패치분배 시스템을 위한 효과적인 계층적 소프트웨어 아키텍처 표현 방안

이수영 이인용 조재익 문종섭
정보경영공학전문대학원
e-mail : leesuyoung@korea.ac.kr

An Efficient Hierarchical Software Architecture Expression Method for Patch Distribution System

Suyoung Lee, Inyong Lee ,Jaeik Cho, Jongsub Moon
Graduate School of Information Management & Security

요 약

소프트웨어 아키텍처는 설계자가 요구하는 사항을 개발자에게 전달하기 위해 사용된다. 많은 아키텍처 방안들은 코드 의존적으로 만들어 졌다. 이는 설계자가 요구하는 사항을 개발자에게 충분히 전달할 수 없다. 또한 현재 많은 네트워크 시스템들은 웹 환경을 포함하고 있기 때문에 본 논문에서는 웹 환경을 포함하고 설계자와 개발자간에 명확한 의사소통을 위해 사용될 수 있는 아키텍처 구성 방안을 제시 한다. 그리고 현재 많은 소프트웨어 취약점으로 인해 보안적인 문제를 가지고 있다. 따라서 불가피하게 패치를 개발하고 분배하는 방법을 필요로 한다. 현재 패치 분배 시스템의 개발은 타기종간의 분산 환경 및 패치 환경 그리고 웹의 개발에 의해서 복잡해 진다. 그래서 제안한 방안을 이용해 패치 분배 시스템을 구성 하여 제시한 방안에 의해 효율적으로 수행해 볼 것이다.

1. 서론

패치 분배 시스템에 대한 규모가 커지고 여러 범용적인 운영환경을 목적으로 패치 분배 시스템이 개발되고 있다. 패치 분배 시스템처럼 복잡하고 큰 시스템을 개발할 경우 시스템은 복잡도, 확장성, 연결성 등 고려사항이 증가한다. 현재 사용자 편의를 위해서 대부분의 패치 분배 시스템이 웹 환경을 이용하여 구축 되고 있다[1]. 웹 어플리케이션을 포함하는 패치 분배 시스템은 더욱 복잡하여 시스템 구조에 컴포넌트를 적절히 배치하고 표현하기가 어렵다. 따라서, 설계자, 관리자, 컴포넌트 생산자 및 소비자들의 이해와 시스템의 유지 보수를 위해 컴포넌트를 이용한 아키텍처를 설계한다 [2]. 아키텍처는 시스템 개발에 관한 대략적인 이해를 위해 만들어지며 컴포넌트는 설계자가 개발자에게 요구사항을 표현하기 위해 만들어진다 [2]. 구조화된 설계 양식을 이용해 시스템을 구성 및 개발 할 경우 컴포넌트의 관계가 명확해 져 이해 관계자들의 혼선을 줄일 수 있을 뿐만 아니라 차후에 유지 보수가 용이해지는 장점이 있다.

본 논문에서는 현재 비즈니스 환경에서 아키텍처 설계를 위해 사용하는 계층적 구조에 추상화와 계층적 구조 특징을 이용하여 패치 분배 시스템을 위해 6 계층으로 세분화 했다. 6 계층으로 나누는 아키텍처 개념을 이용함으로써 시스템 설계 시, 다음과 같은 장점을 가지게 된다. 첫째, 패치 분배 시스템의 개발 분야와 범위를 정확히 파악 하므로써 아키텍터가 의도하는 아키텍처의 일관성을 유지 할 수 있다. 둘째, 컴포넌트 생산자와 소비자가 명확 해지므로 컴포넌트 간의 의존성이 뚜렷해져서 가변적인 상황에 대하여 더욱 유연하게 시스템을 변경 할 수 있다. 셋째, 아키텍터는 계층 구성에 맞게 컴포넌트를 선정 하므로써 패치 분배 시스템을 일목연하게 구성할 수 있게 된다.

