제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

컴포넌트 저장소를 위한 업데이트 엔진 설계

김지현**, 강병욱*
**영남대학교 컴퓨터공학과
*영남대학교 컴퓨터공학과
e-mail:dark1004@yumail.ac.kr

Design of Update Engine for Component Repository

Ji-Hyun Kim**, Byung-Ug Knag*
**Dept of Computer Engineering, Yeung-nam University
*Dept of Computer Engineering, Yeung-nam University

요 약

코드기반 재사용이 가능한 객체기반 프로그래밍(OOP : Object-Oriented Programming)의 문제점으로 인하여 높은 재사용성(Reuse)을 가진 컴포넌트 기반 소프트웨어(CBD : Component Based Development)의 개발에 관한 연구가 활발히 진행 중이며, 이미 개발된 컴포넌트들을 통합적으로 관리하는 저장소(Repository)에 관한 연구도 이루어지고 있다. 그러나, 기존에 설계된 저장소는 형상관리(Configuration Management)중인 컴포넌트에 대한 관리가 미흡하다. 따라서, 본 논문에서는 형상관리중인 컴포넌트의 버전을 지속적으로 감시하는 업데이트 엔진을 설계하고, 형상관리 컴포넌트들을 어떻게 효율적으로 관리 할 것인가에 대한 방법을 제시 한다.

1. 서론

소프트웨어 개발자들은 한 가지 공통된 목적을 가지고 있다. 그것은 최단기간의 개발로 최대의 효과를 얻을 수 있는 애플리케이션을 만드는 것이다. 이러한 이유로 1980년대 초반부터 OOP(Object-Oriented Programming) 기술에 대한 연구가 진행되었다. 하지만, OOP는 재사용 소프트웨어의 산출을 보장하지 못하며, 상속으로 인해 클래스의 완전한 캡슐화(Encapsulation)를 지원하지 못한다[6]. 따라서, 최근에는 높은 재사용성을 가지는 컴포넌트 개발에 대한 관심이 높아지고 있다[5]. 컴포넌트의 등장으로, 컴포넌트 기반 소프트웨어의 개발(CBD : Component Based Development)이 활발히 진행 중이며, 컴포넌트도 다양하게 개발되고 있다. 컴포넌트 종류와 수가 방대해지면서부터, CBD 과정에서 쉽고, 빠르게 컴포넌트를 검색할 수 있는 저장소의 필요성이 대두되고 있다[4][6][7].

저장소는 사용자의 요구에 빠르고, 정확한 결과를 제공해야 한다[3][7]. 하지만, 컴포넌트 개발 작업은 다른 소프트웨어와 마찬가지로 무형의 산출물을 생산하는 것이므로 개발공정(Development Process), 변화과정 및 유지보수(maintenance)에 대한 통제가 어렵다. 이것은 컴포넌트 정보의 누락을 야기한다. 누락된 컴포넌트 정보는 CBD의 시간과 비용 증가를 가져오고, 애플리케이션 개발의 전체적인 효율성을 떨어뜨린다. 따라서, 형상관리중인 컴포넌트에 대한 저장소 차원의 관리가 필요하다. 또한, 형상관리 정보는 컴포넌트의 등록시에만 갱신되고, 형상관리중에 개발된 컴포넌트는 개발자가 수동으로 등록시켜야 하므로, 체계적이고 단계적인 형상관리 정보를 저장소에 구축할 수 없다. 그러므로, 형상관리 정보수집의 자동화가 필요하다.

따라서, 본 논문에서는 형상관리중인 컴포넌트의 관리를 위한 저장소 업데이트 엔진을 설계한다.

2. 관련연구

본 절에서는 기존에 연구된 컴포넌트 저장소와 형상관리방법에 대해 간단한 소개를 한다.

