 소프트웨어 컴퍼넌트에 대해 추상적인 서술을 강조함으로써, 개발자가 소프트웨어의 개발시 고려해야하는 개발 범위를 줄여 줄 수 있다. 뿐만 아니라, 구현의 상세함에 대해 일부분만 관여하도록 해준다. 다음으로 소프트웨어의 설계를 단순한 코딩작업이 아닌 아키텍쳐(architecture)수준에서 인급함으로써, 알고리즘 선택이나 제어 구조 그리고 데이터 구조와 같이 프로그램 구축 과정에서 언급되어야 하지만 정규화시키기 힘든 요소들을 자동화할 수 있는 가능성을 제공한다. 그렇지만 일반적으로 소프트웨어 재사용과 개발을 지원하기 위한 방법론과 툴등의 결여, 부적절한 관리구조와 개발 훈련, 그리고 현실에서 예상치 못했던 문제의 발생등은 소프트웨어의 재사용을 방해하는 요소가 될 수 있다.

그러므로, 본 논문에서는 소프트웨어 재사용에 관한 지금까지의 여러 견해들을 살펴보고, 객체 지향 개발에 초점을 맞추어 보다 효과적으로 재사용 가능한 소프트웨어 컴퍼넌트의 구축에 대해 언급한다. 또한 재사용을 위해 저장되어 있는 컴퍼넌트들에 대해 효율적으로 검색하고 새로운 소프트웨어에 적절하게 수정하며 합성이 가능한 통합된 소프트웨어 아키텍쳐 생성 프레임워크인 FSA를 제안한다.

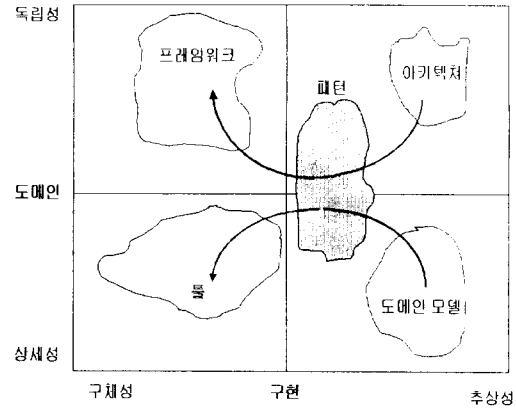
2.4 기술적인 접근방법

소프트웨어 재사용의 기술적 접근방법은 재사용 시스템은 부품이 재사용되는 특성에 따라 부품 조립 기반 시스템과 생성 기반 시스템으로 분류되어 진다. 부품 조립 기반 시스템은 적용 도메인내에서 품질 보증 과정을 거치고 재사용 가능한 컴퍼넌트를 라이브러리에 저장하여 새로운 소프트웨어 개발에 필요한 기능을 제공하는 컴퍼넌트를 다시 사용한다. 즉, 기존의 소프트웨어 개발과정을 최대한 활용하면서 소프트웨어 컴퍼넌트들을 블록으로 만들고 이들 블록들을 서로 조합하여 맞추어 가면서(building block) 하나의 소프트웨어를 완성시키는 방법이다. 이 방법은 bottom-up 형식을 따르며, 성공적인 재사용을 위해서는 컴퍼넌트를 유용하게 쓸 수 있도록 체계적으로 분류해서 라이브러리를 구축해야하며 컴퍼넌트들에 대한 정보의 유용성과 문서화 및 인터페이스의 표준화에 세심한 주의가 필요하다.

생성 기반 시스템은 아키텍쳐를 중심으로 생성하는 방식으로, 관련있는 응용 분야에서 유용한 기능을 종합한 아키텍쳐를 사용하여 각 적용 단계에서 특정한 요소만을 수정함으로써 필요한 컴퍼넌트를 생성해 가는 방법이다. 여기에서 재사용 가능한 아키텍쳐를 프레임워크(framework)라 한다.(그림1)[5,6]

여기서, 아키텍쳐란 설계의 기술 또는 그 과정으로써 같은 유형을 가지는 객체들의 구조를 말한다. 도식적으로 표현되는 소프트웨어 아키텍쳐는 소프트웨어 시스템의 서브시스템이나 컴퍼넌트, 그리고 이들 사이의 관계를 서술한 것으로 이는 소프트웨어 설계 활동의 결과물이라 할 수 있다.

