특집

엔터프라이즈아키텍팅을 통한 시스템 컴포넌트 품질향상 방법

임 남 규, 이 태 공(아주대학교)

요 약

지금까지 개발된 대다수의 정보시스템은 시스템 품질 요소인 상호운용성, 이식성, 확장성 및 재사용성 등이 극히 미흡하여 통합성이 부족할 뿐 아니라 과도한 유지보수비를 요구하고 있다. 더구나 이러한 시스템의 컴포넌트들은 엔터프라이즈 범위에서 개발되지 않아서 해당 시스템 외에서는 재사용을 할 수 없다. 정보기술아키텍처 및 엔터프라이즈아키텍처는 이러한 문제를 해결하기 위하여 출현하였다.

본 논문은 엔터프라이즈아키텍팅 개념을 기반으로 시스템 컴포넌트의 품질을 향상 할 수 있는 방법을 제시하는 것으로, 먼저 아키텍팅 개념을 발전시킨 엔터프라이즈아키텍팅에 대하여 살펴보았다. 둘째, 시스템 컴포넌트의 품질 요소를 정의하기 위하여 시스템기반 상호운용성, 이식성, 확장성, 및 TCO개념을 시스템 컴포넌트개념으로 재정의 하였으며, 셋째, 이를 기반으로 시스템 컴포넌트의 품질 요소인 재사용성을 정의하였고, 마지막으로, 엔터프라이즈아키텍팅에 의해 시스템 컴포넌트의 재사용성이 향상됨을 보여주었다.

I. 서 론

소프트웨어 엔지니어링 기반 정보시스템 개발 방법론에 의해 개발된 대다수의 정보시스템은 넓은 범위인 엔터프라이즈(Enterprise)개념이 반영되지 않은 좁은 범위의 단위 기능을 자동화한 연통형시스템(Stove-Pipe System)이다. 연통형 시스템의 특징은 시스템의 정렬성, 통합성, 상호운용성, 이식성, 확장성, 재사용성 등이 미흡하여 시스템 간 통합이 어려울 뿐 아니라, 시스템의 유지보수비[1] 또한 과다하게 요구되어 새로운 시스템의 개발을 제한하고 있는 실정이다.

더구나 이러한 시스템의 구성요소인 컴포넌트들은 가장 넓은 범위인 엔터프라이즈 범위에서 재사용할 수 있음에도 불구하고, 좁은 범위인 시스템 범위에서 개발되어 재사용이 거의 불가능하다. 우리는 이러한 문제점을 해결하기 위해 구조적 분석 및 설계방법, 객체지향 분석 및 설계방법, 소프트웨어 아키텍처, 컴포넌트기반개발(CBD) 등 많은 노력을 하였으나, 아직까지 기대한 수준만큼의 성과를 달성하지 못한 실정이다[2].

최근 아키텍팅 개념을 도입한 정보기술아키텍처 및 엔터프라이즈아키텍처는 이러한 문제점을 해결하기 위한 또 다른 노력이며, 이미 선진국에서는 이러한 노력에 의한 성과사례가 많이 발표되고 있다[3].

본 논문은 이러한 문제점을 해결하기 위하여, 첫째, 시스템 품질(지표)인 정렬성, 통합성, 상호운용성, 이식성, 확장성, 재사용성, 공유, 유연성 및 TCO(Total Cost of Ownership)에 대하여 살펴보고, 이를 기반으로 시스템 컴포넌트 품질과 관련된 요소를 재정의 한 후, 둘째, 이를 토대로 컴포넌트 품질 표시 방법을 제시하였으며, 셋째, 엔터프라이즈아키텍팅 개념 및 원칙에 의해 시스템 컴포넌트 품질을 향상 시킬 수 있는 방법을 제시한다.

