論文
10-3-4

소프트웨어 재사용성 향상을 위한 설계기법

A Design Technic for The Improvement of Software Reusability

송 월 봉(Worl-Bong Song)[1]

요 약

소프트웨어 재사용을 위한 콤포넌트의 식별은 사용자에게 적절한 콤포넌트가 발견되지 않았을 때 예비 콤포넌트에 대한 정보를 제공하는 안내역할을 수행하여야만 한다. 또한, 믿을 수 있는 소프트웨어 컴포넌트의 재사용은 각 새로운 어플리케이션에 대해 같은 컴포넌트를 재설계하거나 재 코딩하는 것보다 훨씬 위험이 적고 효과적이다. 본 논문에서는 먼저 재사용 프로세스에 대하여 알아보고 이어서 재사용 가능한 컴포넌트 구축을 위한 재사용 모형 및 구축방법을 분석하고자하며, 이를 이용한 재사용 가능한 컴포넌트에 대하여 제안하고자 한다. 이는 향후 효율적인 프로그램을 설계하고 작성하는데 도움이 될 것이다.

Abstract

Identification of reusable software components should guide user to come up with information about candidate components when a proper component is not found. Reusable software components which is able to confidential are a few risk and more effective than redesign or encode same components about each new application program. In this paper reusable processes are considered previously. Reusable model and the method for construction are analyzed for construction reusable components and reusable components which take this method are proposed. These problem will go far toward solving the design and coding of effective program.

논문 접수 : 2009. 5. 18.
심사 완료 : 2009. 6. 1.

1) 시립인천전문대학 컴퓨터정보과 교수

※본 논문은 인천전문대학 연구지원비에 의한 것임.

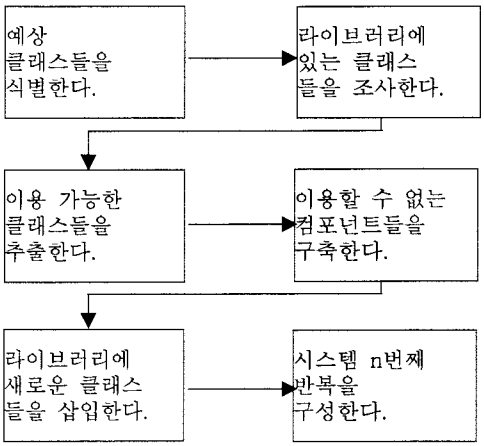
1. 서 론

기존의 소프트웨어를 활용함으로서 새로운 소프트웨어개발에 따른 인력을 줄이고 개발기간을 단축하며, 소프트웨어의 신뢰도를 향상시키기 위한 수단으로써 소프트웨어재사용에 대한 연구[2,3,4,5,6]가 활발하게 진행되고 있다. 소프트웨어 공학에서, 재사용은 오래된 아이디어이면서 또한 새로운 아이디어다.

소프트웨어 재사용에 관한 연구에서 재사용은 소프트웨어 제품 개발 시에 단순한 비용절감보다 더 많은 것을 얻게 해준다고 알려져 있다. 예로서, 믿을 수 있는 소프트웨어 컴포넌트의 재사용은 각 새로운 어플리케이션에 대해 같은 컴포넌트를 재설계 하거나 재 코딩하는 것보다 훨씬 위험이 적으며, 효율성 쟁점은 이전에 존재하는 모듈의 새로운 버전을 계속 최적화하는 것보다 차라리 재사용 가능한 컴포넌트의 집합을 최적화하는 데 관심을 집중하는 것이 효과적이다[4]. 그러나 재사용은 조직화된 장애물, 소프트웨어 재사용의 진정한 성격의 이해 부족, 그리고 재사용 기술을 촉진하고 구현하는 전략의 결여 및 취약 때문에 간혹 과장되기도 한다. 본 논문에서는 먼저 재사용 프로세스에 대하여 알아보고 이어서 재사용 가능한 컴포넌트 구축을 위한 재사용 모형 및 구축방법을 분석하고자하며, 이를 이용한 재사용 가능한 컴포넌트에 대하여 서술하고자 한다.