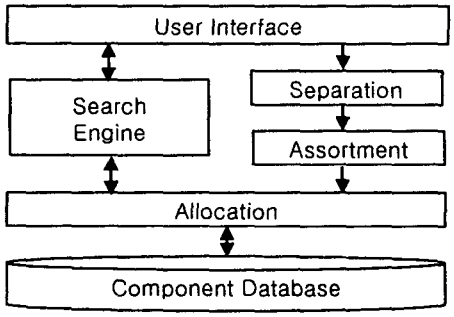
<표 1> 전형적인 컴포넌트 저장소 기능

Function	Description
Security	User Authentication
Management	Registration, Search, Update, Deletion of Component Information.
Configuration Management	Version Management of Component
Upload & Download	Upload, Download of Component
Browsing	Browsing for Component Information

2.1 전형적인 컴포넌트 저장소

전형적인 컴포넌트 저장소의 기능은 보안, 관리, 형상관리, 컴포넌트 다운로드 및 업로드, 컴포넌트 브라우징이다. <표 1>은 전형적인 컴포넌트 저장소의 기능을 나타낸다.

전형적인 저장소의 기능 중에서 컴포넌트 등록과 검색 기능이 가장 비중이 크다. [그림 1]과 <표 2>은 컴포넌트 검색과 등록만을 고려한 저장소 구조와 모듈별 설명을 나타낸 것이다.



[그림 1] 전형적인 컴포넌트 저장소의 구조

<표 2> 저장소 모듈별 기능

Module	Function
Separation	Separation of Specification
Assortment	Component Assortment by Separated Specification
Allocation	Storage by Assortment
Search Engine	Component Searching

2.2 컴포넌트 형상관리

저장소는 형상관리 컴포넌트에 대한 정보를 가지고, 요구하는 사용자에게 형상관리 정보를 제공해야 한다. 전형적인 저장소의 형상관리 서비스는 다운로드 횟수 측정, 버전관리, 컴포넌트 변경사항 관리 등을 제공한다.

다운로드 횟수 측정은 해당 컴포넌트를 사용하는 사용자가 얼마나 되며, 또한 해당 컴포넌트가 지금도 요구되는지에 대한 척도가 된다.

버전관리는 해당 컴포넌트가 몇 단계를 거쳐 개발 되었는지, 혹은 이미 다운로드 받은 사용자가 사용하는 버전과 비교 할 수 있는 자료로 사용된다.

컴포넌트 변경 사항관리는 버전에 따른 컴포넌트의 인터페이스(Interface)에 관한 변경 사항을 관리한다.

관련 연구에서 제시하는 형상관리 방법은, 이미 등록된 컴포넌트를 이용해 형상관리 정보를 구축한다. 하지만, 형상관리중인 컴포넌트의 등록은 수동적이다. 수동적인 등록 방식일 경우, 사용이 중단된 컴포넌트나, 개발과정 중 수정되어야 할 부분이 있는 컴포넌트에 대한 정보갱신이 어렵다. 이것은 잘못된 형상관리 정보가 제공될 가능성이 있다. 그러므로, 형상관리중인 컴포넌트에 대한 능동적인 관리가 필요하다.

따라서, 본 논문에서 제시하는 시스템은 기존에 연구된 저장소에 업데이트 엔진을 탑재함으로써 형상관리중의 컴포넌트를 지속적으로 감시하고, 이를 이용해 보다 나은 형상관리 서비스를 제공한다.

3. 업데이트 엔진 설계

컴포넌트 저장소는 컴포넌트 라이프사이클의 모든 정보들을 컴포넌트 아키텍처에 따라 저장, 등록, 관리 하며 나아가 진보된 검색 서비스 및 정보의 브라우징 기능을 통해 컴포넌트의 재사용을 지원하는 도구다[7]. 따라서 컴포넌트 저장소는 기본적으로 두 가지의 기능을 구비해야 한다.

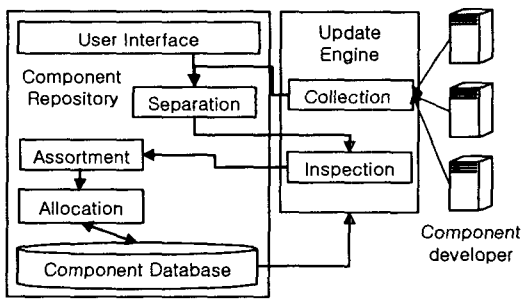
첫째, 컴포넌트 등록이다. 컴포넌트가 등록되고, 등록된 컴포넌트를 어떻게 분류할 것인가에 대한 부분이 필요하고, 자동화 되어있어야 한다.