프레임워크란 시스템 또는 서브시스템의 패밀리에 대한 아키텍쳐로 정의할 수 있으며 프레임워크의 생성을 위해 기본적인 빌딩 블록을 제공한다. 특별한 기능을 획득하기 위해 자신의 일부를 다시 정의하기도 한다. 객체지향 프로그래밍에서의 프레임워크는 추상적인 클래스와 구체적인 클래스로 구성된다. 이러한 프레임워크의 instantiation은 기존클래스의 합성과 서브클래싱을 포함한다. 이러한 아키텍쳐의 재사용은 개발하고자 하는 프로그램 아키텍쳐의 중요한 부분에 맞는 프레임워크가 발견될 수 있고, 적절하게 문서화되어 있다면 소프트웨어 개발 시간과 노력을 상당히 절약할 수 있으리라 기대된다.[7,8,9]



(그림1) 프레임워크와 아키텍쳐의 구분

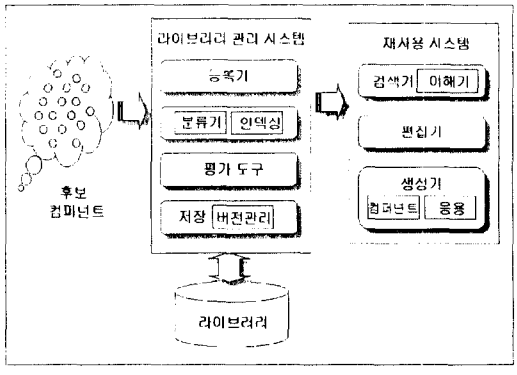
2.5 재사용 시스템의 구축

재사용 시스템을 구축하려고 할 때 가장 주의를 기울여야하는 부분은 컴퍼넌트를 효율적으로 관리하고 저장할 수 있는 라이브러리를 구축하는 것과 그 라이브러리로 부터 개발하고자 하는 응용에 필요한 컴퍼넌트를 찾아 적용시키는 일이다.

먼저 작성하려고 하는 소프트웨어의 요구사항을 분석하여 구조를 설계한 후 적당한 컴퍼넌트가 소프트웨어 라이브러리안에 있는지를 검색한다. 만일 적절한 컴퍼넌트가 발견되지 않을 경우 컴퍼넌트의 분류가 잘 되었는지를 확인하고 새로운 컴퍼넌트를 개발한다. 적당한 컴퍼넌트가 발견되었으면 검색한 컴퍼넌트를 새로운 소프트웨어의 설계 양식에 맞도록 수정한 후 마지막으로 컴퍼넌트를 재사용하거나 새로 개발한 코드 부품과 합성하여 요구에 맞는 새로운 소프트웨어를 구축한다.[10,11]

3. 시스템 설계 및 구현

3.1 시스템 개요



(그림2) FSA 시스템 구성도

본 논문에서는 통합된 소프트웨어 아키텍쳐 생성 프레임워크(Framework for Software Architecture)를 제안한다. FSA는 (그림2)와 같이 재사용 가능한 컴퍼넌트를 저장할 수 있는 라이브러리와 이 라이브러리의 효율적 관리를 위한 라이브러리 관리 시스템, 그리고 사용자들의 컴퍼넌트 검색·편집·생성을 지원해주기 위한 재사용 시스템으로 구성된다.

서브 시스템에 대한 기능은 다음과 같다.

1) 라이브러리 관리 시스템
   : 컴퍼넌트 등록기능, 분류·인덱싱 기능, 평가 기능, 저장·버전관리 기능
2) 재사용 시스템
   : 검색기, 이해기, 편집기, 생성기

먼저, 등록기는 재사용 가능토록 생성되어 있는 임의의 컴퍼넌트들을 라이브러리 관리 시스템으로 받아들이는 작업을 한다. 분류·인덱싱기에서는 분류 체계를 정의하여 등록기로부터 등록된 컴퍼넌트들을 분류한 후 인덱싱을 한다. 이때, 재사용 컴퍼넌트들은 다른 컴퍼넌트들과 쉽게 구분될 수 있도록 그 특성이 잘 표현되어야 하며, 원하는 컴퍼넌트를 쉽게 찾을 수 있도록 체계화되어야 한다. 또한 확장이 쉽고 적절한 컴퍼넌트의 부재시 유사 컴퍼넌트를 제공할 수 있도록 분류되어야 한다.