논문의 구성은 II 장에서 관련연구로 시스템 품질 지표에 관련하여 정렬성, 통합성, 상호운용성, 이식성, 확장성, 재사용성 , 공유, 유연성 및 TCO 정의에 대하여 살펴보았고, 더불어 엔터프라이즈 및 아키텍팅 개념을 소개하였으며, III 장은 첫째, 시스템 품질 지표에 관련된 요소를 기반으로 컴포넌트의 품질(지표)을 재 정의하였으며, 둘째, 이를 토대로, 컴포넌트의 품질을 정의하였고, 셋째, 엔터프라이즈 아키텍팅의 개념 및 원칙에 의해 시스템 컴포넌트 품질을 향상 시키는 방법을 제시하였다.

II. 관련연구

시스템 품질은 시스템품질 지표에 의해서 결정 될 것이다. 시스템 품질에 관해 Delone & Mclean은 품질 지표를 편리성, 유용성, 통합성, 대응성, 신뢰성, 유용성, 활용성, 접근성을 제시 하였고, Zachman은 시스템 엔지니어링의 목표를 정렬성, 통합성, 상호운용성, 이식성, 공유, 유연성, 확장성, 재사용성으로, DoDAF에서는 기존시스템은 정렬성, 통합성, 상호운용성, 공유, 유연성, 확장성 및 재사용성이 없을 뿐 아니라 유지보수비가 과다하다는 것을 문제점으로 제시하였다.

본 논문은 이것을 기반으로 엔터프라이즈아키텍팅에 의해 시스템 품질을 가장 높일 수 있는 지표로 통합성, 정렬성, 상호운용성, 이식성, 확장성, 공유, 유연성, 재사용성 및 TCO를 선정했다.

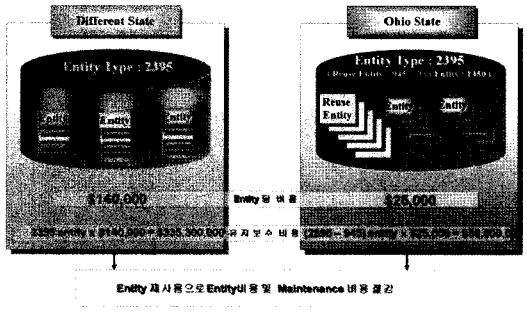
1. 시스템 품질 요소 정의

통합성(Integration)의 정의는 "개별적 체계요소, 구성품, 모듈, 프로세스, 데이터베이스 또는 기타의 개체와 같은 둘 또는 그 이상의 유사한 제품을 하나로 모아지는 정도"이고, 정렬성(Alignment)은 "어떤 연관된 일들 상호간의 배치에 관한 것"으로, 상호운용성(Interoperability)의 정의는 "두 개 이상의 시스템 또는 컴포넌트가 정보를 교환하고 교환된 정보를 사용하는 능력을 의미하는 것"으로, 이식성(Portability)은 "시스템 또는 컴포넌트가 하나의 하드웨어나 소프트웨어 환경에서 다른 하드웨어 및 소프트웨어 환경으로 쉽게 전환될 수 있는 것"이다[4]. 확장성(Scalability)은 "하나의 어플리케이션에서 다양한 기능의 서비스가 요구될 때 다른 크기의 동일한 플랫폼에서 운영이 가능한 것"으로 정의 된다. 공유(Share)란 컴포넌트를 여러 사용자, 프로세스가 골고루 이용하는 것을 의미하고, 유연성(Flexibility)은 "시스템이나 컴포넌트가 본래의 디자인 목적이외의 다른 환

경이나 애플리케이션에서 쉽게 수정하여 사용이 가능한 정도"로 정의 된다. 재사용성(Reusability)의 정의는 "소프트웨어 모듈이나 다른 작업 산출물이 하나 이상의 컴퓨팅 프로그램이나 소프트웨어 시스템에 사용되는 정도"이다[5].

TCO(Total Cost of Ownership)란 "시스템 소유자가 시스템을 소유하기 위해 시스템 생명주기 동안 지불해야 할 비용으로 개발비, 설치비, 유지보수비, 교육훈련비 및 도태비를 포함한 총 소유비용"으로 정의하였다[6].

