韓國 OA學會 論文誌
第3卷 第3號, 1998. 10.

98-3-3-1-7

클래스 라이브러리 관리 시스템의 설계 및 구현

장 영 권*

Design and Implementation of the Class Library Management System

Young-Kweon Jang*

요 약

본 논문은 소프트웨어를 재사용하기 위한 클래스 라이브러리 관리 시스템(CLMS; Class Library Management System)을 설계하고 구현에 관한 것이다. 통신 관련 소프트웨어의 특징은 소프트웨어가 방대하고, 다른 소프트웨어와 달리 개발이 완료되어도 끊임없이 새로운 서비스를 수용해야 하며 개발 후 유지보수 및 신규 서비스의 확장 및 추가가 용이해야 한다는 것이다. 이러한 특징 때문에 소프트웨어 재사용은 생산성과 품질 향상에 기여할 것이다. 클래스 라이브러리 관리 시스템은 객체지향 데이타베이스를 기반으로 클래스 등록기, 클래스 검색기, 클래스 항해기로 이루어져 있으며 원시 코드 뿐만 아니라 SDL 설계 명세서도 재사용 할 수 있도록 고안되었다.

Abstract

This Paper describes design and implementation of the Class Library Management System(CLMS) that is used to efficiently software reuse. Communication softwares are various and wide. They continually must be modified themselves for new demand, maintained previous modules, and extended for new service. Software reuse will have been enhanced of software quality and software developer's productivity. The CLMS consists of the Class Register, the Class Retriever, and the Class Browser. The CLMS considered reuse of source code and SDL design specifications.

* 벽성대학 사무자동화과 전임강사
논문접수:98.8.28. 심사완료:98.10.27.

I. 서 론

이미 개발된 소프트웨어들을 조사해 보면 대부분의 소프트웨어 개발자들이 아주 똑같이 작성하지는 않더라도 비슷한 문제를 해결하기 위해서 비슷한 프로그램을 여러 번 반복하여 작성하고 있다는 것을 알 수 있다. 현재 개발되고 있는 소프트웨어의 상당수는 이미 개발된 소프트웨어임에도 불구하고 조금씩 다른 환경과 요구 등에 의하여 다시 개발되고 있다.

1969년 McIlroy가 처음으로 기계나 전자 부품이 사용되는 방식과 같이 소프트웨어 부품(component)들의 카탈로그 개념을 제안하였다. 그 이후 소프트웨어의 생산성과 품질을 향상시키는 방안의 하나로서 소프트웨어 재사용(software reuse)에 대한 연구가 활발히 진행되고 있다. 소프트웨어를 마치 하드웨어의 부품처럼 재사용한다면 소프트웨어 개발 속도는 매우 빨라질 것이며 소프트웨어 위기 현상을 해결할 수 있다는 전망을 가능하게 할 것이다. 아직까지 현실 세계의 소프트웨어 개발에 재사용이 보편화되고 있지 않으나, 현재 이런 현상에 대한 문제점 제시와 다각적인 해결 방안이 연구되고 있다[1,2].

이전에 개발해 놓은 많은 소프트웨어를 재사용할 수 있다면 생산성 및 소프트웨어의 품질을 향상시킬 수 있다. 또한 재사용 소프트웨어의 사용은 다음과 같은 이점을 가진다.

- 소프트웨어 개발 시간이 단축된다
- 이미 테스트된 부품들을 사용하므로 시스템의 신뢰성이 향상된다
- 이미 입증된 소프트웨어 부품을 사용하므로 개발 부담이 감소된다

재사용에는 원시 코드를 재사용하는 방법과, 설계 단계에서의 설계 명세서와 요구 사항 분석 객체들을 재사용하는 경우가 있다. 일반적으로 원시 코드의 재사용은 손쉽고 표현이 잘 정립되어 있어 많이 사용되고 있으나 프로그래밍 언어와 운영체제, 그리고 응용 분야에 따라 다르기 때문에 재사용에 많은 한계를 가지게 된다. 한편 시스템 설계 단계에서의 부산물을 이용하는 것은 개발 환경이나 응용 분야에 관계없이 재사용이 가능하므로 더 다양한 재사용성을 제공할 수 있으나 일반적으로 잘 정립되어 있지 않다. 다행히도 통신 관련 소프트웨어는 SDL[3] 언어를 사용하여 형식적으로 설계하므로 설계 단계에서 높은 재사용성을 얻을 수 있다.