컴퍼넌트의 분류 기법에는 enumerative 기법과 facet 기법으로 나누어 언급된다. enumerative 분류법은 넓은 영역의 지식을 작게 분해해 나가면서 각 요소를 분류하고 그 요소들간의 계층적인 관계를 표현하는 방법이며, facet 분류법은 분류하려는 요소들 가운데서 공통적인 용어들을 하나의 facet으로 구성하고, 여기에서 추출한 용어들을 조합하여 요소들을 식별해내며 각각 다른 계층에 속한 두 개 이상의 개념을 관련짓는 방법이다. 전자의 경우, 요소들간의 상호관계를 잘 표현하지만 확장에는 어려움이 많으므로 FSA에서는 확장성을 지원하는 facet 분류 기법을 따른다.

평가 도구에서는 라이브러리에 저장될 컴퍼넌트의 품질을 평가하는 도구로써 컴퍼넌트의 분류와 인덱싱에 대한 정확성과 무결성, 그리고 각 컴퍼넌트의 응집력과 결합력에 대해 평가하며, 기존 컴퍼넌트와의 유사성까지 평가한다. 만일 기존의 컴퍼넌트로 부터 갱신된 컴퍼넌트가 있을 경우 이들은 저장단계에서 새로운 컴퍼넌트로 저장하는 것이 아니라 버전 관리기에서 기존의 컴퍼넌트를 갱신시켜준다.

사용자는 먼저 개발하고자 하는 소프트웨어의 요구사항을 분석하여 소프트웨어의 구조를 설계한 후 재사용 시스템내의 검색기를 통해 새로운 소프트웨어에 사용되어질 적절한 컴퍼넌트를 라이브러리로 부터 찾아낸다. 검색은 동의어 처리 및 사전 관리를 포함하는 질의 형식의 검색과 사용자의 불명확한 질의에 대해 해당 도메인에 적합한 컴퍼넌트를 선별 제시하여 필요한 컴퍼넌트에 접근하도록 도와주는 지정 검색, 하이퍼 링크 검색등으로 구성된다. 만일 적절한 컴퍼넌트의 검색에 실패한 경우, 편집기를 통해 새로운 소프트웨어에 적절한 컴퍼넌트를 개발하여 이후의 재사용을 위해 라이브러리에 저장될 수 있도록 한다. 그렇지 않고 적절한 컴퍼넌트를 검색한 경우, 해당 컴퍼넌트를 자신의 프로그램으로 통합하기 전 잠정적으로 결정되어진 컴퍼넌트에 대한

이해가 필요하다. 이를 위해 재사용 되어질 컴퍼넌트의 이름, 원시코드, 상속관계등에 대한 정보를 제공해 줌으로써 사용자의 오해로 인한 잘못된 재사용을 방지할 수 있다.

선택된 컴퍼넌트에 대해 충분히 이해 되었으면 이를 새로운 소프트웨어의 설계 양식에 맞게 조절하고 수정하는 작업이 필요하다. 이와같이 편집기에서는 새로운 컴퍼넌트의 개발뿐 아니라 재사용 되어질 컴퍼넌트에 대한 편집 기능을 지원해 준다. 생성기에서는 컴퍼넌트들의 재사용으로 응용 프로그램을 생성해내는 반면, 검색 실패로 새롭게 만들어낸 컴퍼넌트는 라이브러리 관리 시스템을 통해 재등록된다.

