(19)대한민국특허청(KR)
(12) 등록특허공보(B1)

(51) 。Int. Cl.
G06F 9/45 (2006.01)

(45) 공고일자 2007년01월24일
(11) 등록번호 10-0673313
(24) 등록일자 2007년01월17일

(21) 출원번호 10-2004-0116775
(22) 출원일자 2004년12월30일
심사청구일자 2004년12월30일

(65) 공개번호 10-2006-0078095
(43) 공개일자 2006년07월05일

(73) 특허권자 재단법인서울대학교산학협력재단
서울특별시 관악구 봉천동 산 4-2

(72) 발명자 홍성수
서울특별시 영등포구 신길동 897-2 (28/2) 삼환아파트 105-703

박지용
서울 관악구 봉천6동 1685-4번지 305호

(74) 대리인 특허법인화우

(56) 선행기술조사문헌
KR1019980080502 A
US20030236794 A1 *
KR1020030056295 A
US20040261059 A1 *
* 심사관에 의하여 인용된 문헌

심사관 : 윤혜숙

전체 청구항 수 : 총 5 항

(54) 코드조각 번호 매김을 이용한 프로그램 간의 코드조각결합방법

(57) 요약

본 발명은 서로 다른 프로그램 간의 코드를 결합하는 방법에 관한 것으로, 각 프로그램의 코드조각에 대한 번호를 정의한 상태에서 코드조각 번호를 참조하여 특정 프로그램에 다른 프로그램의 코드조각을 삽입하거나 특정 프로그램의 코드조각을 다른 프로그램의 코드조각으로 대체하는 방법을 제공한다. 본 발명에 따르면 특정 프로그램 함수 내의 임의의 위치에 다른 프로그램의 어드바이스(코드조각)을 삽입(추가)하거나 특정 프로그램 함수 내의 코드조각을 다른 프로그램의 코드조각으로 대체(수정)할 수 있는 효과, 및 삽입/대체 대상 프로그램의 소스코드를 고치지 않고서도 두 프로그램의 코드조각을 결합할 수 있기 때문에 프로그램의 구조 및 가독성을 그대로 유지할 수 있고 프로그래머의 코드 변경에 따른 번거로움을 줄일 수 있는 효과가 있다.

대표도

도 3b

특허청구의 범위

청구항 1.

제1 프로그램의 코드조각에 대해 코드조각 번호가 정의된 상태에서 상기 제1 프로그램의 특정 코드조각 사이에 신규 코드조각이 삽입되면 상기 신규 코드조각에 상응하는 제2 프로그램을 생성하는 제 1 단계;

상기 제1 프로그램의 코드조각 삽입 위치의 윗 부분에 상응하는 코드조각 번호를 제1 코드조각 번호로 재정의하고, 상기 제2 프로그램의 코드조각 번호를 제2 코드조각 번호로 정의하고, 상기 제1 프로그램의 코드조각 삽입 위치의 아랫 부분에 상응하는 코드조각 번호를 제3 코드조각 번호로 재정의하는 제 2 단계; 및

상기 각각 재정의된 제1 프로그램 및 제2 프로그램의 각 코드조각 번호를 참조하여 번호가 가장 작은 코드조각 번호에 대응되는 코드조각부터 번호가 가장 큰 코드조각 번호에 대응되는 코드조각 순서대로 구조화하는 제 3 단계를 포함하는 코드조각 번호 매김을 이용한 프로그램 간의 코드조각 결합방법.

청구항 2.

제1 프로그램의 코드조각에 대해 메이저넘버 및 마이너넘버로 이루어진 코드조각 번호가 정의된 상태에서 상기 제1 프로그램의 특정 코드조각이 신규 코드조각으로 대체되면 상기 신규 코드조각에 상응하는 제2 프로그램을 생성하는 제 1 단계;

상기 제1 프로그램의 코드조각 대체 대상의 윗 부분에 상응하는 코드조각 번호를 제1 코드조각 번호로 재정의하고, 상기 제1 프로그램의 코드조각 대체 대상에 상응하는 코드조각 번호를 제2 코드조각 번호로 재정의하고, 상기 제1 프로그램의 코드조각 대체 대상의 아랫 부분에 상응하는 코드조각 번호를 제3 코드조각 번호로 재정의하고, 상기 제2 프로그램 코드조각 번호의 메이저넘버를 상기 제1 프로그램 대체 대상 코드조각 번호의 메이저넘버로 정의하고, 상기 메이저넘버가 변경된 제2 프로그램 코드조각 번호의 마이너넘버를 상기 제1 프로그램 대체 대상 코드조각 번호의 마이너넘버보다 크도록 각 코드조각 번호의 마이너넘버를 재정의하는 제 2 단계; 및

상기 각각 재정의된 제1 프로그램 및 제2 프로그램의 각 코드조각 번호를 참조하여 동일한 메이저넘버를 갖는 코드조각 중 마이너넘버가 큰 코드조각을 남기고, 메이저넘버가 가장 작은 코드조각 번호에 대응되는 코드조각부터 메이저넘버가 가장 큰 코드조각 번호에 대응되는 코드조각 순서대로 구조화하는 제 3 단계를 포함하는 코드조각 번호 매김을 이용한 프로그램 간의 코드조각 결합방법.

청구항 3.

제 1 항 또는 제 2 항에 있어서,

상기 제1 프로그램의 코드조각 번호 및 상기 제2 프로그램의 코드조각 번호는 각 프로그램 코드 상에서의 줄번호 및 열번호를 각각 포함하는 것을 특징으로 하는 코드조각 번호 매김을 이용한 프로그램 간의 코드조각 결합방법.

청구항 4.