이에 본 논문은 통신 관련 소프트웨어를 설계 단계에서 재사용할 수 있도록 클래스 라이브러리 관리 시스템(CLMS: Class Library Management System)을 설계하고 구현하고자 한다.

II. 소프트웨어 분류와 검색

2.1 소프트웨어 분류

소프트웨어 재사용이 소프트웨어 개발에 중요한 접근 방식이 되기 위해서는 소프트웨어 부품들의 집합을 분류하는 것이 핵심이 된다. 분류는 한정된 영역 내에서 비슷한 특성을 갖는 객체들을 하나의 클래스로 묶고, 객체와 클래스들 간의 관련성을 표현하는 것이다. 한 클래스에 속하는 모든 객체들은 다른 클래스의 객체들이 갖지 못한 특성을 적어도 하나 이상 공유한다. 탐색과 검색 방법이 없으면 분류된 부품들의 집합은 쓸모가 없으므로 효과적인 검색 시스템에는 잘 정의된 분류 구조가 요구된다.

부품들의 집합이 소프트웨어 요구사항들과 관련된 애크리뷰트들에 의해서 조직된다면 불필요한 부품이 검색될 확률은 감소될 것이다. 이는 다시 부품의 이해와 수정 단계를 향상시킨다. 더욱이 분류와 검색 시스템은 사용자들이 부품들의 집합 안에서 매우 유사한 항목들을 구별할 수 있도록 지원해야 한다. 검색된 샘플은 사소한 구현상의 차이점만 있는 매우 유사한 여러 부품들을 포함 할 수 있다.

분류 체계는 구조적이고 제어된 인덱스 어휘(index vocabulary)에 근거하여 체계적인 순서를 생성하는 기법이다. 인덱스 어휘는 개념이나 클래스들을 나타내는 이름이나 심볼들의 집합으로서 클래스들 간의 관계를 표현하기 위해 체계적인 순서로 리스트된다. 현재 널리 사용되고 있는 분류 체계로는 열거(enumerative) 분류 방식과 패싯(facet) 방식이 있다[4].

열거 분류 방식에서는 객체들의 집합을 점차 좁은 클래스로 분할해 가면서 그들 사이의 계층적인 관련성을 표현하는데 중심을 둔다. 열거 분류에서는 모든 가능한 클래스들이 미리 정의된다. 열거 분류 방식의 대표적인 예로는 Dewey의 십진분류법이 있다. 반면에 패싯 분류 방식에서는 객체들의 공통적인 측면을 하나의 패싯으로 모으고, 여러 개의 각 패싯에서 검색하고자 하는 객체에 알맞은 용어(term)를 선택하여 조합함으로써 특정 객체를 식별할 수 있다.

열거 분류 방식에서는 객체의 위치가 계층 구조 내에서 결정되므로 객체들 간의 계층 관계가 자연스럽게 드러나지만 계층구조가 복잡해지면 확장에 어려움이 따른다. 이에 비하여 패싯 분류에서는 객체들이 갖고 있는 여러 속성들을 각 패싯 별로 나누어 이해하므로, 분류 대상에 따라 효과적으로 적용하기가 용이할 뿐만 아니라 확장성도 열거 분류 방식보다 좋다. 그러나 객체들 간의 관련성을 명시적으로 나타낼 수 없는 단점이 있다. 또한 일단 설계되면 분류 체계는 정적이고 고정된다.

2.2 소프트웨어 검색

정보 검색에 대한 현재의 접근 방식은 흔히 프리-텍스트 분석에서부터 제어된 어휘(controlled vocabulary) 방식 사이의 등급에 따라서 분류된다. 제어되지 않은 어휘(uncontrolled vocabulary)라고도 부르는 프리-텍스트 분석 방식은 원문에 있는 단어의 빈도수를 분석한다[5]. 관련된 키워드들이 통계적이고 위치적인 특성들에 의해서 자동적으로 추출되어 자동적인 인덱스를 형성한다. 자동적인 인덱싱은 어떤 항목을 나타내거나 정의하기 위해서, 선택된 용어들을 지정하는 것이다.