3.2 FSA 실행 예

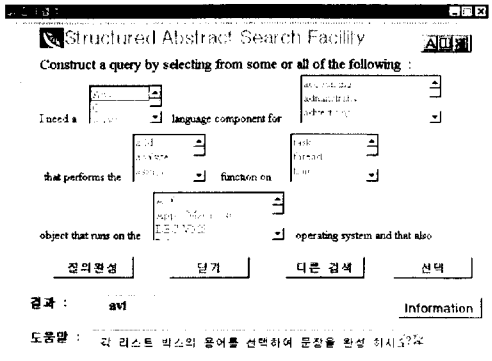
3.2.1 컴퍼넌트 검색을 위한 질의 작성 서브 시스템

(1) 질의 검색

(그림3)은 SA(Srtuctured Abstract) 메소드로 구축된 질의 검색을 위한 사용자 인터페이스이다. 이 메소드는 일반적인 부품 표현 문장을 구조적인 추상화를 통해 도메인(Domain), 함수(Function), 객체(Object), 운영체제(Operation system), 언어(Language)의 기능적 특성을 인덱스로 하여 구축된 템플리트이다. 논리적이고 해석적인 형식으로 소프트웨어에 관한 정보를 제시하는데, 같은 순서로 같은 종류의 정보를 나타냄으로써 컴퍼넌트 분류와 프로그램 이해에 관한 문제를 해결한다. 또한 자연적인 텍스트 프리젠테이션을 사용므로 직접 컴퍼넌트 기능에 접근할 수 있다.

(2) 동의어 처리를 통한 검색

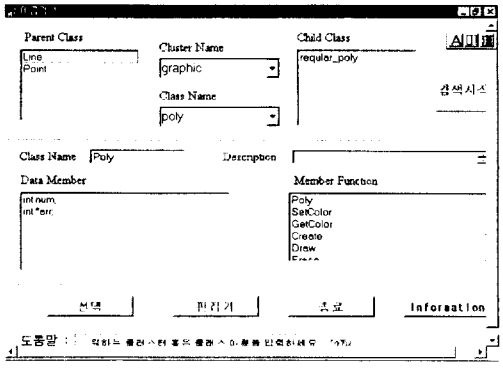
동의어 처리는 시스템과 재사용자 간의 어휘적 의미 차이를 최소화하고 명확하지 않는 요구를 구체화함으로써 사용자가 원하는 질의를 쉽게 작성할 수 있도록 지원한다. 이는 구조적이고 관념적인 코드 관점의 정보를 지식기반 관점에서 기능적 특성으로 통합한 것이다. FSA는 동의어 사전 관리 즉, 동의어의 추가, 삭제, 찾기를 사용자에게 허용하여 응용 도메인에 적합한 특화된 동의어를 갖도록 한다.



(그림3) SA 메소드의 질의 검색 UI

(3) 지정 검색

컴퍼넌트 라이브러리를 이용한 재사용 시스템에서 검색이나 관련 이해 정보의 제공이 비록 중요하지만 무엇보다도 이런 제반의 기능을 수행할 수 있는 정확한 사용자 요구 표현이 선행되어야 한다. 사용자가 막연하며 불투명한 컴퍼넌트 개념으로 검색을 시도할 경우도 도메인에 적합한 부품을 선별 제시함으로써 점차적으로 응용에 필요한 부품의 검색으로 접근 가능 하도록 유도해야 한다. FSA에서는 이를 위해서 잘 분류되어진 항목들의 리스트를 제공하여 사용자가 추상적 의미의 개념을 검색의 구체화된 항목으로 대치시키도록 이름 검색, 패싯 검색 그리고 부울리언 검색 세 가지를 제공한다.



(그림4) 이름 검색

① 이름 검색

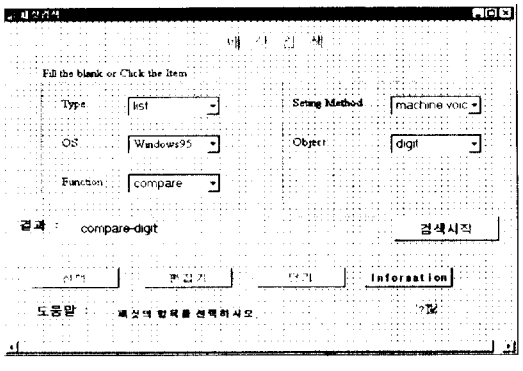
사용자가 열거형 방식에 의해 분류된 응용 도메인을 의미하는 클러스터 항목을 선택하면 해당 클러스터에 속하는 클래스의 이름이 나열된다. 여기서는 포인팅 장치에 의한 항목 선택 뿐 아니라 확신하는 부품의 이름을 직접 타이핑함으로써 부품을 획득할 수 있다.(그림4)

② 패싯 검색

(그림5)는 부품의 공통적인 속성을 모아 하나의 패싯으로 모으고 여러 패싯 중 찾으려 하는 컴퍼넌트에 알맞은 속성을 나타내는 항목들을 선택, 합성함으로써 특정 부품을 검색하는 FSA의 패싯 분류에 의한 인터페이스이다. 인터페이스에 나타난 패싯은 부품의 효과적인 표현이 가능하도록 항목의 대표성을 지닌 패싯으로 결정한다.