제 3 항에 있어서,

상기 제1 프로그램의 코드조각 번호 및 상기 제2 프로그램의 코드조각 번호는 외부 파일에 각각 기록되고, 상기 외부 파일에 기록된 코드조각 번호를 변경하여 프로그램의 코드조각에 대한 코드조각 번호를 재정의하는 것을 특징으로 하는 코드조각 번호 매김을 이용한 프로그램 간의 코드조각 결합방법.

청구항 5.

제 1 항 또는 제 2 항에 따른 코드조각 번호 매김을 이용한 프로그램 간의 코드조각 결합방법을 실현시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체.

명세서

발명의 상세한 설명

발명의 목적

발명이 속하는 기술 및 그 분야의 종래기술

본 발명은 서로 다른 프로그램의 코드를 결합하는 방법에 관한 것으로, 더욱 상세하게는 각 프로그램의 코드조각에 대한 코드조각 번호를 각각 정의한 상태에서 코드조각 번호를 참조하여 특정 프로그램에 다른 프로그램의 코드조각을 삽입하거나 특정 프로그램의 코드조각을 다른 프로그램의 코드조각으로 대체하는 코드조각 번호 매김을 이용한 프로그램 간의 코드조각 결합방법에 관한 것이다.

최근에 프로그램의 개발/유지/보수 등을 용이하게 할 수 있는 영역 지향 프로그래밍(AOP; Aspect-Oriented Programming, 이하 "AOP"라 함)이 각광받고 있는데, 자금이체 프로그램을 예로 들어 AOP를 설명하면 다음과 같다.

자금이체 프로그램은 출금계좌, 입금계좌, 이체금액이 입력되면 SQL 문장을 수행시켜 해당 이체금액을 출금계좌로부터 입금계좌로 이체하는 코드로 이루어지는데, 자금이체를 위해서는 해킹방지용 프로그램이 사용자 컴퓨터에 탑재되어 있는지 검사해야 하고 정상 사용자인지 인증해야 하며 상대방 은행에서 제대로 이체내역을 처리하는지를 검사해야 하고 이체내역 로그를 시스템에 남겨야 된다. 즉, 자금이체가 이루어지기 위해서는 구현하고자 하는 자금이체 기능뿐만 아니라 보안, 인증, 로그 등과 같은 부가 기능들도 함께 구현되어야 되며, 이들 부가 기능을 위한 코드는 자금이체 프로그램이 아닌 시스템 상에 산재된 다른 여러 프로그램에 각각 기록되어 있다. 여기서, 구현하는 기능(즉, '자금이체')을 프라이머리 컨선(Primary Concern)이라 하고, 부가 기능(즉, '보안/인증/로그')을 크로스커팅 컨선(Cross-cutting concern)이라고 한다.

일반적으로, AOP가 적용되지 않은 객체 지향 프로그래밍 기법에서는 프라이머리 컨선을 구현한 프로그램의 소스코드에 크로스커팅 컨선을 구현한 프로그램의 코드를 포함시켰다. 이러한 방식에서는 프라이머리 컨선 또는 크로스커팅 컨선을 구현한 코드의 일부를 삽입/수정/대체/삭제하기 위해서는 프로그래머가 시스템 상에 산재된 각 프로그램의 각각의 코드를 일일이 직접 찾아서 변경시켜야만 된다. 즉, 이들 방식에서는 프라이머리 컨선과 크로스커팅 컨선이 하나의 프로그램 내에 구현되어 있기 때문에 프로그램 가독성이 떨어지고, 코드 변경이 어려우며, 그에 따라 프로그램 개발/유지/보수가 어려운 문제점이 있다.

이러한 문제점을 극복하기 위해 AOP가 대두되었으며, AOP에서는 크로스커팅 컨선을 어떻게 다룰지에 대한 프로그래밍 기법을 정의하고 있다. AOP에서는 크로스커팅 컨선을 구현한 코드["어드바이스(Advice)" 또는 "코드조각(code fragment)", 이하 "코드조각"이라 함]로 하나의 프로그램을 개발하고 프라이머리 컨선을 구현한 코드로 다른 하나의 프로그램을 개발한 상태에서 포인트컷(pointcut) 정보를 이용하여 두 프로그램을 결합(weaving)하는 것에 대한 명세를 표현하고 있다. 이하, AOP 기법이 사용된 프로그램을 "애스펙트 프로그램(aspect program)"이라 하고, 기존의 프로그래밍 기법이 사용된 프로그램을 "베이스 프로그램(base program)"이라 한다. 여기서, 포인트컷이란 애스펙트 프로그램의 특정 코드조각을 베이스 프로그램의 어느 위치에 둘 것인지에 대한 정보를 나타낸다.

도 1a 및 도 1b는 종래 방식에서 베이스 프로그램과 애스펙트 프로그램 간의 관계에 대한 일 실시예 설명도이다.

도 1a에 도시된 바와 같이, 애스펙트 프로그램에는 여러 코드조각과 각 코드조각에 대한 포인트컷 정보가 기술되어 있으며, 이러한 포인트컷 정보를 어떻게 기술하는지에 따라 프로그램의 개발/유지/보수가 용이해 짐을 알 수 있다.

종래 방식, 즉 기존에 개발된 AOP에서는 베이스 프로그램의 구성 요소 중 이름이 명명된 구성 요소에 대해서만 포인트컷 정보를 애스펙트 프로그램에 기술하고 있다. 여기서, 이름이 명명된 구성 요소로는 클래스, 함수, 변수 등을 들 수 있다. 예를 들어, 애스펙트 프로그램에 "x 함수" 및 "y 함수"가 구현되어 있고 베이스 프로그램에 a라고 명명된 함수 및 b라고 명명된 함수가 있다면 애스펙트 프로그램의 포인트컷 정보로는 "a 함수가 시작되기 전", "b 함수가 끝난 후"등이 될 수 있으며, 애스펙트 프로그램에는 "a 함수가 시작되기 전 x 함수 수행", "b 함수가 끝난 후 y 함수 수행" 등이 기술된다.