본 논문의 순서는 다음과 같다. 2 장에서는 과거에 연구되었던 아키텍처 종류 및 분류에 대해 설명하며, 3 장에서는 본 논문에서 6 계층 구성 방안을 통한 아키텍쳐 설계 방법을 제시한다. 4 장에서는 설계 단계의 시스템에 제안하는 방법을 적용한 웹 네트워크 패치 분배 시스템을 설계하고, 같은 시스템에 대해 2 장에서 설명되었던 과거의 연구 기법과 비교한다. 마지막으로 5 장에서 실험을 통한 비교 결과에 대해 결론을 맺는다.

2. 아키텍처 구성 연구

Perry, Wolf 의 논문에서는 요소, 폼, 이론적 근거를 바탕으로 아키텍처를 구분했다[3]. Soni, Nord, Hofmeister 가 제안한 방법에서는 4 가지의 뷰모델을 이용하여 아키텍처를 설명했다 [6]. 개념 아키텍처 (concept architecture)는 주요한 디자인 요소와 이들 관계의 사

* 본 연구는 정보통신부 및 정보통신연구진흥원의 대학 IT 연구센터 지원사업의 연구결과로 수행되었음. (IITA-2006-(C1090-0603-0025))

이를 설명한다. 모듈 상호 아키텍처 (module interconnection architecture) 에서는 기능적 분해와 레이어의 두 수직적 구조를 나타낸다. 실행 아키텍처 (execution architecture)는 시스템의 동적인 구조를 설명한다. 코드 아키텍처(code architecture)는 환경을 구성하는 소스코드, 라이브러리, 바이너리 가 어떻게 구성하는지 기술 한다. 이 중에서 모듈 아키텍처가 레이어로 구성 되어 있지만 모듈에 한정적이라는 단점이 있다. Tanuan 이 쓴 비즈니스 소프트웨어 도메인에서 소프트웨어 아키텍처에서는 시스템 전체적인 이해를 돕기 위해 논리 뷰를 찾고 개념적으로 컴포넌트를 나누었다[7]. 이와 같이 여러 아키텍처가 존재하고 이에 따라 이해관계자들의 이해를 돕기 위해 많은 뷰들이 존재 한다. 많은 뷰들이 각 아키텍처에 대응해서 만들어 진다. 이런 아키텍처와 뷰들은 어디에 어떻게 매치 되는지 알기 쉽지가 않다. 또한 좋은 표현 뷰일지라도 다른 개념으로 사용하면 설계자와 개발자간의 상반되는 지식으로 오해를 일으킨다. 각 아키텍처와 뷰들이 따로 정의되어 있을지라도 이런 개념들이 위치하는 곳을 적절히 배치 시켜준다면 오해의 소지는 많이 줄어든다.

3. 아키텍처 구성 방법

현재 웹에 의한 소프트웨어 개발이 현저하게 증가함으로써 시스템에 대한 웹의 비중 또한 커 졌다. 이전에는 단일 환경을 겨냥한 시스템과 아키텍터가 수용할 수 있을 정도의 개념을 가지고 시스템을 구성하였다. 그러나 웹 환경을 고려한 개발은 다중 환경과 복잡한 기술적 요소가 포함 된다. 그뿐만 아니라 웹이라는 특수한 환경적 요인도 개발 범위에 포함해야 한다. 그래서 다음과 같은 6 단계로 세분화 된 소프트웨어 아키텍처를 제안 한다. 각 계층은 여러 요소를 수용할 수 있는 컨테이너로 각 개념을 분리 한다. UML 로 정의 될 수 있는 모델, 다이어그램, 뷰가 들어 간다.