③ 부울리언 검색

부울리언 검색은 원하는 항목을 복수의 색인어로 표시하고 색인어 간의 관계를 파악하여 질의 항목을 AND, OR, NOT 등의 부울리언 연산자로 조합하여 질의를 작성하는 방식이다. 사용자는 제시된 키워드를 선택하고 각 키워드 간의 관계를 부울리언 연산자 버튼을 클릭함으로써 원하는 질의를 완성한다. 작성된 질의는 다음 검색에서 시행 오류를 피하며 신속성과 재사용성을 고려하여 질의어 히스토리에 나타낸다.



(그림5) 패싯 검색

(4) 선택된 컴퍼넌트에 대한 DB

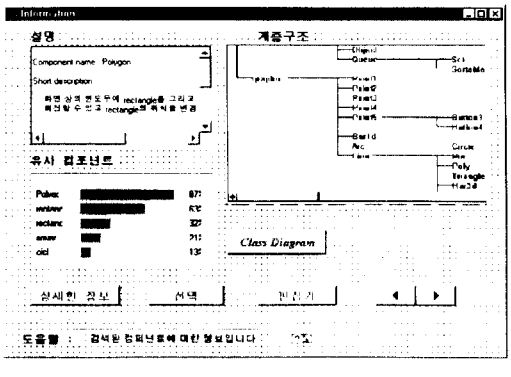
검색된 부품은 각 검색 윈도우의 선택 버튼을 누르므로써 선택 데이터베이스에 등록되어진다. 이것은 잠정적인 검색 결과를 수정할 필요가 있을 때 편집기상에서 검색된 부품의 히스토리를 통해 해당 부품의 소스 코드를 로드할 수 있도록 한다. 그러므로 통합된 재사용 시스템에서 검색, 편집, 재구축 단계가 일관성 있게 연계될 수 있다.

3.2.2 컴퍼넌트 이해를 위한 정보제공 서브시스템

재사용될 컴퍼넌트에 대한 사전 이해를 위해 FSA에서는 사용자의 이해도에 따라 컴퍼넌트 정보를 2단계에 걸쳐 나타내며 즉각적이고 구체적인 이해를 위해 시각적 다이어그램도 제공하는데 그 형태는 파일, 하이퍼링크, 텍스트 그리고 그래픽이다.

FSA에서 제공하는 정보는 다음과 같다.

- Class information : 검색된 클래스 자체에 대한 메타 정보로 기능적 서술과 외부와의 인터페이스 함수, 컴퍼넌트 등록 정보 등이 제시된다.
- Class inheritance hierarchy : 클래스 상속 구조를 하이퍼링크로 제공 한다.
- Source code : 실제 재사용 대상인 소스 코드를 제공하여 편집하고 재구축할 수 있도록 한다.



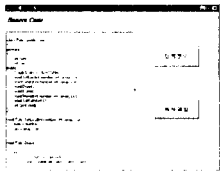
(그림6) FSA의컴퍼넌트에 관한 기본정보

(1) 검색된 부품의 기본 정보

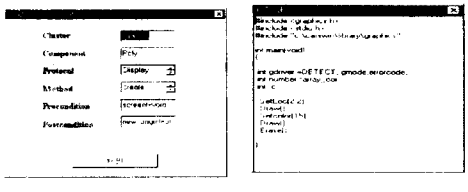
사용자 이해를 위한 첫 번째 수준의 정보로 검색된 컴퍼넌트 이름과 기능적 서술 그리고 외부 인터페이스를 갖는 서술적 정보를 제공한다. 또한 유사 컴퍼넌트에 대해 수치적이며 가시적인 표현 형식으로 그래프와 퍼센트(%)를 나타내며 기능적 유사성에 의한 계층 구조와 클래스 다이어그램을 나타낸다. (그림6)은 부품의 기본 정보이다.