그런데, 상기와 같은 종래 기술에서는 베이스 프로그램의 구성 요소 중 이름이 명명된 구성 요소에 대해서만 포인트컷 정보를 애스펙트 프로그램에 기술할 수 있을 뿐 베이스 프로그램의 구성 요소 중 이름이 명명되지 않은 구성 요소(예; 함수(메쏘드) 내의 코드 등)에 대해서는 포인트컷 정보를 애스펙트 프로그램에 기술하지 못하는 문제점이 있다. 즉, 상기와 같은 종래 기술에서는 함수 내의 코드를 포인트컷 정보로 지칭할 수 없기 때문에 베이스 프로그램의 코드 중간에 애스펙트 프로그램의 코드조각을 삽입시켜 해당 코드조각에 상응하는 기능이 수행되도록 할 수 없다.

도 1b에 도시된 바와 같이, 종래 기술에서는 애스펙트 프로그램의 코드조각을 베이스 프로그램의 함수 중간에 삽입하는 것이 불가능하며, 단지 함수의 시작 부분 또는 함수의 끝 부분에 삽입하는 것만 가능하다.

한편, 상기와 같은 종래 기술에서 애스펙트 프로그램의 코드조각을 베이스 프로그램의 함수 중간 부분에 삽입할 수는 있다. 즉, 베이스 프로그램의 삽입 위치에 빈 함수(dummy function)를 호출하는 코드를 추가하거나 기 존재하고 있던 함수를 삽입 위치에서 두 개의 함수로 분할하여 새로운 함수의 시작 부분 또는 끝 부분에 코드조각을 삽입하면 된다. 그러나, 이러한 방식에서는 베이스 프로그램을 변경해야 하므로 프로그래밍 효율을 떨어뜨릴 뿐만 아니라 베이스 프로그램의 구조가 흐트러져서 프로그램의 가독성을 떨어뜨리는 문제점이 있다. 즉, 이러한 방식도 기존의 객체 지향 프로그래밍 기법과 마찬가지로 프로그래머가 일일이 코드를 직접 찾아서 수작업을 통해 변경시켜야만 된다.

발명이 이루고자 하는 기술적 과제

본 발명은, 상기와 같은 문제점을 해결하고 상기와 같은 요구에 부응하기 위하여 제안된 것으로, 각 프로그램의 코드조각에 대한 코드조각 번호를 각각 정의한 상태에서 코드조각 번호를 참조하여 특정 프로그램에 다른 프로그램의 코드조각을 삽입하거나 특정 프로그램의 코드조각을 다른 프로그램의 코드조각으로 대체하는 방법을 제공하는데 그 목적이 있다.

발명의 구성

이하 첨부된 도면을 참조하여 본 발명에 따른 바람직한 실시예를 상세히 설명한다. 하기에서 본 발명을 설명함에 있어 관련된 공지 기능 또는 구성에 대한 구체적인 설명이 본 발명의 요지를 불필요하게 흐릴 수 있다고 판단되는 경우에는 그 상세한 설명을 생략할 것이다. 그리고 후술되는 용어들은 본 발명에서의 기능을 고려하여 정의된 용어들로서 이는 사용자, 운용자의 의도 또는 관례 등에 따라 달라질 수 있다. 그러므로 그 정의는 본 명세서 전반에 걸친 내용을 토대로 내려져야 할 것이다.

도 2a 및 도 2b는 본 발명에 따른 코드조각 번호에 대한 일 실시예 설명도이다. 도 2a에는 베이스 프로그램의 코드조각에 대한 코드조각 번호가 XML 파일 A에 기록되어 있으며, 도 2b에는 애스펙트 프로그램의 코드조각에 대한 코드조각 번호가 XML 파일 B에 기록되어 있다.

도 2a에 도시된 바와 같이, 본 발명에서는 베이스 프로그램의 코드, 특히 함수 내의 코드조각에 이름을 부여하기 위해서 별도의 파일(예; XML 파일 등, 이하 "XML 파일"을 예로 들어 설명함)에 각 코드조각에 대한 코드조각 번호(즉 이름)를 정의한다. 여기서, XML 파일에는 베이스 프로그램의 소스코드(source code) 상에서 각 코드조각의 위치가 코드조각 번호(줄번호, 열번호)로 기록된다. 이처럼 베이스 프로그램의 코드조각에 대한 코드조각 번호를 XML 파일에 정의한 상태에서 베이스 프로그램의 특정 코드조각 사이에 애스펙트 프로그램의 코드조각을 삽입하고자 하면 베이스 프로그램의 소스코드를 고치지 않고서도 XML 파일의 일부만을 수정, 즉 XML 파일에 기록된 삽입 위치에 대응되는 코드조각 번호를 수정하고 이 수정된 코드조각 번호에 따라 새로운 코드조각을 삽입하면 된다.

도 2b에 도시된 바와 같이, 애스펙트 프로그램 역시 일종의 프로그램이므로, 본 발명에서는 애스펙트 프로그램의 함수 내의 코드조각에 대해서도 별도의 XML 파일에 각 코드조각에 대한 코드조각 번호를 정의한다. 여기서, XML 파일에는 애스펙트 프로그램의 소스코드(source code) 상에서 각 코드조각의 위치가 코드조각 번호(줄번호, 열번호)로 기록된다.