둘째, 컴포넌트 검색이다. 컴포넌트를 필요로 하는 사람에게 어떻게 쉽고, 빠르고, 정확하게 정보를 제공 하는가에 대한 부분이다. 이를 위해 컴포넌트의 속성별 분류는 필수적이다.

본 논문에서는 위의 두 가지 기능 중 등록기능에 업데이트 엔진을 탑재함으로써 컴포넌트 업데이트 여부의 자동검사가 가능하다. 또한 자동화된 형상관리 정보갱신은 보다 빠르고, 정확한 형상관리 정보를 제공한다.

3.1 업데이트 엔진을 추가한 저장소 시스템 구조

[그림 2]는 전형적인 저장소에 업데이트 엔진을 탑재한 구조를 보여 주고 있다.



[그림 2] 업데이트 엔진을 탑재한 저장소 구조

3.2 저장소 모듈별 역할

이 절에서는 저장소가 가져야할 기본적인 모듈을 제시하고, 업데이트 엔진을 탑재함으로써 추가된 모듈의 기능에 대해 설명한다.

앞 절에서 소개했던 것과 같이, 기본적인 검색 시스템의 구조는 User Interface, Separation, Assortment, Search Engine, Allocation으로 이루어지고, 각 모듈의 역할은 전형적인 저장소의 모듈과 동일하다.

업데이트 엔진을 추가하면서 삽입된, Collection 모듈과 Inspection 모듈은, 각각 형상관리중인 컴포넌트에 대한 수집과 버전 검사를 하는 모듈이다.

4. Component Database

Component Database는 저장되어있는 컴포넌트에 대한 정보를 담고 있다.

버전확인에 필요한 저장소의 데이터를 3가지로 분류하면, 업데이트 판단을 위한 데이터와 업데이트 확인에 필요한 데이터, 그리고 컴포넌트 인터페이스에 대한 데이터가 있다.

첫째, 업데이트 판단을 위한 데이터는, 컴포넌트 이름, 컴포넌트 버전, 작성자에 대한 정보등으로 구성된다.

둘째, 업데이트 확인에는, 해당 데이터의 상태(업데이트 과정중), 탐색위치, 자신보다 최근에 개발된 컴포넌트가 있는지에 대한 데이터가 필요하다.

마지막으로, 컴포넌트 명세서에서 분리 가능한 컴포넌트 인터페이스에 대한 정보데이터가 필요하다.

<표 3>은 업데이트 엔진을 탑재함으로써 추가되는 Database의 레코드 구조를 나타낸 것이다.

<표 3> Component Database의 레코드 구조

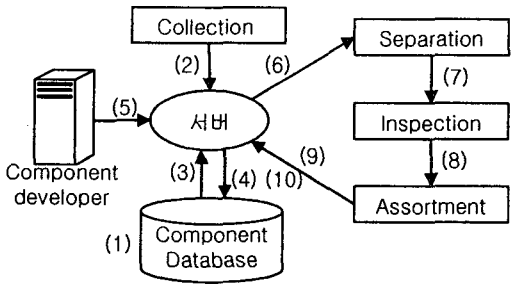
Field Name	설 명	
ID	컴포넌트 고유 ID	
Name	컴포넌트의 Name	
Version	컴포넌트의 Version	
Writer	컴포넌트의 작성자 혹은 회사	
Locate	컴포넌트를 제공하는 서버 위치	
Updated	컴포넌트의 업데이트 여부	
	R	업데이트 대기 상태(Ready)
	A	버전 정보를 확인했음(After)
	C	버전 정보를 비교 중(Confirm)
	N	네트웍 장애나 다른 이유로 확인 불가(Not Confirm or Number)
	E	형상관리가 종료된 컴포넌트(End)
New	컴포넌트의 버전 관리 정보	
	T	해당 컴포넌트 데이터가 최신 버전임
	F	다른 최신의 컴포넌트가 존재
Day	컴포넌트의 등록 날짜	
Down	컴포넌트의 다운로드 횟수	
Sig	명세서에서 추출한 인터페이스 정보	

'R'은 아직 업데이트 검사가 되지 않은 상태로서 업데이트 검사가 되어야 하는 상태가 되고 'A'는 이미 업데이트 검사를 하고 지나간 상태가 되며, 'C'는 현재 업데이트 검사 중인 상태를 나타낸다. 'N'은 카운터의 값을 저장 할 공간으로, 회선의 장애나 컴포넌트 제공 서버로부터 응답이 없을 때, 검사 사이클 당 1회 카운트 한다. 'E'는 형상관리가 끝난 컴포넌트 데이터를 의미하며, 이 데이터는 개발자가 수정한다.