이 방식이 기본적으로 갖는 가정은 통계적인 분석을 정당화할 만큼 방대한 양의 원문 자료가 존재한다는 것이다. 이 기법은 책이나 학술지 기사와 같이 원문의 비중이 높은 문서에 비교적 효과적인 것으로 입증되었다. 이에 반하여 제어된 어휘 방식에서는 미리 정의된 키워드들의 집합이 인덱싱 용어로 사용된다. 키워드는 전문가가 추출하고 정의하며, 도메인에 관계된 개념들을 최적으로 기술하거나 표현하도록 설계된다.

이에 반하여 제어된 어휘 방식에서는 미리 정의된 키워드들의 집합이 인덱싱 용어로 사용된다. 키워드들은 전문가가 추출하고 정의하며, 도메인에 관계된 개념들을 최적으로 기술하거나 표현하도록 설계된다. 소프트웨어 부품들은 프리-텍스트 분석보다 제어된 어휘 방식이 유리한 다음과 같은 특징들을 갖고 있다.

- 원시 코드는 프리-텍스트 수준이 매우 낮다.
- 키워드의 의미는 보통 관례나 프로그래머의 선호에 따라서 지정된다.
- 부품이 수행하는 역할과 역할을 수행하는 방법이 분명하지 않다.
- 부품의 기능 기술시 사람의 관여가 필요하다.

제어된 어휘에서 키워드로 표현되는 개념들은 분류 체계에 따라 조직된다. 분류 체계는 미리 정의된 관계들의 네트워크를 제공하여 프리-텍스트 분석에서 나타내지 못하는 의미상의 정보를 표현할 수 있다.

III. 통신 관련 소프트웨어의 개발 과정

일반적인 통신 관련 소프트웨어 개발 과정은 그림 1을 따른다[6,7].

요구사항 분석(requirement analysis) 단계의 목적은 시스템의 문제 영역(problem domain)과 사용자 요구 사항을 파악하고 분석하여 문서화하는 것이다. 이 단계에서 생성되는 모델에는 텍스트 요구 사항 문서, 시스템이 정의되고 사용되는 다양한 방법을 나타내는 Requirement Use Case 모델, 문제 영역과 외부 요구사항의 공통된 이해를 위해 객체와 그들 간의 관계를 나타낸 OMT 객체 모델, 문제 영역과 요구 사항에 나

타나는 중요한 개념을 나열한 자료 사전(data dictionary)이 있다.

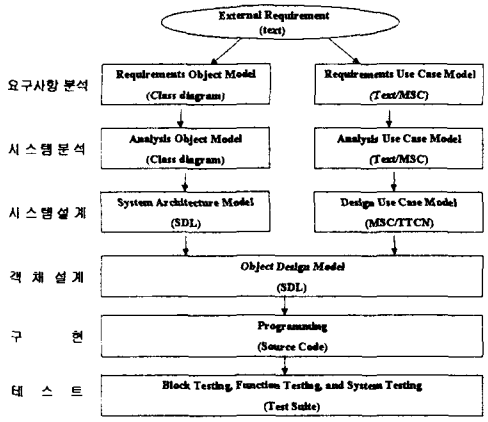


그림 1. 통신 관련 소프트웨어 개발 과정
Fig. 1. Development Course of the Communication S/W

시스템 분석(system analysis) 단계는 시스템 자체 구조와 요구된 기능을 충족 시키는데 필요한 객체를 찾아내는 것을 목적으로 한다. 이 단계에서 생성되는 모델에는, 문제 해결을 위해 필요한 객체를 표현하고 객체들 간의 구조를 기술하는데 사용되는 객체 모델, 객체들 간의 상호 작용을 표현하는데 사용되는 Analysis Use Case 모델, 객체 모델이나 Use Case 모델로 표현하기 어려운 내용을 결정 사항이나 시스템 제약 사항을 기술하는 텍스트 문서가 있다.