위 정의를 토대로 재사용 시와 재개발 시 가장 차이가 있는 TCO의 비용요소를 분석하면 개발비, 유지보수비 및 교육훈련비이다. 이것은 재사용 시는 개발된 컴포넌트를 재사용하기 때문에 {개발비+유지보수비+교육훈련비}는{(개발비+유지보수비+교육훈련비)/재사용정도}로 나타내고, 재개발 시는 필요할 때마다 재개발해야 하기 때문에 {개발비+유지보수비+교육훈련비}는 {개발비+유지보수비+교육훈련비}*재개발횟수(재사용정도)}된다. 그러나 설치비 및 도태비는 재사용 시와 재개발 시에 동일하게 (설치비+도태비)*재사용정도(재개발횟수)가 된다. 더구나 설치비 및 도태비는 개발비, 유지보수비 및 교육훈련비에 비해 아주 미미하기 때문에 재사용 시 컴포넌트의 TCO는 급격히 감소할 것이다. 그러므로 재사용 시 TCO를 계산하는 식은 "{(개발비+유지보수비+교육훈련비)/(재사용정도)}+{(도태비+설치비)*(재사용정도)}"로, 재개발 시 TCO 계산식은 {(개발비+유지보수비+교육훈련비+설치비+도태비)*재개발횟수(재사용정도)}로 표현된다.

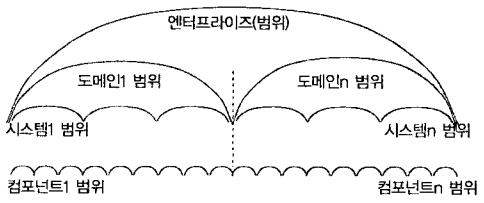


〈그림 1〉 Ohio주 재사용 사례

<그림 1>은 미 Ohio주의 엔티티 재사용에 의해 유지보수비를 절약한 사례이다. Ohio주는 총 2395개의 엔티티 중 945개를 재사용하여 엔티티당 비용을 140,000$에서 25,000$로 줄였을 뿐 아니라 유지보수비도 또한 336,300,000에서 총 36,000,000$로 감소시켰다[7].

2. 엔터프라이즈

엔터프라이즈(Enterprise)의 정의는 "모든 목적을 가지는 활동들, 또는 경제적인 활동을 목적으로 구성된 단체"이다[8]. 또한 엔터프라이즈는 범위(Scope)개념을 암묵적(Implicity)으로 내포하고 있는 데 일반적으로 우리가 엔터프라이즈라고 하면 가장 넓은 범위를 의미한다.



〈그림 2〉 엔터프라이즈 범위

예를 들면, <그림 2>와 같이 엔터프라이즈 범위는 {도메인1,..,도메인n}으로 구성되는 가장 넓은 범위이고, 그 다음, 도메인 범위는 {시스템1,..,시스템n}으로, 가장 좁은 범위인 시스템 범위는 {컴포넌트1,..,컴포넌트n}으로 구성된다.

3. 아키텍팅

아키텍처의 정의는 "구성요소(컴포넌트)의 구조이고, 구성요소(컴포넌트)들의 상호관계이며, 또한 상호요소(컴포넌트)들의 설계 및 추후 진화를 관리할 수 있는 원칙과 지침"이다[9]

아키텍처의 역할은 i) 복잡한 엔터프라이즈(시스템)의 실체를 간단한 모델로 제시함으로 이해를 용이하게 하고, ii) 컴포넌트에 의해 변화 및 충격을 완화할 수 있고, iii)요구사항 세부명세 및 관계를 정의하며, iv) 설계해야 할 엔터프라이즈(시스템)의 다양한 면을 묘사함으로써 의사소통 수단이 되며, v) 엔터프라이즈(시스템) 재설계 시 변하지 말아야 할 핵심 요소를 지시한다.