<표 1> 계층 분류표

Presentation	UI 화면, 보고서 형태, 음성 인터페이스, HTML, XML, ASP.NET, JSP, javascript, … 에서 생성 되는 요소들을 포함한다. 사용자 행동과 직접 관련이 있으며, 시스템과 사용자간에 의사 소통 가능한 컴포넌트들을 표현한다.
Session	상위 계층의 요구 사항을 처리하는 로직과 워크플로우, 세션 상태 처리, 스테이트 다이어그램, 시스템 스퀀시 다이어그램 같은 동적인 개념들을 포함 한다.
Domain(s)	소프트웨어 시스템의 논리적인 처리 로직 뿐만 아니라 프로그램 전체의 상태를 확인하는 로직도 포함 되어 있으며 이는 소프트웨어 시스템에 중추적인 계층이라고 볼 수 있다.
Software Infrastructure(SI)	Domain 계층에서 정의 된 로직을 수행 하기 위해서 낮은 수준의 클래스들 집합으로 반복 사용 가능한 요소들이 존재 한다.
Technical Support(TS)	SI 및 Foundation 계층에 기술적인 기반이 되는 문서, 보안 문서, 영속성 문서가 배치된다. 이는 이후에 소프트웨어 유지보수에 유용하게 사용 가능한 계층이다.
Foundation	소프트웨어의 하부 구조로서 데이터 구조 정의, 파일 구조 정의, DB 관리, 네트워크 구조 정의, 부분을 가지고 있으며 Technical Support (TS)에서 정의된 기술을 실현 하는 부분이며 이는 Domain 과 SI 구조에서 필요한 기본 요소를 지원 한다.

많은 아키텍처 개념들이 존재 하고 있다. 하지만 정작 소프트웨어에 대한 청사진을 그릴 때 어떤 아키텍처가 필요한지, 정확히 부합하는지 확인하기는 매우 어렵다. 이 논문에서 제시한 웹 응용 시스템에서의 계층적 구조는 소프트웨어 개발 시에 산출물로 나올 수 있는 모든 결과물들을 적절한 계층에 위치 시키므로써 웹 환경에서의 소프트웨어 개발이 더욱 견고 해질 수 있다. Presentation 계층에서는 사용자와 상호 의사 소통 할 수 있는 모든 객체 및 아키텍처가 들어 간다. 이 계층에서는 사용자가 느끼는 관점에 초점을 맞춘다. 표현 하는 기능 외에 논리적인 흐름과 로직은 Domain 계층으로 넘어 가므로써 복잡도를 최대한 줄일 수 있다. 다음으로 Session 계층은 사용자 및 객체의 상태를 나타내는데 주로 표현된다. 이 계층은 통상 동적인 개념들을 포함 한다. 이 논문에서 제시한 Domain 계층은 컴포넌트를 구분해서 담는 역할 중 가장 중요한 처리 로직, 업무 로직, 기타 소프트웨어의 중추적인 로직이 들어 간다. 또한 하부 계층과 상위 계층을 연결 해주는 중요한 다리 역할을 포함 하고 있다. Software Infrastructure(SI) 계층은 Technical Support(TS) 계층의 논리적인 부분과 Foundation 의 기본 단위인 요소 들를 실현 시켜 주는 부분이 된다.

소프트웨어 개발 시에 개념을 세분화 함으로서 여러 이점을 얻을 수 있다. 첫 번째로는 복잡한 웹 네트워크 구조를 계층 요소에 의존 관계로 적절히

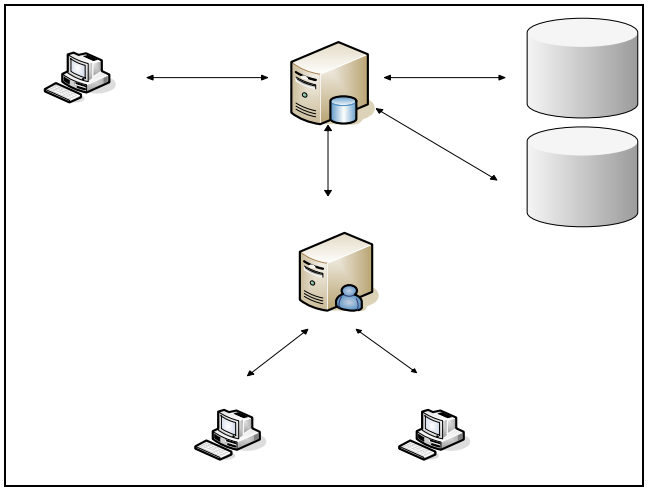
표현 하므로써 시스템 전체적인 흐름을 이해 할 수 있다. 둘째, 계층으로 나누어 생각 하므로 각 계층의 역할에 적합한 모델을 제시하고 사용 한다면 계층간의 추상화를 이룰 수 있기 때문에 독립적으로 운영 가능 해진다.