전술한 바와 같이, 본 발명에서는 XML 파일에 기술된 코드조각 번호를 참조하여, 베이스 프로그램의 코드조각 사이에 애스펙트 프로그램의 코드조각을 삽입하거나 베이스 프로그램의 코드조각 사이에 다른 베이스 프로그램의 코드조각을 삽입하거나 애스펙트 프로그램의 코드조각 사이에 베이스 프로그램의 코드조각을 삽입하거나 애스펙트 프로그램의 코드조각 사이에 다른 애스펙트 프로그램의 코드조각을 삽입할 수 있다.

따라서, 본 발명에서는 베이스 프로그램과 애스펙트 프로그램간의 관계가 동등하며, 이하 베이스 프로그램과 애스펙트 프로그램을 피쳐 프로그램(feature program)이라고 통칭하기로 한다.

이하, 본 발명의 이해를 돕고자 도 3a 내지 도 3c를 참조하여 피쳐 프로그램 A의 코드조각 사이에 피쳐 프로그램 B의 코드조각을 삽입하는 과정을 설명하고, 도 4a 내지 도 4c를 참조하여 피쳐 프로그램 A의 코드조각을 피쳐 프로그램 B의 코드조각으로 대체하는 과정을 설명하기로 한다.

도 3a 내지 도 3c는 본 발명에 따른 프로그램에 코드조각을 삽입하는 과정에 대한 일 실시예 설명도이며, 도 4a 내지 도 4c는 본 발명에 따른 프로그램의 코드조각을 대체하는 과정에 대한 일 실시예 설명도이다. 이하, 코드조각 삽입 대상(target) 및 코드조각 대체 대상 프로그램이 피쳐 프로그램 A인 경우를 예를 들어 설명하기로 한다.

앞서 언급한 바와 같이, 본 발명에서는 피쳐 프로그램 A의 코드조각 사이에 피쳐 프로그램 B의 코드조각을 삽입하는 것이 가능하고 피쳐 프로그램 A의 코드조각을 피쳐 프로그램 B의 코드조각으로 대체하는 것이 가능하다. 이를 위하여, 본 발명에서는 코드조각 번호를 메이저넘버(major number) 및 마이너넘버(minor number)로 정의한다. 메이저넘버는 피쳐 프로그램 A의 코드조각 사이에 피쳐 프로그램 B의 코드조각을 삽입하는데 사용되며, 마이너넘버는 피쳐 프로그램 A의 코드조각을 피쳐 프로그램 B의 코드조각으로 대체하는데 사용된다. 예를 들어, "3.4"라는 코드조각 번호 중 정수 부분인 "3"을 메이저넘버라 명명하고 소수점 아래 부분인 "4"를 마이너넘버라 명명한다.

도 3a에 도시된 바와 같이, 사용자가 본 발명이 적용된 프로그래밍 도구(이하 "프로그래밍 도구"라 함)를 사용하여 피쳐 프로그램 A에 관한 코드를 작성하면 프로그래밍 도구는 소스코드 파일 A를 생성하고 이 소스코드 파일 A의 코드조각에 대한 코드조각 번호를 1.0으로 XML 파일 A에 기록한다. 이 상태에서 사용자는 프로그래밍 도구를 사용하여 피쳐 프로그램 A의 특정 코드조각 사이에 새로운(다른) 코드조각을 삽입시킬 수 있다.

도 3b에 도시된 바와 같이, 사용자가 프로그래밍 도구를 사용하여 피쳐 프로그램 A의 특정 코드조각 사이에 새로운 코드조각을 추가적으로 작성(삽입)하면 프로그래밍 도구는 사용자가 작성한 코드조각에 상응하는 피쳐 프로그램 B를 생성한다. 여기서, 피쳐 프로그램 B는 사용자가 추가적으로 작성한 코드조각에 상응하는 소스코드 파일 B 및 이 소스코드 파일 B의 코드조각에 대한 코드조각 번호가 기록되는 XML 파일 B로 이루어진다. 이 때, 프로그래밍 도구는 XML 파일 A와 XML 파일 B를 수정하여 새로운 코드조각이 삽입된 상태를 표현한다.

즉, 프로그래밍 도구는 사용자에 의해 작성된 피쳐 프로그램 A의 소스코드 상의 특정 코드조각 사이에 소스코드 파일 B의 코드조각이 삽입됨을 알리는 의미로 피쳐 프로그램 B의 코드조각 번호를 2.0으로 설정하여 XML 파일 B에 기록한다(상대적 코드조각 번호). 그리고 XML 파일 A의 코드조각을 피쳐 프로그램 B의 코드조각이 삽입된 위치를 기준으로 두 개의 코드조각으로 나누고서 윗 부분에 해당되는 부분의 코드조각 번호를 1.0으로 변경하고 코드조각 삽입 위치의 아랫 부분에 해당되는 부분의 코드조각 번호를 3.0으로 변경하여 XML 파일 A에 기록한다.

여기서, 피쳐 프로그램 A의 소스코드 파일 A에 기록된 코드는 전혀 변경되지 않고서(즉, 사용자가 프로그래밍 도구를 사용하여 피쳐 프로그램 A의 소스코드에 새로운 코드를 추가하였더라도) 단지 이 소스코드 파일 A의 코드조각에 대한 코드조각 번호만이 XML 파일 A에서 변경되는 것이다. 이 상태에서 사용자는 프로그래밍 도구를 사용하여 피쳐 프로그램 A와 피쳐 프로그램 B를 결합시킬 수 있는 것이다.