설계 단계는 시스템 설계(system design) 단계와 객체 설계(object design) 단계로 나누어진다. 시스템 설계 단계는 시스템 구현 구조를 정의하고 전체 설계 전략을 수립한다. 이 단계에서는 시스템 구조를 표현하는 SDL 시스템/블럭 다이어그램, 시그널과 원격 절차 호출(remote procedure call)을 이용한 SDL 인터페이스 정의, 시스템 내의 상이한 요소들 간의 동적인 상호 작용을 나타내는 MSC(Message Sequence Chart) 다이어그램이 존재한다. 시스템 설계 단계 후에는 객체 설계 단계를 수행한다. 이 단계의 목적은 완전하고 검증된 시스템 동작을 기술하는 것이다. 이 단계에서 사용되는 유일한 모델은 SDL 다이어그램이며, 특히 SDL 프로세스 다이어그램이 중심이 된다.

구현 단계는 적당한 프로그램 언어를 이용하여 코딩한다. 일반적으로 C 언어가 많이 사용되고 재사용성을 높이기 위해서는 객체지향 언어인 C++, CHILL-96[8]를 사용하는 것이 좋다.

시험 단계는 개발된 소프트웨어가 원하는 기능을 올바르게 수행하는지의 여부와 성능을 시험하는 단계로 블럭 시험, 기능 시험, 시스템 시험으로 나누어져 있다.

위에서 살펴본 바와 같이 통신 관련 소프트웨어 개발 과정의 각 단계에서는 OMT 클래스 다이어그램, MSC, SDL 설계 명세서, 소스 코드, 시험조(test suite) 등이 생성되고 이들은 모두 재사용의 대상이 될 수 있다. 본 논문에서는 설계 명세서와 소스 코드를 재사용 대상으로 하여 구현하였으며 다른 것들도 재사용 할 수 있도록 확장성 있게 설계하였다.

Ⅳ. 클래스라이브러리 관리시스템의 구현

일반적으로 클래스들은 다음과 같은 과정을 통해 새로운 소프트웨어 개발에 사용될 수 있다. 먼저 등록 단계에서는 일정한 절차에 따라 하나의 클래스에 대한 재사용성을 평가하고, 재사용성이 높은 클래스에 대해 특정 기능이나 도메인 특성에 따라 분류하고 데이터베이스에 등록한다. 이렇게 등록되어 재사용되는 클래스를 검색하기 위해 검색에 대한 요구를 특정 형태의 질의로 표현하면, 검색 단계에서는 사용자 질의와 클래스에 등록된 패싯 키워드나 클래스 기능 설명서에 있는 단어의 빈도수를 비교하여 연관성이 높은 순으로 찾아준다. 사용자는 이렇게 검색된 클래스들의 내용을 파악하면서 그 중에서 적합성을 평가한다. 이렇게 적합성 평가를 통해 선정된 클래스들은 개발될 소프트웨어 요구에 따라 적당한 수정을 통해 재사용된다. 이러한 과정을 효율적으로 지원하기 위해서는 도구의 개발이 절실하다.

클래스 라이브러리 관리 시스템은 재사용 부품인 클래스의 등록, 정보 관리, 검색, 항해 기능을 제공한다. 관리해 주는 정보들에는 개발자가 SDL 설계 문서나 소

스 코드에 관한 것 뿐만 아니라 저장된 소스에 관한 메타 정보가 포함된다. 이들 메타 정보에는 클래스의 기능, 클래스들 간의 상속 관계, 객체 식별 정보, 클래스의 특성 정보 등이 있다.

그림 2는 클래스 라이브러리 관리 시스템의 전체구조이다. 사용자는 Motif로 구현된 통합 사용자 인터페이스를 통해 시스템을 사용할 수 있다. 클래스 등록기를 통하여 클래스를 등록하고, 클래스 검색기를 통하여 질의를 통해 클래스를 검색하고, 클래스 항해기를 통하여 클래스의 상속 구조를 항해하면서 특정 클래스에 대한 정보를 볼 수 있다. 객체지향 데이타베이스에 저장된 클래스는 내부적으로 〈클래스 식별자, 설계 명세서, 구현 코드, 패싯 키워드, 클래스 설명서〉 형태로 표현된다.