(2) 상세 정보

(그림7)과 같이 상세정보로 원시코드, 속성상속 다이어그램, 컴퍼넌트 등록정보, 예제화일을 제공한다. 검색한 컴퍼넌트에 대한 충분한 이해가 없이 재사용된 부품은 사용자의 의도를 훌뜨릴 뿐 아니라 시스템의 기능적 착오를 유발시킬 수 있다. 그러므로 클래스의 인터페이스 수준에서 상세한 구현 수준까지의 정보를 시스템이 제공한다면 구체적이지 않는 재사용의 의도를 올바르게 정립시킬 수 있다.



(a) 원시 코드



(b) 등록 정보 (c) 예제 파일

(그림7) 상세 정보

3.2.3 부품의 편집과 구축을 위한 인터페이스

획득되어진 재사용 컴퍼넌트들은 사용자의 원래 요구를 충분히 만족 하도록 자신의 응용 영역에 맞게 새로이 편집 및 재구성하는 작업이 필요하다. FSA에서는 간단한 범용의 에디터인 편집기를 제공하는데 소스 코드의 편집과 재구축 작업은 하나의 연계된 일관된 작업이라 할 수 있으므로 편집기상에서 통합된 작업으로 처리 가능하다. 다음은 이 서브 시스템의 주요 기능이다.

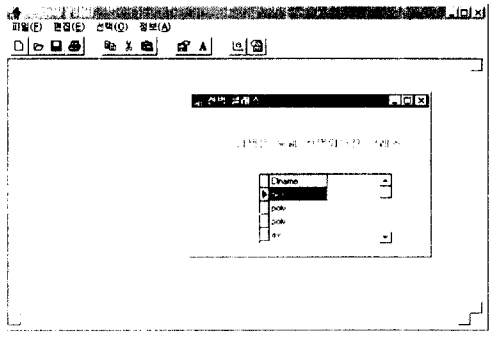
- Class Redefinement : 검색된 컴퍼넌트에 대한 정보를 바탕으로 새로운 클래스를 관련된 클래스들의 다중 상속을 통해 생성한다.
- Class Composition : 검색된 컴퍼넌트에 대한 소스 코드 윈도우를 참조로 일부 코드를 수정하여 새로운 클래스로 재구성 한다.
- C++ 컴파일러 호출 : 생성된 컴퍼넌트가 완전한 하나의 클래스로 등록하기 위해 MSVC의 통합 환경(IDE)을 불러 컴파일 한다.

부품의 편집 및 사용자에 의해 실질적으로 재사용 가능한 단일의 완전한 컴퍼넌트로 재구축하기 위해서 편집기에서는 데이터 변환 기법을 이용해 MSVC의 통합 개발 환경을 호출하여 데이터를 전달 한후 컴파일을 수행한다.

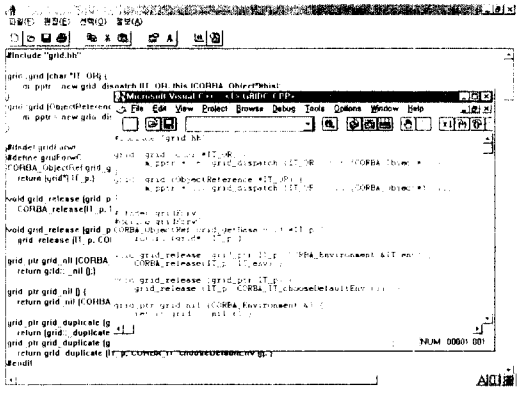
편집기의 메뉴 구성은 기본적인 에디터 형식을 따르고 있으나 2가지의 특별한 기능을 가진다. 하나는 검색기를 통해 검색된 컴퍼넌트 중 사용자가 잠정적으로 선택한 부품을 포함하는 선택 컴퍼넌트 데이터베이스를 제시하는 것이고 다른 하나는 편집기상에서 수정 작업이 완료된 재사용 단위인 C++ 소스 코드를 완전히 실행가능한 코드로 생성하기 위해 MSVC의 통합 환경을 호출하는 것으로 화면에서 마지막 두 개의 아이콘이 이 작업을 수행한다.