도 3c에 도시된 바와 같이, 사용자가 프로그래밍 도구를 사용하여 피쳐 프로그램 A와 피쳐 프로그램 B를 결합시키면 프로그래밍 도구는 XML 파일 A 및 XML 파일 B를 참조하여 코드조각 번호의 순서대로 구조화시킨 피쳐 프로그램 C를 생성한다. 여기서, 새로이 생성된 피쳐 프로그램 C의 소스코드는 각 XML 파일 상에서 메이저넘버가 가장 작은 코드조각 번호에 대응되는 코드조각부터 메이저넘버가 가장 큰 코드조각 번호에 대응되는 코드조각 순서대로 정렬, 즉 피쳐 프로그램 A의 소스코드 중 코드조각 삽입 위치의 윗 부분인 코드조각, 이어서 피쳐 프로그램 B의 코드조각, 이어서 피쳐 프로그램 A의 소스코드 중 코드조각 삽입 위치의 아랫 부분인 코드조각으로 정렬된다.

한편, 마이너넘버를 사용하여 피쳐 프로그램 A의 특정 코드조각을 피쳐 프로그램 B의 코드조각으로 대체하는 과정을 설명하면 다음과 같다.

도 4a에 도시된 바와 같이, 사용자가 프로그래밍 도구를 사용하여 피쳐 프로그램 A에 관한 코드를 작성하면 프로그래밍 도구는 소스코드 파일 A를 생성하고 이 소스코드 파일 A의 코드조각에 대한 코드조각 번호를 1.0으로 XML 파일 A에 기록한다. 이 상태에서 사용자는 프로그래밍 도구를 사용하여 피쳐 프로그램 A의 특정 코드조각을 새로운(다른) 코드조각으로 대체시킬 수 있는 것이다.

도 4b에 도시된 바와 같이, 사용자가 프로그래밍 도구를 사용하여 피쳐 프로그램 A의 특정 코드조각을 삭제하고서 새로운 코드조각을 작성(대체)하면 프로그래밍 도구는 사용자가 작성한 코드조각에 상응하는 피쳐 프로그램 B를 생성한다. 여기서, 피쳐 프로그램 B는 사용자가 새로이 작성한 코드조각에 상응하는 소스코드 파일 B 및 이 소스코드 파일 B의 코드조각에 대한 코드조각 번호가 기록되는 XML 파일 B로 이루어진다. 이 때, 프로그래밍 도구는 피쳐 프로그램 A의 대체 대상 코드조각이 피쳐 프로그램 B의 코드조각으로 대체된 후에 피쳐 프로그램 A에서 보여질(구조화될) 순서대로 피쳐 프로그램 A의 코드조각(즉 대체 대상 코드조각 및 그 외 코드조각)에 대한 메이저넘버를 설정하고, 피쳐 프로그램 B의 코드조각에 대한 메이저넘버를 상기 대체 대상 코드조각에 대한 메이저넘버로 일치시킨 후에 피쳐 프로그램 B의 코드조각에 대한 마이너넘버를 피쳐 프로그램 A의 대체 대상 코드조각에 대한 마이너넘버보다 크게 피쳐 프로그램 A의 대체 대상 코드조각에 대한 마이너넘버 및 피쳐 프로그램 B의 코드조각에 대한 마이너넘버를 각각 설정한다.

즉, 프로그래밍 도구는 사용자에 의해 작성된 피쳐 프로그램 A의 소스코드 중 대체 대상 부분에 해당되는 코드조각이 소스코드 파일 B의 코드조각으로 대체됨을 알리는 의미로 피쳐 프로그램 A의 코드조각을 두 개의 코드조각으로 나눈다. 여기서, 하나의 코드조각은 대체가 이루어지기 전의 원래대로 남아 있는 코드조각이고 다른 코드조각은 피쳐 프로그램 B의 코드조각으로 대체되는 코드조각이다. 그런후, 원래대로 남아 있는 코드조각에 대한 코드 조각 번호를 1.0으로 설정하고(그대로 유지하고), 대체될 코드조각에 대한 메이저넘버를 2.0으로 설정하고 나서, 피쳐 프로그램 B의 코드조각에 대한 메이저넘버를 상기 대체 대상 코드조각에 대한 메이저넘버인 2.0으로 일치킨 후에, 피쳐 프로그램 B의 코드조각에 대한 마이너넘버를 피쳐 프로그램 A의 대체 대상 코드조각에 대한 마이너넘버보다 크게 설정한다. 즉, 피쳐 프로그램 A의 대체 대상 코드조각에 대한 마이너넘버를 2.1로 설정하여 XML 파일 A에 기록하고 피쳐 프로그램 B의 코드조각에 대한 마이너넘버를 2.2로 설정하여 XML 파일 A에 기록한다.

여기서, 피쳐 프로그램 A의 소스코드 파일 A에 기록된 코드는 전혀 변경되지 않고서(즉, 사용자가 프로그래밍 도구를 사용하여 피쳐 프로그램 A의 소스코드 중 일부를 삭제하고서 새로운 코드를 추가하였더라도) 단지 이 소스코드 파일 A의 코드조각에 대한 코드조각 번호만이 XML 파일 A에서 변경되는 것이다. 이 상태에서 사용자는 프로그래밍 도구를 사용하여 피쳐 프로그램 A와 피쳐 프로그램 B를 결합시킬 수 있는 것이다.

도 4c에 도시된 바와 같이, 사용자가 프로그래밍 도구를 사용하여 피쳐 프로그램 A와 피쳐 프로그램 B를 결합시키면 프로그래밍 도구는 XML 파일 A 및 XML 파일 B를 참조하여 동일한 메이저넘버를 갖는 코드조각 중 마이너넘버가 작은 코드조각을 마이너넘버가 큰 코드조각으로 대체하면서 메이저넘버의 순서대로 구조화시킨 피쳐 프로그램 C를 생성한다.