New Field는 해당 데이터 보다 최근에 개발된 컴포넌트가 있는지를 알려주는 값이 된다.

5. Update 과정

이 절에서는 컴포넌트의 버전 검사방법에 대한 과정을 서술한다. [그림 3]에 컴포넌트 업데이트 엔진의 구조와 역할이 표시되어 있다.



[그림 3] 컴포넌트 업데이트 엔진의 구조

(1) 컴포넌트 검색을 위해서는 먼저 Database 의 초기화가 필요하다. Updated Field : 'A'를 'R'로 바꾸어 재검사 할 수 있게 한다.

(2) 버전을 조사해야 할 컴포넌트의 데이터를 요구. (Database의 데이터를 순차적으로 요구)

(3) Inquiry가 요구하는 데이터를 전송. 서버에서는 받은 컴포넌트 데이터가 최신 버전의 컴포넌트 인지 검사한다. (최신버전은 New Field가 'T') 만약, 저장소에 등록된 컴포넌트가 최신의 버전이 아니라면, 버전 검사에서 컴포넌트가 새로 개발되었다고 할 수 있으므로, 항상 최신 버전의 데이터를 가지고 비교해야 한다.

(4) (3)의 과정에서 최신 버전의 데이터임이 판별 되면 Updated Field에 'C'를 기록한다. 이 것은 버전 검사 중에 등록자가 해당 컴포넌트의 데이터를 수정, 삭제 하지 못하게 한다.

(5) Inquiry가 요구한 데이터를 가지고 검색된 컴포넌트를 서버로 가져온다. 서버로 가져오는 방법은 Locate에 있는 컴포넌트 제공 서버의 주소를 탐색해서 얻을 수 있다. 만약, 해당 서버의 주소에서 지정된 컴포넌트를 찾을 수 없다면 Updated Field 에 'N'을 기록한다. 제공되지 않는 컴포넌트에 대한 관리의 경우에는 카운트한다.

(6) 컴포넌트의 명세를 분리 하여 패싯을 만든다.

(7) 분리한 패싯을 Inspection 모듈로 넘긴다.
버전 검사는 컴포넌트 이름, 컴포넌트 버전, 컴포넌트 작성자로 검사 할 수 있다.
컴포넌트 이름과 작성자, 버전 정보가 같다면 같은 컴포넌트로 규정하고, 버전 정보만 다른 경우라면 버전의 상호 비교 후 새로 개발된 된 컴포넌트로 처리한다.
컴포넌트 이름이나 작성자의 정보가 다른 경우, 신뢰 할 수 없는 컴포넌트로 간주한다.

(8) 새로운 컴포넌트 정보로 분류한다.

(9) 분류된 정보를 서버에 알리고 Database에 저장할 것을 요구한다.

(10) Assortment 모듈로부터 받은 분류 정보를 이용해 컴포넌트 정보를 기록한다. 기존 데이터의 New Field에 'F'를 기록한다. 이것은 해당 데이터 보다 최근에 개발된 컴포넌트가 등록 되어있음을 나타낸다.

(11) Database의 마지막 데이터가 될 때 까지 (2)~(10)를 반복 한다.
(12) Database의 마지막 데이터를 검사한 후, (1)을 다시 반복하여 컴포넌트를 검사한다.

6. 형상관리 서비스

본 시스템에서 제공하는 형상관리에 대한 세부적 결정은 저장소의 서비스 형태에 맞게 수정이 가능하고, 제공하는 형상관리에 대한 서비스는 두 가지이다.

첫째, 일정 기간 이전 버전의 컴포넌트에 대한 관리를 제공한다.