2. 재사용 프로세스

재사용은 모든 소프트웨어 프로세스의 통합 부분이 되어야 한다. [그림 1]은 "컴포넌트 어셈블리 모델"로서 재사용 가능한 컴포넌트의 라이브러리들이 전형적인 단계적 프로세스 모델에 어떻게 통합되는지를 보여주고 있다.



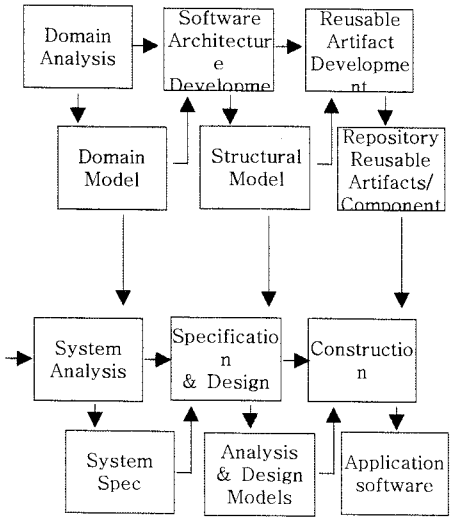
[그림 1] 컴포넌트 어셈블리 모형
[Fig.1] The model of component assembly

2.1 프로세스 모델

재사용을 위해 다양한 프로세스 모형들이 제안되었다. 이들 각각은 도메인 공학이 소프트웨어 공학과 동시에 발생하는 병렬 추적을 강조한다.

도메인 공학(domain engineering)은 소프트웨어 엔지니어가 재사용할 수 있는 소프트웨어 가공물의 집합을 설정하는 데 요구되는 작업을 수행한다. 이들 가공물들은 소프트웨어 공학으로부터 도메인 공학을 분리시키는 "경계"를 통하여 전달된다.

[그림 2]는 재사용을 명백하게 수용하는 전형적인 프로세스 모형을 설명하고 있다. 도메인 공학은 소프트웨어 공학 흐름에서 사용자 요구사항을 분석하는 기초로 사용되는 어플리케이션 도메인 모형을 생성한다. 소프트웨어 아키텍처는 어플리케이션 설계에 대한 입력을 제공해 준다.



[그림 2] 재사용을 강조한 프로세스 모형

[Fig. 2] The process model which stressed reuse

3. 재사용 가능한 컴포넌트 구축

재사용 가능한 소프트웨어를 생성하는 것은 마술과 같다. 추상화, 은폐, 기능 독립, 정제, 구조적 프로그래밍, 객체 지향 방법, 테스팅, SQA, 정확성 확인 방법 등과 같은 설계 개념 모두가 재사용 가능한 소프트웨어 컴포넌트의 생성에 기여한다.

3.1 재사용을 위한 분석과 설계

데이터, 기능, 행위 모형은 특정한 어플리케이션이 달성해야 되는 사항을 기술하기 위해 생성된다. 작성된 명세는 이들 모형을 기술하는데, 그리고 요구 사항의 완벽한 기술이 될 수 있게 사용된다.

분석 모형은 기존의 재사용 가능한 가공물을 지적한 모형의 컴포넌트들을 결정하기 위해 분석된다. 이 문제는 "명세 매칭(specification matching)"을 유도할 수 있는 형태인 요구 사항 모형으로부터 정보를 추출한다. 객체 지향 시스템에 대한 접근법[2]의 경우, 컴포넌트들은 이전의 어플리케이션에서 제품의 공학적 서술이 있는 각 클래스와 함께 추상화의 다양한 수준에서 명세 설계, 구현으로 정의되고 저장된다. 명세 지식(개발 지식)은 재사용 제안(reuse suggestion) 클래스의 형식으로 저장되고, 이것은 이들 서술의 기초로 재사용 가능한 컴포넌트를 검색하며, 검색 후에 이들을 합성시키고 조립하는 방향을 갖고 있다.