여기서, 새로이 생성된 피쳐 프로그램 C의 소스코드에는 각 XML 파일 상에서 동일한 메이저넘버를 갖는 코드조각 중 마이너넘버가 큰 코드조각이 남겨져 있고 메이저넘버가 가장 작은 코드조각 번호에 대응되는 코드조각부터 메이저넘버가 가장 큰 코드조각 번호에 대응되는 코드조각 순서대로 정렬, 즉 피쳐 프로그램 A의 대체 대상 코드조각의 윗 부분인 코드조각, 이어서 피쳐 프로그램 B의 코드조각 순서대로 정렬된다.

한편, 본 발명에서는 피쳐 프로그램 A의 하나의 코드조각 사이에 피쳐 프로그램 B의 하나의 코드조각을 삽입하거나 피쳐 프로그램 A의 하나의 코드조각을 피쳐 프로그램 B의 하나의 코드조각으로 대체하는 것을 예로 들어 설명하였으나, 피쳐 프로그램 A의 여러 코드조각 사이에 피쳐 프로그램 B의 여러 코드조각을 각각 삽입하거나 피쳐 프로그램 A의 여러 코드조각을 피쳐 프로그램 B의 여러 코드조각으로 각각 대체할 수 있다는 것을 당업자라면 쉽게 이해할 수 있을 것이다. 또한, 피쳐 프로그램 A 및 피쳐 프로그램 B로 이루어진 두 개의 프로그램을 결합하는 것을 예로 들어 설명하였으나, 다수의 피쳐 프로그램들을 선택적으로 조합하여 결합할 수 있다는 것을 당업자라면 쉽게 이해할 수 있을 것이다.

이상 본 발명의 내용이 실시예를 들어 설명되었으나, 본 발명의 실시예는 본 발명의 예시에 불과하며 본 발명의 범위를 제한하는 것으로 해석되어서는 안 된다. 본 발명이 속하는 분야의 기술자는 본원의 특허청구범위에 기재된 원리 및 범위 내에서 본 발명을 여러 가지 형태로 변형 또는 변경할 수 있다.

발명의 효과

상기와 같은 본 발명은 코드조각에 대한 코드조각 번호를 정의하고 이 코드조각 번호를 참조하여 두 프로그램의 코드조각을 결합함으로써, 특정 프로그램 함수 내의 임의의 위치에 다른 프로그램의 어드바이스(코드조각)을 삽입(추가)하거나 특정 프로그램 함수 내의 코드조각을 다른 프로그램의 코드조각으로 대체(수정)할 수 있는 효과가 있다.

또한, 본 발명은 삽입/대체 대상 프로그램의 소스코드를 고치지 않고서도 두 프로그램의 코드조각을 결합할 수 있기 때문에 프로그램의 구조 및 가독성을 그대로 유지할 수 있고, 프로그래머의 코드 변경에 따른 번거로움을 줄일 수 있도록 하는 효과가 있다.

도면의 간단한 설명

도 1a 및 도 1b는 종래 방식에 따른 서로 다른 프로그램 간의 관계에 대한 일 실시예 설명도.

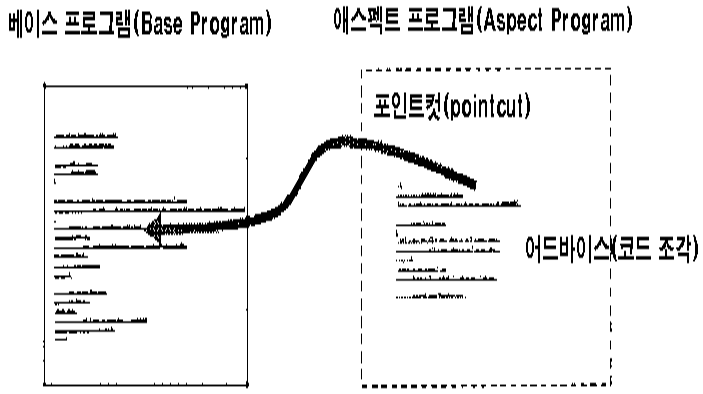
도 2a 및 도 2b는 본 발명에 따른 코드조각 번호에 대한 일 실시예 설명도.

도 3a 내지 도 3c는 본 발명에 따른 프로그램에 코드조각을 삽입하는 과정에 대한 일 실시예 설명도.

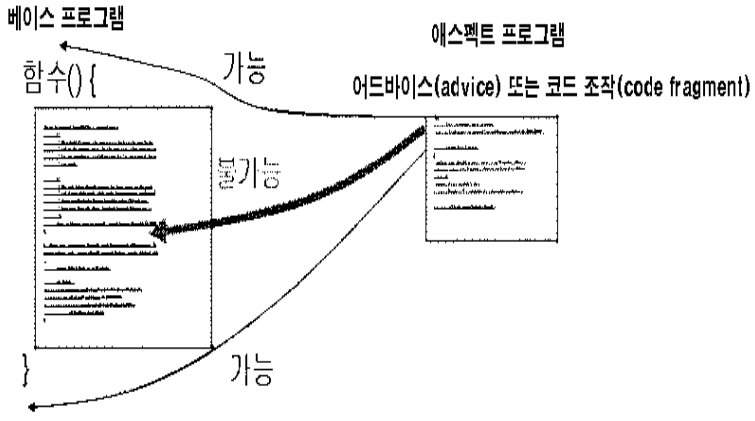
도 4a 내지 도 4c는 본 발명에 따른 프로그램의 코드조각을 대체하는 과정에 대한 일 실시예 설명도.