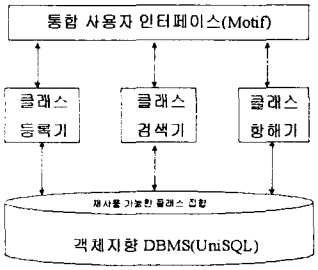


그림 2. 클래스라이브러리관리시스템의 구조
Fig. 2 Structure of the CLMS

그림 3에서 보는 바와 같이 통합 사용자 인터페이스는 메뉴바, 아이콘 팔레트, 검색 결과 영역, 액션 영역, 메시지 영역으로 이루어져 있다.

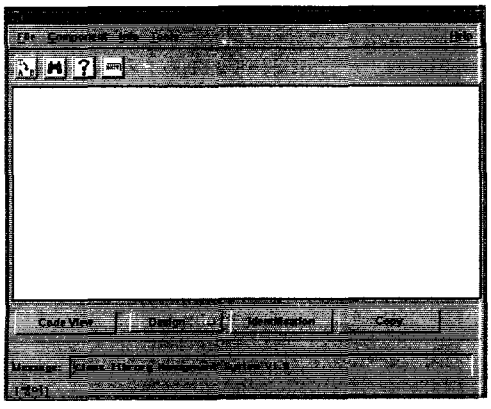


그림 3. 통합 사용자 인터페이스
Fig. 3. Integrated User Interface

아이콘 영역에는 메뉴 중에서 자주 쓰는 기능만을 모아놓은 것이다. 사용자가 마우스를 아이콘에 위치하면 메시지 영역에 아이콘의 기능이 나타난다. 각 아이콘의 기능은 좌로부터 클래스 등록, 클래스 검색, 도움말, 종료 기능을 나타낸다. 검색 결과 영역에는 검색식을 만족하는 클래스들이 나타난다. 액션 영역에 있는 버튼들은 검색 결과 영역에 나타난 클래스들에 대하여 동작하는 것들이다. 메시지 영역에는 사용할 때 발생한 오류 사항이나 지시 사항을 나타내는 영역이다.

그림 4는 클래스 등록기에 정보를 입력하는 예를 보여준다. 클래스 등록기에는 클래스 이름, 패싯 키워드, 클래스 기능, 화일 위치, 설계자, 버전, 등록일을 입력한다.

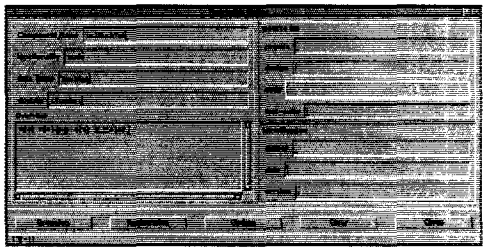


그림 4. 클래스 등록기
Fig. 4. Class Register

그림 5는 클래스 검색기를 보여준다.

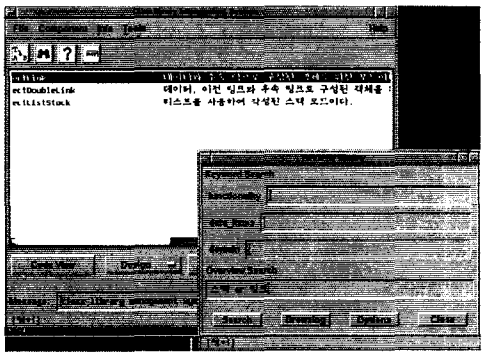


그림 5. 클래스 검색기
Fig. 5. Class Retriever

질의는 패싯 키워드 질의와 프리-텍스트 질의를 모두를 지원한다. 패싯은 기능, 데이터 항목, 응용 분야로 구분했다. 패싯 키워드 질의를 할 때 사용자는 등록된 키워드를 모르기 때문에 패싯 브라우저를 사용하여 키

워드를 선택할 수 있도록 하였다. 프리-텍스트 질의에서는 부울리언 연산자 and, or, not을 사용할 수 있다.

그림 6에 나타난 클래스 항해기는 클래스 상속 구조를 항해하면서 특정 클래스에 대한 상세 정보를 볼 수 있도록 해준다. 즉, 클래스 내의 상속 프로시듀어, 공개 프로시듀어 등의 메타 정보를 제공하고, 사용자에게 관련 클래스들간의 계층 구조와 관계를 그래픽 정보로 보여주는 것이다.