이 장에서 제시한 6 단계의 구조는 현재 웹을 포함하는 네트워크 시스템 구조를 표현하기 적합하게 구성 하였다. 이 구조는 소프트웨어 개발 시 이해관계자들 에게 시스템 요소들 사이의 관계를 계층 배치와 의존 관계 선을 이용해 적절히 표현 하므로 혼선을 줄 일 수 있다. 또한 점점 시스템 구조가 복잡해지고 관련된 기술이 많아 지므로 이 또한 아키텍처에 요소의 하나로 배치 하므로 기술이 변화 함에 따라 유연 하게 대체 할 수 있는 구조가 될 수 있다.

4. 패치 시스템 아키텍처 구성

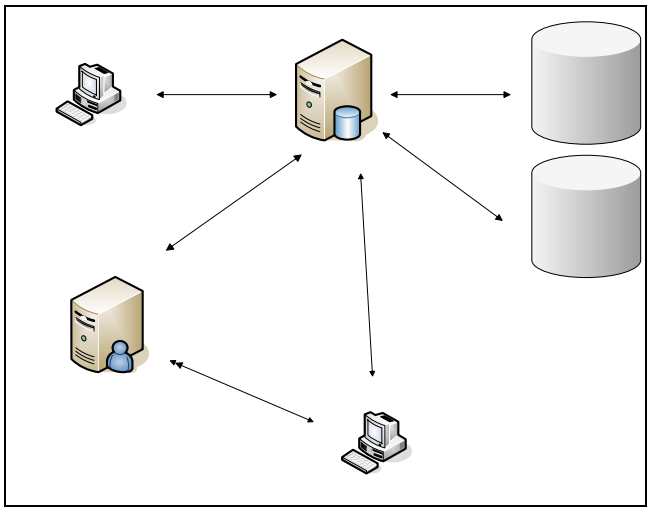
본 논문에서 제시한 컨테이너를 이용한 계층적 아키텍처 구성 방안을 패치 분배 시스템에 적용해본다. 패치는 프로그램 버그가 발견 되거나 취약점이 존재할 경우 수정사항을 위해 만들어진다. 하지만 이런 취약점을 보안 하기 위해 적용하는 패치가 도리어 취약점을 공격자에게 알려주는 단점을 가지고 있다. 따라서 이런 정보를 노출 하지 않고 패치를 수행하여 취약점을 없애는 것이 보안 패치의 주요 목적이다.

모든 클라이언트는 독립된 망에서 서버의 명령만으로 통신을 연결 하는 단방향 연결 체제 이다. 이런 연결은 사설 망에서 유용할 수 있다. 패치 분배에서 생길 수 있는 보안적 문제점을 해결 하기 위해 클라이언트에 서버가 연결 하는 방식을 유지한다. 처음으로는 서버와 클라이언트간에 인증을 확인 한다.



(그림 1) 파일분배

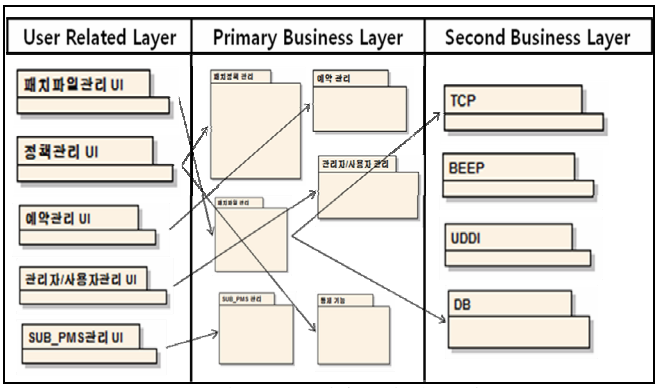
다음은 서버와 파일 서버 인증 후 암호화된 패치 파일 분배가 이루어 진다. 파일 서버와 클라이언트는 인증 서버로부터 받은 대칭키를 가지고 파일 전송을 하게 된다. 서버에는 인증과 통제를 담당 하는 기능과 패치 파일 분배에 대한 정책 부분 그리고 부과 적인 기능을 가지고 있다. 패치 파일 분배 서버는 대용량 패치 파일 분배를 위해 로드 밸런싱으로 묶여 있다. 모든 클라이언트는 인증된 패치 파일 분배 서버로부터 패치 파일을 받게 된다.