자동화 도구는 기존의 재사용 가능한 컴포넌트(클래스들)를 기술한 것과 함께 현재의 명세서에 언급된 요구 사항을 일치시키기 위해 저장소를 찾는 데 사용된다. 특성화 기능과 주요 키워드들은 잠재적인 재사용 가능한 컴포넌트를 찾는데 도움을 주기 위해 사용된다. 만약에 명세 매칭이 현재 어플리케이션의 필요 사항에 적합한 컴포넌트를 만들어 낸다면, 설계자는 재사용 라이브러리로부터 이런 컴포넌트를 추출할 수 있으며 또한 새로운 시스템의 설계에 이용할 수 있다. 만일 설계 요소가 발견될 수 없다면, 소프트웨어 엔지니어는 그것을 창안하기 위해 기존의 설계 방법이나 객체 지향 설계 방법을 적용해야 한다. 이것은 설계자가 새로운 컴포넌트를 생성하기 시작할 때 재사용을 위한 설계(DFR : design for reuse)가 고려되어야 한다.

재사용을 위한 설계는 소프트웨어 엔지니어에게 확실한 소프트웨어 설계 개념과 원리를 적용하도록 요구한다. 그러나 어플리케이션 도메인의 특성도 고려되어야 한다.

재사용을 위한 설계에서 기본으로 고려되어야 할 많은 주요 쟁점[3]들은 다음과 같다.

-. 표준 데이터(standard data) 어플리케이션 도메인이 조사되어야 하고, 또한 표준 전역 데이터 구조(예 : 파일 구조 또는 완전한 데이터베이스)도 식별되어야 한다. 모든 설계 컴포넌트들은 표준 데이터 구조가 사용될 수 있도록 특성화되어야 한다.

-. 표준 인터페이스 프로토콜(standard inter

face protocols)은 세 가지 수준이 설정되어야 한다. 즉, 내부 모듈 인터페이스 성격, 외부 기술적인(비인간적인) 인터페이스의 설계, 그리고 휴먼 기계 인터페이스이다.

-. 프로그램 형판(program templates) 구조 모형(structure model)은 새로운 프로그램 아키텍처의 설계(architectural design)에 형판 역할을 한다. 일단 표준 데이터, 표준 인터페이스, 그리고 프로그램 형판 등이 설정되면, 설계자는 설계를 생성하는 데 필요한 프레임워크를 갖게 된다. 이러한 프레임워크에 적합한 새로운 컴포넌트들은 다음에 재사용될 확률이 더 높아진다.

3.2 구축 방법

구축 기법 중 대표적인 예는 Netron[1]사가 개발한 프레임 기술(frame technology)이다. Netron사의 접근법은 프레임(frame)이라고 부르는 적응 가능하고 일반적인 컴포넌트의 집합을 정의한다. 객체처럼 프레임은 데이터와 연산을 캡슐화 시키지만, 소프트웨어 엔지니어가 프레임을 구성하는 어떤 서브컴포넌트를 선택, 제거, 수정, 또는 반복시켜 프레임을 적용할 수 있는 메커니즘과 작동됨으로써 객체의 정의를 확대시킨다.

어플리케이션은 프레임 계층으로부터 컴포넌트를 조립해서 구축된다. 계층의 하부에 있는 프레임은 분해된 아키텍처의 작업자 모듈과 유사하다. 즉, 그들은 하위 수준의 시스템 기능(예: 운영체제 상호작용, 인터페이스 구축, 데이터베이스 상호작용)을 수행하는 데이터 구조와 연산을 포함한다. 계층의 중간에 있는 프레임은 특정한 정보 시스템 도메인(예: 트랜잭션 프로세싱, 뱅킹, 고객 봉사)과 관련된 기능에 초점을 맞추고 있다. 계층의 최상위에서는 "시스템에 대한 마스터 청사진과 개발자가 시스템을 정의하기 위해 생성한 프레임"처럼 작동하는 "명세 프레임"이 있다. 재사용 가능한 소프트웨어 컴포넌트는 여러 방식으로 표현될 수 있지만, 이상적인 표현은 Tracz[6]가 3C Model이라고 부른 concept, content, 그리고 context이다. 소프트웨어 컴포넌트의 개념(concept)은 "a description of what the component does"[5]이다. 컴포넌트에 대한 인터페이스는 완전하게 기술되고 그리고 사전 및 사후조건의 문맥으로 표현된 의미(semantic)가 식별된다. 이 개념은 컴포넌트의 의도를 전달해야 된다.