급속한 하드웨어 발전으로 저장 공간도 같이 방대 해졌다. 하지만, 최신 버전의 컴포넌트가 개발 되었고, 일정 단계 이전 버전의 컴포넌트가 더 이상 쓰이지 않게 된다면, 해당 컴포넌트는 사용자에게 제공할 필요가 없다.

컴포넌트의 쓰임새는 등록된 날짜와 다운로드 횟수의 임계치에 따라 결정 할 수 있다. 컴포넌트 등록일이 일정 기간 보다 오래 되었고, 해당 컴포넌트가 등록된 이후 몇 단계의 컴포넌트 등록이 있었고, 다운로드의 횟수에 임계치를 주어 쓰이지 않는 컴포넌트임이 확인되면, 해당 컴포넌트는 삭제 가능한 컴포넌트로 규정한다.

둘째, 개발 되었다가 중단된 컴포넌트에 대한 관리를 제공한다.

개발자가 형상관리 컴포넌트를 개발 중에 컴포넌트 제공 서버를 장기간 찾을 수 없거나, 탐색위치에서 해당 컴포넌트를 장기간 찾을 수 없게 된다면, 더 이상 신뢰할 수 없는 컴포넌트로 규정한다.

해당 컴포넌트를 검사하기위해 서버로 가져오는 시점에서 컴포넌트가 제공 위치에서 탐색되지 않는다면, Updated Field에 카운트되고, 이 카운트의 임계치에 따라 저장소에서 삭제 가능하다.

7. 결론 및 향후 연구 방향

본 논문에서는 전형적인 컴포넌를 저장소에 업데이트 엔진을 탑재하고, 저장소가 스스로 형상관리 중인 컴포넌트를 감시 할 수 있도록 하였다. 컴포넌트 저장소에 업데이트 엔진을 탑재하기 위해서는 엔진의 모듈과 업데이트에 필요한 최소한의 Component Database 구조를 설계했다. 업데이트에 필요한 모듈을 설계 하였고 업데이트 상세 프로세스를 설계했다.

저장소에 업데이트 엔진을 탑재함으로써, 개발자가 개발된 컴포넌트를 직접 저장소에 등록하지 않아도 저장소가 스스로 컴포넌트를 탐색하여 컴포넌트에 대한 신속한 자료 갱신이 이루어진다. 또한, 이 업데이트 엔진은 기존에 연구된 서로 다른 컴포넌트 저장소에 간단한 수정만으로도 탑재할 수 있고, 버전이 낙후되어 쓰이지 않거나, 형상관리 중에 개발이 중단 된 컴포넌트의 자동적 관리를 제공한다. 따라서, 형상관리중 불안한 컴포넌트에 대해서도 저장소 차원의 버전관리로 안전하게 컴포넌트를 사용자에게 제공한다.

향후 연구과제로 이 논문에서 설계한 업데이트 엔진을 실제 구현 해보고 여러 컴포넌트 저장소에 탑재 해보는 것과 데이터의 양에 따른 서버 트래픽의 조사가 이루어져야 한다.

참고문헌

[1] N.H. Lassing, D.B.Brjiewnbrij, and J.C van Vliet, "A View on Component" Proc. Of 9th Int. Work-shop on Database and Expert Sys. Applications, pp.769-77, 1998.
[2] P.Freeman, "Reusable Software Engineering : Concepts and Research Directions," Proc. Workshop Reusability in Programming, pp.2-16, 1983.
[3] J. Guo and Luqi, "A survey of Software Re-use Repositories," Proceedings of the 7th IEEE Int. Conf, & W/S on the Engineering of Computer Based System, pp88-96, 2000.
[4] C. McClure, The Three Rs of Software Auto-mation : Reengineering, Repository, Reusability, Prentice-Hall, 1993
[5] 이 윤, 김태공 "소프트웨어 재사용을 위한 분산 컴포넌트에 대한 연구" Natural Sci. INJE Univ.
[6] 서성권 "컴포넌트 검색엔진을 탑재한 소프트웨어 컴포넌트 저장소" 고려대학교대학원 전산학과 학위논문집. 2000.
[7] 금영욱, 박병섭 "확장된 소프트웨어 컴포넌트 서술자에 기초한 컴포넌트 저장소의 검색", 정보처리 학회논문지 D 제9-D권 제3호(2002, 6).