도면

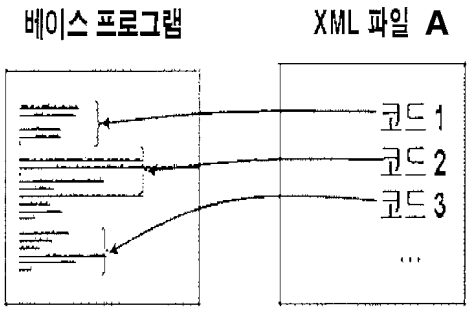
도면1a



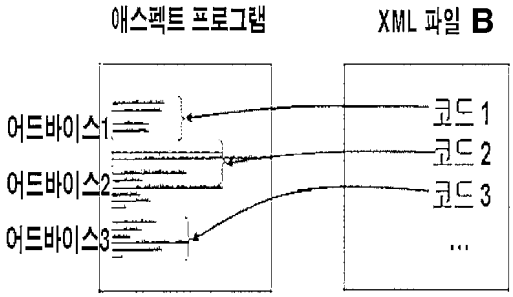
도면1b



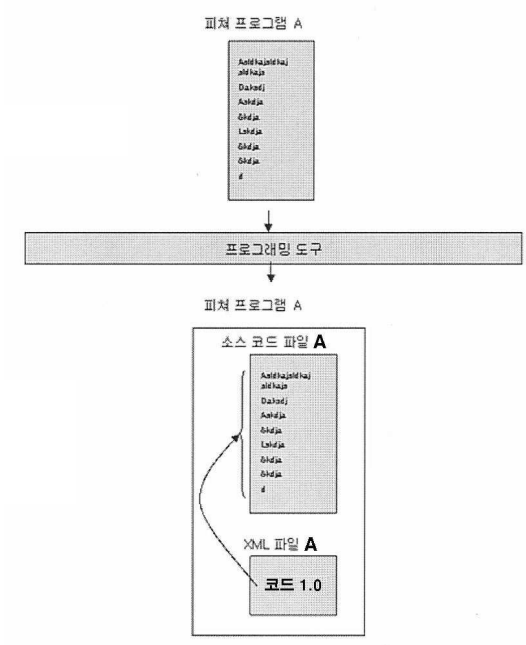
도면2a



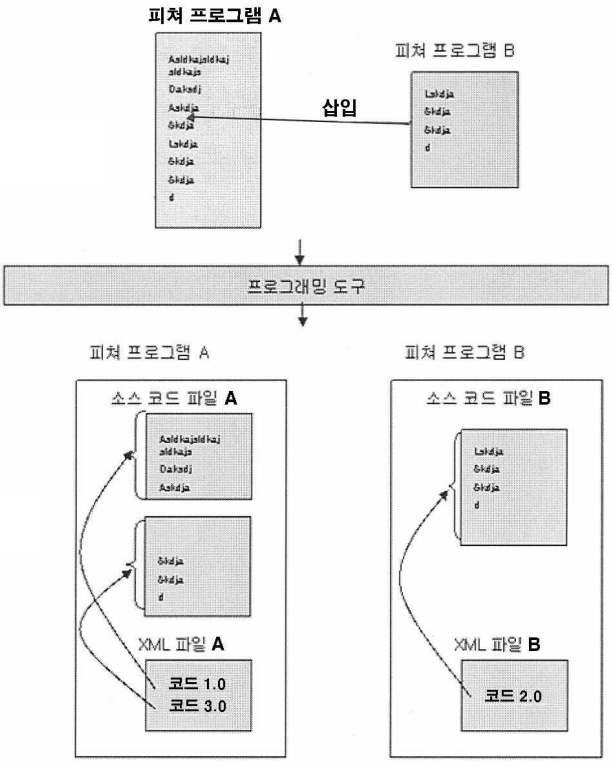
도면2b



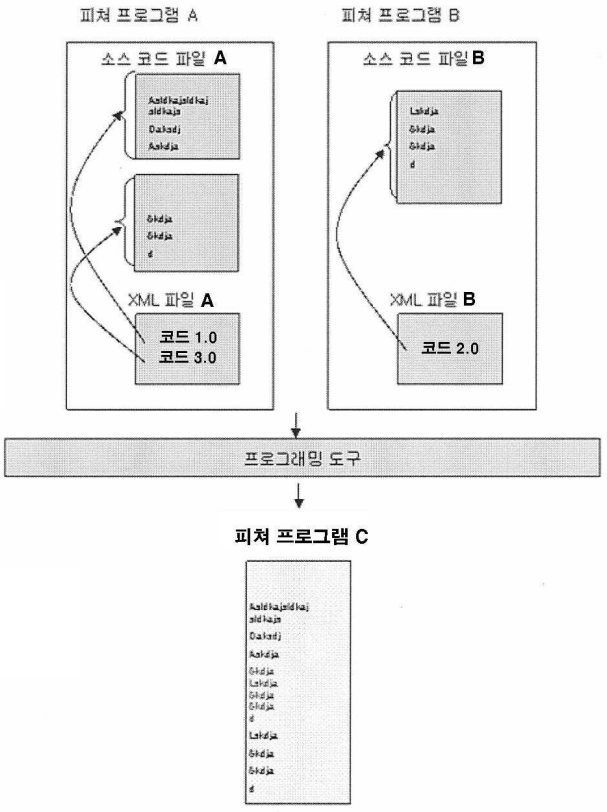
도면3a



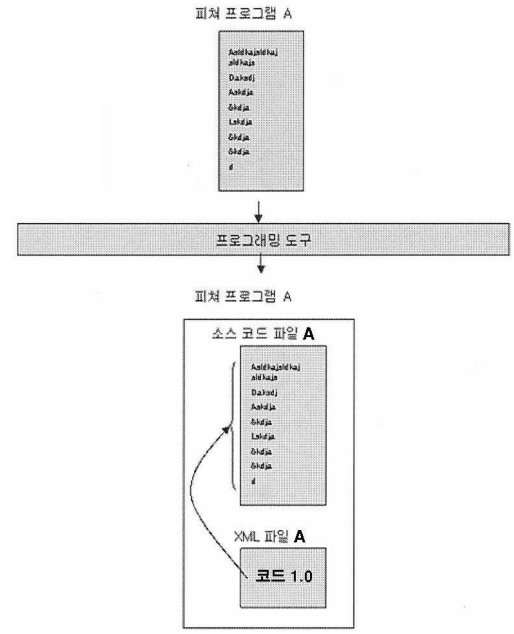
도면3b


피쳐 프로그램 A
피쳐 프로그램 B
삽입
프로그래밍 도구
피쳐 프로그램 A
피쳐 프로그램 B
소스 코드 파일 A
소스 코드 파일 B
XML 파일 A
코드 1.0
코드 3.0
XML 파일 B
코드 2.0

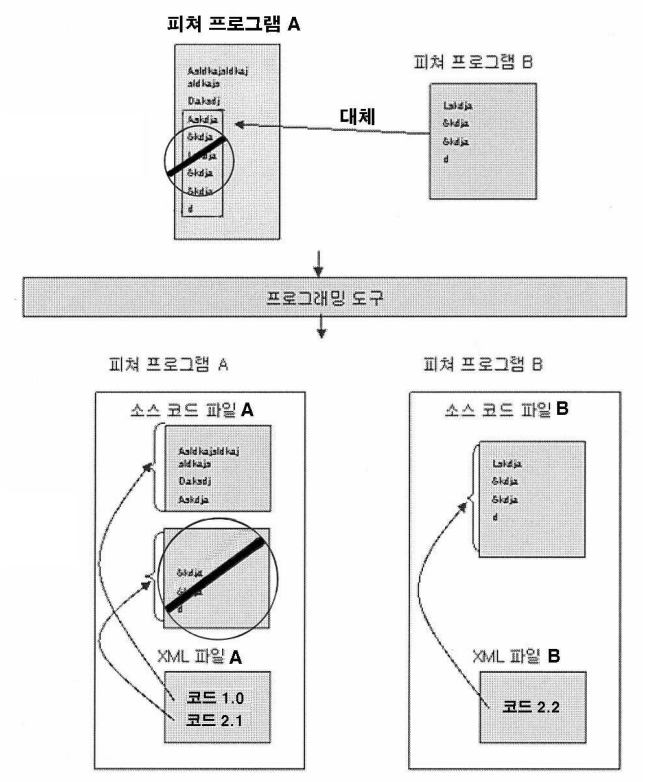