컴포넌트의 내용(content)은 개념이 어떻게 실현되는지를 설명한다. 본래 내용은 컴포넌트를 수정하려는 사람에게만 알려질 필요가 있고 우연한 사용자에게는 은폐되는 정보이다.

문맥(context)은 응용성의 도메인 안에서 재사용 가능한 소프트웨어 컴포넌트를 마련한다. 즉, 개념적, 운영적, 구현 특징들을 명세화 시킴으로써 문맥은 소프트웨어 엔지니어가 어플리케이션 요구 사항을 위해 적절한 컴포넌트를 찾도록 할 수 있다. 실제적으로 사용되기 위해서는 개념, 내용, 문맥은 구체적인 명세로 변환되어야 한다. 재사용 가능한 소프트웨어 컴포넌트의 분류 기법에 관해 많은 논문과 기사들이 있다. 제안된 이 방법들은 3개의 주요 영역으로 분류될 수 있다. 즉, 라이브러리와 정보과학 방식, 인공지능 방식, 하이퍼텍스트(hypertext) 시스템 등이다. 오늘날까지 작업의 대다수는 컴포넌트 분류를 위한 라이브러리 과학 방식이 사용되었다.

4. 소프트웨어 컴포넌트에 대한 분류 기법

4.1 Enumerated classification

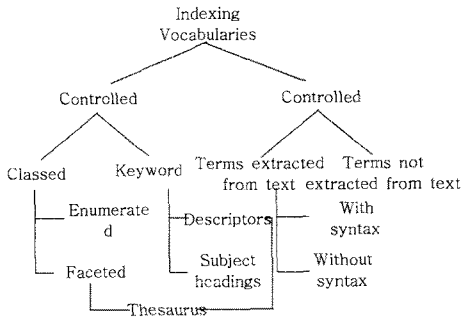
컴포넌트는 등급과 소프트웨어 컴포넌트의 서브클래스의 다양한 수준이 계층적 조직을 정의해서 나타낸다. 실제 컴포넌트들은 열거된 계층에서 어떤 경로의 최하위 수준에 나열된다. 예로서, 윈도우 동작들을 열거한 계층은 다음과 같다.

```
window operations
    display
        open
            menu-based
                    openWindow
            system-based
                    sysWindow
        close
            via pointer
                  ......
        resize
            via command
                setWindowSize, stdResize,
                shrinkWindow
            via drag
                pullWindow, stretchWidnow
            up/down      shuffle
                  .....
            move
                  .....
            close
                  .....
```

열거된 분류 기법의 계층적 구조는 그것을 이해하고 사용하기 쉽게 만들어 준다. 그러나 계층이 구축되기 이전에, 도메인 공학은 해당 계층에 적합한 엔트리들의 충분한 지식을 사용할 수 있게 수행되어야 한다.

4.2 Faceted classification

도메인 영역이 분석되고 나면, 기본적인 서술 특징들의 집합이 식별된다. 양상(facet)들이라고 부르는 이러한 특징들은 중요성에 따라서 우선순위가 부여되어 컴포넌트로 연결된다. 양상은 컴포넌트가 수행하는 기능, 처리되는 데이터, 그들이 적용되는 전후 관계, 또는 어떤 특징 등을 표현할 수 있다. 컴포넌트를 기술하는 양상들의 집합을 facet descriptor라고 부른다. 일반적으로 양상 기술은 7개에서 8개의 양상으로 제한된다.



[그림 3] 도서학의 인덱싱 방법에 의한 분류
[Fig. 3] The indexing sort method in the library

컴포넌트 분류에서 양상을 사용하는 간단한 예로 다음과 같은 양상 기술자의 사용기법을 고려해 보자.