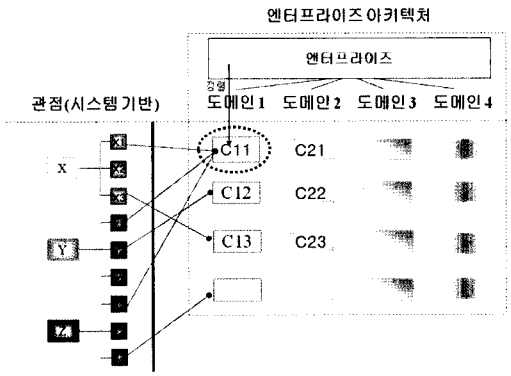
아키텍팅은 "아키텍처를 설계하는 것"이며, 아키텍트는 "아키텍팅을 통하여 아키텍처를 개발하는 사람"이다. 아키텍팅은 아키텍팅 개념 및 원칙에 의해서 수행된다.

아키텍팅 개념은 i) 도메인(Domain)은 엔터프라이즈(시스템)를 복잡도 및 변화 관리를 위해 도메인으로 나누는 것이다. ii) 각 도메인의 분할계층도(Decomposition Hierarchy)를 개발하는 것이다. iii) 관점(View)은 엔터프라이즈(시스템)를 각기 다른 창으로 관찰하는 것이다.

아키텍팅 원칙은 i) 컴포넌트는 적당한 복잡도와 최소한의 결합도 및 최대한의 응집도를 가져야 하고, ii) 분할계층도에 층 개념과 같은 층의 컴포넌트들이 서로 상호 배타적(Mutually Exclusive)이 있기 때문에 다른 층에 영향을 주지 않고 층 내부에서만 변화를 이루게 하여 컴포넌트의 유연성(Flexibility)을 제공한다. iii) 구조적 안정도 원칙에 의하면 아키텍처(시스템)는 여러 컴포넌트로 분리 가능하고, 특정 컴포넌트의 영향을 받지 않고 기능을 수행할 수 있으며, 컴포넌트는 미래로의 진화를 용이할 수 있다. iv) 아키텍처(시스템)에 의해 변화를 예측하고, 변화가 허용되거나 허용되지 않는 컴포넌트는 식별될 수 있어야 한다.

III. 시스템 컴포넌트 품질 요소 재정의

앞장에서 정의한 시스템 품질 요소인 통합성, 정렬성, 상호운용성, 이식성, 확장성, 공유, 유연성 및 재사용성은 암묵적으로 범위를 내포하고 있으나, 지금까지 우리가 이러한 용어를 사용할 때, 어느 범위에서 통합하고, 정렬하며, 상호운용하고, 이식하며, 확장하고, 공유하



〈그림 3〉 컴포넌트 기반 정렬성, 통합성, 상호운용성, 이식성, 확장성, 공유, 유연성 및 재사용성 개념도

며, 유연성을 가지고, 재사용할 것인가를 명시하지 않은 채 막연히 사용하고 있는 실정이다.

본장은 컴포넌트의 품질요소를 앞장에서 정의한 시스템 품질요소를 기반으로 범위개념을 추가하여 재정의 하겠다.

1. 정렬성이란?

정렬성이란 "컴포넌트가 수직적으로 정렬된 형태"로 정의 된다. 정렬성 정도는 "정렬된 컴포넌트가 사용되는 횟수}이고, TCO는 (개발비/정렬성정도, 설치비*정렬성정도, 유지보수비/정렬성정도, 교육훈련비/정렬성정도, 도태비*정렬성정도}이고, 정렬성이 있는 컴포넌트 C의 품질 정의는 {정렬범위, 정렬형태, 정렬성정도, 정렬대상, TCO}으로 정의 된다.

예를 들면, <그림 3>의 C_{11}의 정렬 형태는 {엔터프라이즈-도메인1-C_{11}}이고, 정렬성정도는 시스템 X1, Y1, Z1가 C_{11}에 의해 정렬되기 때문에 3이고, 컴포넌트 C_{11} 의 품질은 {정렬범위: 엔터프라이즈, 정렬형태:(엔터프라이즈-도메인1-C_{11}), 정렬성정도=3, 정렬대상:(X1, Y1, Z1), TCO(개발비/3, 설치비*3, 유지보수비/3, 교육운련비/3, 도태비*3)}이다.