(그림 2) 인증

구성된 시스템 아키텍처는 인증 부분의 구조와 파일 분배시의 구조가 다르게 나타난다. 인증 시에는 패치 파일 분배 서버 및 클라이언트 각각을 인증하게 구성 하였다. 이렇게 구성 하므로써 패치 파일 분배 시 어떤 패치를 분배 하는지 은닉 하므로 패치로부터 취약점을 얻어 내는 방법을 차단 할 수 있다. 이렇게 구성된 시스템 아키텍처를 바탕으로 이 전 장에서 제안한 계층적 구조의 소프트웨어 아키텍처를 설계 했다.

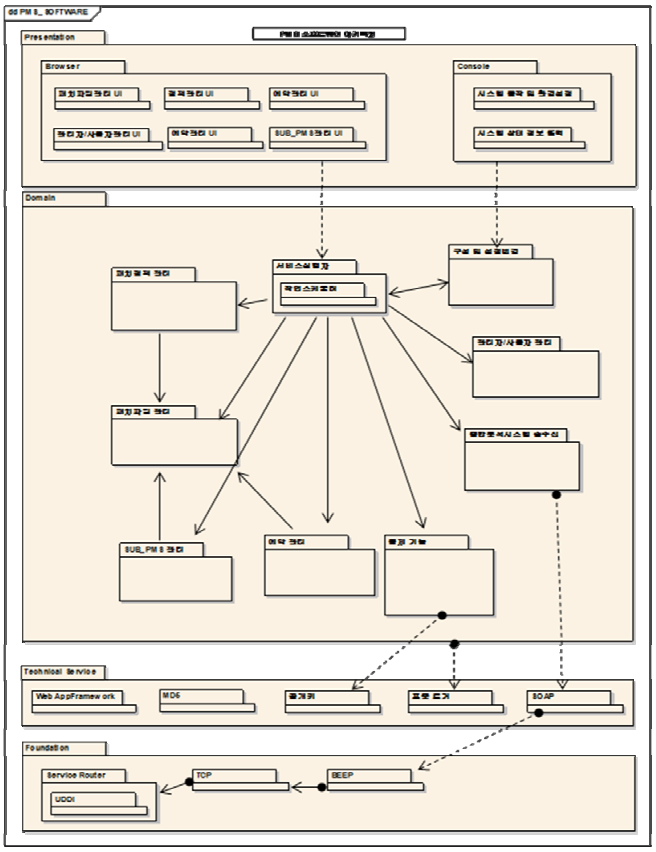
그럼 이렇게 구성한 시스템을 앞장에서 제시한 방안을 이용해 아키텍처를 구성 해본다. 현재 많이 사용하는 구조적인 아키텍처 구성은 다음과 같다.



(그림 3) 3 계층 아키텍처

그림 3 은 패치 분배 시스템을 3 계층 아키텍처로 분류 한 것이다. 처음 계층은 사용자와 관련된 컴포

넌트를 배치하며 중간에는 비즈니스 핵심 기능에 대한 컴포넌트가 마지막으로는 비즈니스 계층을 지원하는 컴포넌트가 들어 간다. 이 아키텍처는 기술적인 컴포넌트에 대한 표현은 안 된다.



(그림 4) 제안한 방법으로 설계한 아키텍처

그림 4 은 본 논문에서 제안한 계층 뷰를 나타내며, 계층 상하는 제안한 방법대로 연관성을 가지고 있다. 세분화 하여 계층에 담으므로써 계층적 배치 아키텍처 구조에서 의존관계가 명확해짐을 알 수 있다. 기술 문서와 연결된 선은 꼬리에 원으로 구분하여 사용하는 기술을 명시 하고 있으며 점선은 계층간을 넘어서 의존성을 표시할 때 나타내고 실선은 같은 계층간에 의존관계를 나타낸다. 따라서, 선들은 각 객체가 어디에 의존적인지 명시적으로 알 수 있다. 그러므로 이런 설계단계의 청사진 이 있다면 유지 보수에 유용할 것이다. 또한 개발시의 설계자와 개발자들 또한 명확하게 뜻을 이해 할 수 있기 때문에 혼선을 줄일 수 있다. 그리고 유지보수 문제에 있어서도 각 기능과 논리적인 위치를 살펴 확인 함으로서 유연하게 대처 가능 한다.