{function, object type, system type}

양상 기술에서 각 양상은 일반적으로 서술적인 키워드인 하나 이상의 값을 갖고 있다. 예로서 function이 컴포넌트의 한 양상이라면, 이 양상에 할당된 전형적인 값은 다음과 같다.

function=(copy, from) 또는 (copy, replace, all)

다중 양상 값(multiple facet value)의 사용은 기본 함수 copy가 보다 완전하게 정제되도록 해준다. 키워드(값)들은 재사용 라이브러리에 있는 각 컴포넌트에 대한 양상의 집합에 할당된다. 소프트웨어 엔지니어가 설계를 위해 가능한 컴포넌트에 관해 라이브러리에 질의를 요청했을 때, 값들의 목록이 명시되고 라이브러리는 맞는 것을 찾아내기 위해 검색된다. 자동화 도구는 동의어(thesaurus function)을 통합시키는데 사용될 수 있다. 이것은 검색이 소프

트웨어 엔지니어가 명시한 키워드분만 아니라 이들 키워드에 대한 기술적인 동의어까지 포함할 수 있게 해준다.

양상 분류 기법은 도메인 엔지니어가 컴포넌트의 복잡한 기술을 명시하는 데 큰 유연성을 제공해 준다. 새로운 양상의 값들이 쉽게 추가될 수 있기 때문에 양상 분류 기법이 해당 열거형 접근법(enumeration approach)을 확장시키고 채택하는 것이 쉬워진다.

5. 결 론

소프트웨어의 재사용은 소프트웨어 품질, 개발자 생산성, 전체 시스템 비용 등에서 내재된 이익을 제공해 준다. 그리고 아직도 많은 장애물은 재사용 프로세스 모형이 산업체를 통해 널리 이용되기 전에 극복되어야 한다.

소프트웨어 재사용의 경제학은 한 가지 질문으로 설명한다. 미진하게 구축되었지만 더 재사용하는 것이 비용 절감을 하는 것인가? 일반적으로 그 답은 "yes" 이지만, 소프트웨어 프로젝트 계획자는 재사용 가능한 컴포넌트의 적응과 통합에 연관된 명확한 비용을 고려해야 한다.

즉 소프트웨어 공학 내에서 재사용될 수 있는 것과 재사용에 연계된 비용이 실제로 무엇인지를 우선 이해해야 한다. 그렇게 함으로써 재사용에 대한 효율성과 비용 대 효과 분석을 파악하는 것이 가능할 것으로 사료된다.

참 고 문 헌

[1] Bassett, P. G., What Is Frame Technology? Netron, Inc., Toronto, Canada, October 1994.

[2] Bellinzona R., M. G. Gugini and B. Pernici, "Reusing Specifications in OO Application," IEEE Software, March 1995, pp. 65~75.

[3] Binder, R., "Design for Reuse is for Real,"American Programmer, vol, 6, no. 8, August 1993, pp. 33~37.

[4] Frakes, W. B., and T. P. Pole, "An Empirical Study of Representation Methods for Reusable Software Components." IEEE Trans. Software Engineering, vol. 20, no. 8. August 1994, pp. 617~630.

[5] Liao, H., and Wang, F., "Software Reuse Based on a Laege Object-Oriented Library." Software Engineering Notes, vol. 18, no. 1, January 1993, pp. 74~80.

[6] Tracz, W., "Where Does Reuse Start?" Proc. Realities of Reuse Workshop, Syracuse University CASE Center, January 1990.

[7] Adler, R. M., "Emerging Standards for Component Software. IEEE Computer," vol. 28, no. 3, March 1995. pp. 68~77.

[8] Christensen, S. R., "Software Reuse Initiatives at Lockheed," Cross-Talk, vol. 8, no. 5, May 1995, pp. 26~31.

송 월 봉

1974년 숭실대 공학사
1982년 한양대 공학석사
1998년 순천향대학교
공학박사(전산학)
1978년～현재 시립인천전문대학 컴퓨터정보과 교수
관심분야 : 병렬처리컴파일러, 알고리즘