2. 통합성이란?

통합성이란 "컴포넌트(들)를 통합하는 능력"이고, 통합성 정도는 "컴포넌트에 의해 통합된 컴포넌트 숫자"이고, TCO는 (개발비/통합성정도, 설치비*통합성정도, 유지보수비/통합성정도, 교육훈련비/통합성정도, 도태비*통합성정도)이며, 통합성이 있는 컴포넌트 C의 품질은 {통합성, 통합성적용범위, 통합성정도, 통합대상, TCO}로 정의 된다.

예를 들면 <그림 3>의 C_{11}은 통합성이 있는 컴포넌트로, 통합정도는 3이다. 왜냐하면 C_{11}는 시스템 X1, Y1, Z1가 통합된 지점이기 때문이다. C11의 품질은 {통합범위:엔터프라이즈, 통합성정도:3, 통합대상(X1, Y1, Z1), TCO(개발비/3, 설치비*3, 유지보수비/3, 교육훈련비/3, 도태비*3)}이다.

3. 상호운용성이란?

상호운용성이란 "컴포넌트가 시스템(들)에 사용될 수 능력"이고, 상호운용성 정도는 "컴포넌트가 시스템(들)에 사용 되는 횟수"이고, TCO는(개발비/상호운용성정도, 설치비*상호운용성정도, 유지보수비/상호운용성정도, 교육훈련비/상호운용성정도, 도태비*상호운용성)이며, 상호운용성이 있는 컴포넌트 C의 품질은 {상호운용성 적용범위, 상호운용성 정도, 상호운용대상, TCO }이다.

예를 들면 <그림 3>의 C_{11}은 시스템 X1, Y1, Z1에 대해 상호운용성이 있고, 상호운용성정도는 3이다. 또한 컴포넌트 C_{11}의 품질은 {상호운용성 적용범위: 엔터프라이즈, 상호운용성 정도:3, 상호운용대상:(X1, Y1, Z1), TCO(개발비/3, 설치비*3, 유지보수비/3, 교육훈련비/3, 도태비*3)}이다.

4. 이식성이란?

이식성은 "컴포넌트가 하나의 플랫폼에서 다른 플랫폼으로 쉽게 전환 될 수 있는 능력"이고, 이식성 정도는 "하나의 컴포넌트가 플랫폼(들)에 사용되는 횟수"이고, 이식성 있는 컴포

넌트 C의 품질은 {이식성적용범위, 이식성정도, 이식대상, TCO(개발비/이식성정도, 설치비*이식성정도, 유지보수비/이식성정도, 교육훈련비/이식성정도, 도태비*이식성정도)}이다.

예를 들면 만약 <그림 3>의 X1, Y1, Z1이 플랫폼이면, C_{11}은 3정도의 이식성이 있고, 이식성이 있는 컴포넌트 C_{11}의 품질은 {이식성적용범위: 엔터프라이즈, 이식성정도:3, 이식대상:(X1 Y1, Z1), TCO(개발비/3, 설치비*3, 유지보수비/3 교육훈련비/3, 도태비*3}이다.

5. 확장성이란?

확장성은 "컴포넌트가 여러 컴퓨터(들)에 사용 될 수 있는 능력"을 나타내는 것이다. 여기서 컴퓨터(들)는 {슈퍼컴퓨터, 메인컴퓨터, 미니컴퓨터, 마이크로컴퓨터}이다. 확장성 정도는 "하나의 컴포넌트가 여러 종류의 컴퓨터(들)에 사용되는 횟수"이고, 확장성이 있는 컴포넌트 C의 품질은 {확장성 적용범위, 확장성정도, 확장대상, TCO(개발비/확장성정도, 설치비*확장성정도, 교육훈련비/확장성정도, 도태비*확장성정도}이다.

만약 <그림 3>의 X1, Y1, Z1이 컴퓨터이면 C_{11}은 확장성정도가 3이고, 확장성이 있는 컴포넌트 C11의 품질은 {적용대상: 엔터프라이즈, 확장성정도:3, 확장대상(X1, Y1, Z1), TCO(개발비/3, 설치비*3, 유지보수비/3, 교육훈련비/3, 도태비*3)}이다.