검색기의 검색 결과를 기록하는 선택 컴퍼넌트 데이터베이스는 (그림8)에 나타나 있다. 이 데이터베이스를 통해 검색과 편집 그리고 재구축을 일관성 있게 연관된 작업으로 수행할 수 있다.

(그림9)는 수정 완료된 부품의 재구축을 위해 MSVC의 IDE를 호출하고 여기로 데이터가 전달되어진 화면이다.



(그림8) 편집기상의 컴퍼넌트 선택 데이터베이스



(그림9) MSVC에서 부품의 재구축

4. 결론

최근 하드웨어의 대량 보급과 적용 분야의 확산으로 소프트웨어의 규모와 그 개발 비용이 매우 커져 가고 있으며 그중 유지보수 비용의 증가는 소프트웨어의 생산성 향상에 관심을 집중시키기에 충분했다. 생산성을 향상시키기 위한 많은 노력들중 재사용은 소프트웨어 개발시 컴퍼넌트 단위의 개발을 지원하여 이들 컴퍼넌트를 재사용함으로써 소프트웨어를 조립하여 완성시키는 방법이다.

따라서, 본 논문에서는 소프트웨어 재사용에 관한 여러 견해들에 대해 논하였고, 객체 지향 개발에 초점을 맞추어 보다 효과적으로 재사용 가능한 소프트웨어 부품의 구축에 대해 언급하였다. 또한 소프트웨어 컴퍼넌트에 대한 검색, 합성, 변형 시스템으로 소프트웨어 아키텍쳐 생성 프레임워크인 FSA를 제안하였다. 이는 사용자가 원하는 재사용 부품을 검색하고 수정하며 사용자의 관점에서 새로이 조절 가능하다. 즉, 통합된 재사용 시스템으로써 새로운 재사용 시스템 구축의 모델로서뿐 아니라 어플리케이션 생산에 적용으로 생산성과 품질을 향상시킬 수 있다.

향후 연구 방향으로는 재사용 대상을 소스 코드와 제한적인 문서 인덱싱에 의한 집의 수준에서 설계 정보 및 그 이상의 추상화된 수준의 정보까지 확장하고 유사 부품 검색을 위한 보다 효율적인 검색 알고리즘을 적용하며 라이브러리 구축시 지식 기반의 정보를 활용함으로써 재사용 부품이 응용 개발의 모든 과정을 위해 사용되어질 수 있는 재사용 프레임워크 개발이 필요하다. 아울러 생산된 부품에 대한 품질 평가 도구를 가지는 통합 CASE로의 발전이 요구된다.

【참고문헌】

[1] Hafedh Mili, Fatma Mili, and Ali Mili, "Reusing Software: Issues and Research Directions", IEEE Transactions on Software Engineering, Vol.21 No.6, 1995, pp.528-562

[2] Freeman P.,"Reusable Software Engineering: Concepts and Research Directions", ITT proceeding of the workshop on Reusability in Programming, 1983, pp.129-137,

[3] Ted Biggerstaff, Charls Richer, "Reusability Framework, Assessment, and Directions", IEEE Software, 1987, pp.41-49

[4] 이경환, "소프트웨어 재사용을 위한 객체 모델링 기법", 교학사, 1993,

[5] 이경환 외14, "소프트웨어 재이용을 위한 연구", 과학기술처, 1989.12

[6] Norman L. Kerth, "Using Patterns to Improve Our Architectural Vision", IEEE Software, January, 1997, pp.53-60

[7] James O.Coplien, Bell Laboratories, "Idioms and Patterns as Archtectural Literature", IEEE Software, January, 1997, pp.36-42

[8] Ted Lewis, Erich Gamma, "Object-Oriented application frameworks", Manning Publications, 1995,

[9] Frank Buschmann, Regine Meunier "Pattern-oriented software architecture: A System of Patterns", John wiley & Sons, 1996

[10] 김행곤 외6, "프로토타이핑 지원 재사용 시스템 개발에 관한 연구", 중간보고서, 한국 전자통신 연구원, 1996.7

[11] 김행곤, 차정은, "객체지향 프로토타이핑 지원을 위한 컴퍼넌트 이해 시스템 개발에 관한 연구", 한국 정보처리학회, 4권 6호 1997