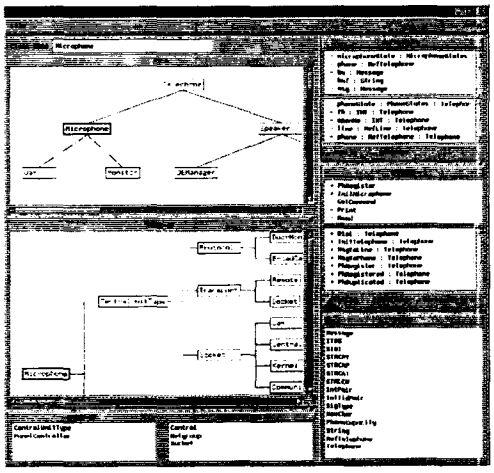


그림 6. 클래스 항해기
Fig. 6. Class Browser

V. 결 론

소프트웨어 재사용은 D.McIlroy가 30여년전에 처음 제안한 바와 같이 표준 부품들에 근거한 소프트웨어 공장의 개념에 훨씬 못 미치고 있다. 재사용의 정의는 초기의 교환 가능한 부품에서, 지식을 포함하여 소프트웨어 프로젝트에 연관된 모든 것을 재사용하는 개념으로 넓어지고 있다. 또한 재사용 공동체는 재사용이 기술적인 문제 뿐만 아니라 다른 여러 요인들도 포함하고 있다고 인식하게 되었다.

본 논문에서는 통신 관련 소프트웨어 개발 과정에서 생성되는 SDL 설계 명세서나 소스 코드를 재사용하는 과정을 지원하는 클래스 라이브러리 관리 시스템에 대해서 기술하였다. 클래스 라이브러리 관리 시스템은 객체지향 DBSMS에 클래스와 메타 정보를 저장해두고 클래스 등록기, 검색기, 항해기를 동작시키는 구조였다. 실제 사용 가능성을 확인해 보기 위해 100여종의 클래스를 구축하여 시험해 보았다.

대부분의 프로그래머들은 소프트웨어 재사용이 좋다는 것은 인식하고 있지만 실제로 활용하는 데는 많은 어려움을 느꼈다. 재사용 도구의 개발 뿐만 아니라 관리적인 측면에서의 지원이 반드시 요구되며 오랜 시일과 노력이 필요하다고 생각된다. 프로그래밍의 생산성과 품질을 향상시킬 수 있는 조치들은 일반적으로 소프트웨어의 재사용성도 높여 준다. 따라서 올바른 관리 방안은 프로그래머와 설계자에게 보다 나은 도구와 환경을 제공해야 할 것이다.

참 고 문 헌

[1] T. Biggerstaff and C. Ritchter, Reusability Framework, Assessment, and Directions, IEEE Software, pp.41-49, March 1997.

[2] C.W.Krueger, Software Reuse, ACM Computing Surveys, Vol.24, pp.131-183, 1992.

[3] ITU-T, Recomendation Z.100, Specification and Description Language(SDL), Geneva, March 1992.

[4] R.Prieto-Diaz, Implementing Faceted Classification for Software Reuse, Comm. ACM, pp.88-97, May 1991.

[5] G.Salton and M.Smith, On the Application of Syntatic Methodologies in Automatic Text Analysis, SIGIR89, Cambridge, Mass., pp.137-150, June 1989.

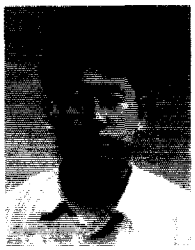
[6] C.S.Keum, et al. Integrated Environment

based on Object-Oriented Methodology for Real-time Systems, ISORC98, Kyoto, Japan, pp.284-288, April 1998.

[7] D.G.Lee, et al., A New Integrated Software Development Environment Based on SDL, MSC, and CHILL for Largescale Switching Systems, ETRI Journal, vol. 18, no. 4, pp.265-286, Jan. 1997.

[8] ITU-T, CCITT High Level Language(CHILL), Recommendation Z.200, Geneva 1996.

저 자 소 개



장 영 권

1994년 : 서울시립대학교 전산통계학과(이학사)

1996년 : 서울시립대학교 전산통계학과(이학석사)

1996년 ~ 현재 : 벽성대학 사무자동화과 전임강사