6. 공유란?

<그림 3>와 같이 공유란 "컴포넌트가 사용되는 것"이고, 공유정도는 "컴포넌트가 사용되는 횟수"이다. 그러므로 공유되는 컴포넌트 C의 품질은 {공유범위, 공유정도, 공유대상, TCO(개발비/공유정도, 설치비*공유정도, 유지보수비/공유정도, 교육훈련비/공유정도, 도태비*공유정도)}이다.

예를 들면 <그림 3>의 C11은 시스템 X1, Y1, Z1이 공유하고 있으므로 공유정도는 3이고, 공유 가능한 컴포넌트 C_{11}의 품질은 {공유범위: 엔터프라이즈, 공유정도:3, 공유대상(X1, Y1, Z1), TCO(개발비/3, 설치비/3, 유지보수비/3, 교육훈련비/3, 도태비*3)}이다.

7. 유연성이란?

유연성이란 "컴포넌트가 주변 환경에 적응할 수 있는 능력"이고 유연성 정도 "컴포넌트가 주변 환경에 적응된 횟수"이고, 유연성이 있는 컴포넌트 C의 품질은 {유연성 적용범위, 유연성정도, 유연성대상, TCO(개발비/유연성정도, 설치비*유연성정도, 유지보수비/유연성정도, 교육훈련비/유연성정도, 도태비*유연성정도}이다.

예를 들면 <그림 3>의 C_{11}이 시스템 X1, Y1, Z1 변화에 적응하였다고 하면, C_{11}의 유연성정도는 3이고, 유연성이 있는 컴포넌트 C_{11}의 품질은 {유연성적용범위: 엔터프라이즈, 유연성정도3, 유연성대상(X1, Y1, Z1), TCO(개발비/3, 유지보수비/3, 교육훈련비*3, 도태비*유3)}이다.

8. 재사용성이란?

지금까지 살펴 바와 같이 컴포넌트 기반의 재사용성은 정렬성, 통합성, 상호운용성, 이식성, 공유 및 유연성 개념을 포함하는 가장 넓

은 개념이다. 즉, 재사용은 정렬성에 의한 재사용, 통합성에 의한 재사용, 상호운용성에 의한 재사용, 이식성에 의한 재사용, 확장성에 의한 재사용, 공유에 의한 재사용, 유연성에 의한 재사용 등이다.

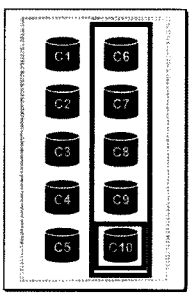
이것을 토대로 재사용성을 정의하면, 재사용은 "컴포넌트가 정렬성, 통합성, 상호운용성, 이식성, 확장성, 공유 및 유연성에 의해 사용될 수 있는 능력"으로, 재사용 정도는 "컴포넌트가 정렬성, 통합성, 상호운용성, 이식성, 확장성, 공유 및 유연성에 의해 사용되는 횟수"이다. 그러므로 재사용성이 있는 컴포넌트 C의 품질은 {재사용 적용범위, 재사용정도(재사용 방법), 재사용대상, TCO(개발비/재사용정도, 설치비*재사용정도, 유지보수비/재사용정도, 교육훈련비/재사용정도, 도태비*재사용정도}으로 나타낼 수 있다. 여기서 재사용 방법은 {정렬성, 통합성, 상호운용성, 이식성, 확장성, 공유, 유연성}이다.

예를 들면 <그림 3>의 C_{11}은 재사용 정도가 3이고, 재사용성이 있는 컴포넌트 C_{11}의 품질은 {적용범위: 엔터프라이즈, 재사용정도(상호운용성):2, 대상(시스템1, 시스템2), TCO(개발비/2, 설치비*2, 유지보수비/2, 교육훈련비/2, 도태비*2)}로 표현된다.