도면3c


피쳐 프로그램 A
피쳐 프로그램 B
소스 코드 파일 A
소스 코드 파일 B
XML 파일 A
코드 1.0
코드 3.0
XML 파일 B
코드 2.0
프로그래밍 도구
피쳐 프로그램 C

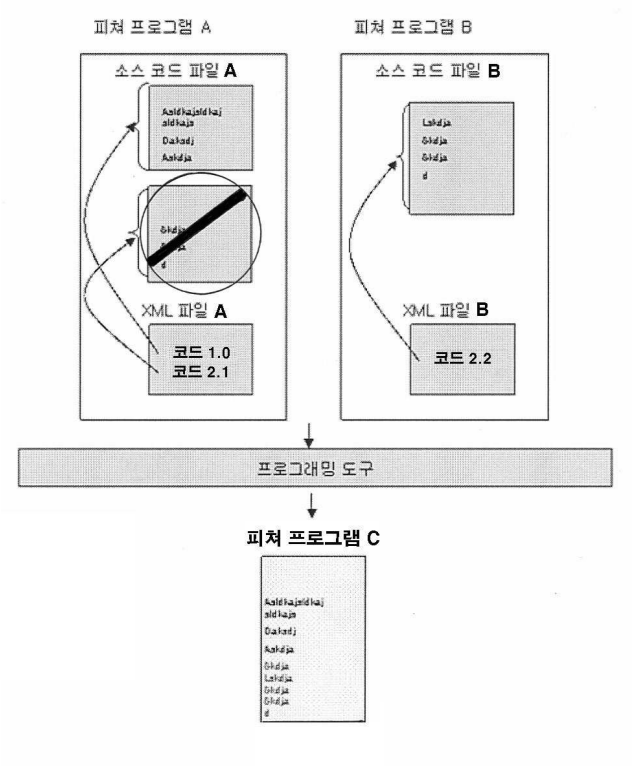
도면4a


피쳐 프로그램 A
프로그래밍 도구
피쳐 프로그램 A
소스 코드 파일 A
XML 파일 A
코드 1.0

도면4b


피쳐 프로그램 A
피쳐 프로그램 B
대체
프로그래밍 도구
피쳐 프로그램 A
피쳐 프로그램 B
소스 코드 파일 A
소스 코드 파일 B
XML 파일 A
XML 파일 B
코드 1.0
코드 2.1
코드 2.2

도면4c


피쳐 프로그램 A
피쳐 프로그램 B
소스 코드 파일 A
소스 코드 파일 B
XML 파일 A
코드 1.0
코드 2.1
XML 파일 B
코드 2.2
프로그래밍 도구
피쳐 프로그램 C