5. 결 론

현재 여러 아키텍처를 표현하는 방법이 UML 기반에서 언어화 되었다. 그렇지만 이전의 아키텍처가 의미하는 표현 방안은 설계자와 개발자간에 명확한 의사소통을 하기가 어렵다. 특히 의존성 관계와 적절한 배치에 대한 혼선을 초래 한다.

본 논문에서는 이런 단점을 보완할 아키텍처 컨테이너 개념을 도입해 6 계층으로 세분화 하여 적용해 보았다. 본 논문에서 제시한 방안은 숙련되지 않은 아키텍터 일지라도 일정 수준의 일반화를 거친 배치를 확인하므로 프로그램 개발에 있어서 설계자와 개발자간 의사 소통의 혼선을 줄일 수 있다. 이는 차후에 있을 유지 보수에 큰 도움이 될 수 있다.

이후에는 의존관계에서 생길 수 있는 각 컴포넌트들의 연결 관계 표현 문제점 및 계층 연결 부분에 생길 수 비가시성에 대한 문제를 해결해야 할 것이다.

참고문헌

[1] San Murugesan, Yogesh Deshpande, “Meeting the challenges of web application development”, International Conference on Software Engineering, pp.687~688, 2002

[2] Seung C. Lee, Ashraf I. Shirani, “A component based methodology for web application development”, Journal of Systems and Software, pp.177~187, 2004

[3] Dewayne E. Perry, Alexander L. Wolf, “Foundations for the Study of Software Architecture”, ACM SIGSOFT Software Engineering Notes, pp. 40~52, 1992

[4] Dilips Soni, Robert L. Nord, Christine Hofmeister, “Software Architecture in Industrial Applications”, International Conference on Software Engineering, pp. 196 ~207, 1995

[5] Meyer Tanuan, “Software Architecture in the Business Software Domain: The Descartes Experience, Foundations of Software Engineering”, pp. 145~ 148, 1998

[6] Edward A. Schneider, “Security Architecture-Based System Design”, ACM Press, pp. 25 ~ 31, 1999

[7] Dewayne E. Perry, Alexander L. Wolf, “Foundations for the Study of Software Architecture”, ACM SIGSOFT Software Engineering Notes, Volume 17, Issue 4, pp. 40 ~52 , 1992

[8] 박원영, 박수용, “자율성 및 상호작용성을 위한 에이전트 아키텍처 설계, 정보과학회논문지”, 제 30 권, 제 9·10 호, pp. 955~972, 2003

[9] 백희숙, 전재우, 오삼권, “3-Tier 구조를 갖는 웹 데이터베이스 관리 시스템의 설계 및 구현”, 제 26 권 제 2 호(Ⅰ), pp. 87~89, 1999

[10] 임성빈, 송치양, 문창주, 백두권, “3 계층 표현 방식 아키텍처에서 UML 기반 컴포넌트를 이용한 시스템 모델링 기법”, 제 27 권, 제 2 호(Ⅰ), pp. 448~450, 2000

[11] 이혜선, 이은배, 고현희, 박재년, “분산환경에서의 비즈니스 정보 시스템 아키텍처 분류”, 제 31 권, 제 1 호(B), pp. 448~450, 2004

[12] Greg Hoglund, Gray McGraw, “EXPLOITING SOFTWARE: How to break code”, Addision Wesley Professional, pp. 44~53, 2004

[13] 김윤주 이상원 손태식 문종섭 서정택 윤주범 박응기, “확장성을 고려한 계층적 패치 분배 시스템 프레임 워크 설계”, 한국정보과학회 학술발표논문집, 제 31 권, 제 1 호(A), pp. 199~201, 2004