Mary Ewing은 시뮬레이션을 통하여 재사용성 정도에 따른 비용 효과를 <그림 4>과 같이 제시하였다. 이 도표는 컴포넌트의 재사용정도(%)별 재사용(재사용성 정도=2)결과를 기반으로 컴포넌트의 개발비를 계산한 것이다[10].

<그림 4>는 총투자비용이 1,000,000$ 일 때를 기준으로, i) 100% 재사용 시 총투자비용, 즉 1,000,000-(1,000,000/2) = 500,000$, ii) 50% 재사용 시는 총 투자비용, 즉, 1,000,000-(500,000*0.5) = 750,000$, iii) 10% 재사용 시 총투자비용, 즉, 1,000,000-(500,000*0.1)=950,000$를 나타낸 것이다.

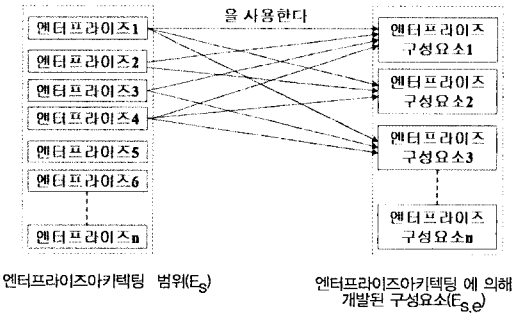
위 사례에서와 같이 재사용 컴포넌트의 TCO비용은 제시한 식의 결과와 같으며 재사용정도가 증가할수록 TCO는 감소함을 알 수 있다. 따라서 컴포넌트의 품질은 재사용성정도에 비례하고 TCO을 감소시킨다. 그러나 컴포넌트를 재사용하더라도 재사용을 위한 부대비용이 요구 될 수 있기 때문에 실제 컴포넌트 TCO는 위에서 제시한 식의 비용보다 다소 증가될 것이다.



재사용정도	투자사금 ($1,000,000)	절감액
10 %	$ 950,000	$ 50,000
50 %	$ 750,000	$ 250,000
100 %	$ 500,000	$ 500,000

〈그림 4〉 재사용성 분석

IV. 엔터프라이즈아키텍팅에 의한 컴포넌트 품질 향상



〈그림 5〉 엔터프라이즈와 엔터프라이즈 구성요소와 관계

<그림 5>과 같이 엔터프라이즈(E_i)와 엔터프라이즈구성요소($E_i.e$) 간의 관계는 엔터프라이즈(E_i)가 엔터프라이즈(E_i)의 구성요소($E_i.e$))를 "사용하는 관계"이다. 또한 엔터프라이즈구성요소($E_i.e_j$)는 엔터프라이즈아키텍팅에 의해 개발된 엔터프라이즈구성요소($E_s.e$)를 "재사용하는 관계"이다. 여기서, 첫째, Es={E1,..,En}는 엔터프라이즈아키텍팅 범위에 포함된 엔터프라이즈(Ei)의 집합이고, 둘째, $E_s.e$={$E_s.e_1$,..,$E_s.e_n$}는 E의 구성요소의 집합이다.

그러므로 엔터프라이즈아키텍팅에 의해 이상적으로 개발된 엔터프라이즈구성요소($E_s.e$)는 재사용성 정도가 최대가 될 것이다.

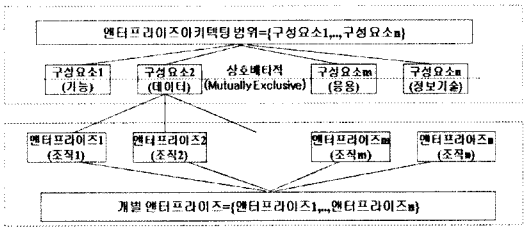
엔터프라이즈아키텍팅이란 "엔터프라이즈 범위에서 아키텍팅 개념 및 원칙을 적용하여 아키텍팅 작업을 수행하는 것"으로, 다음과 같은 단계로 수행한다.

1. 엔터프라이즈아키텍팅 범위 결정 단계

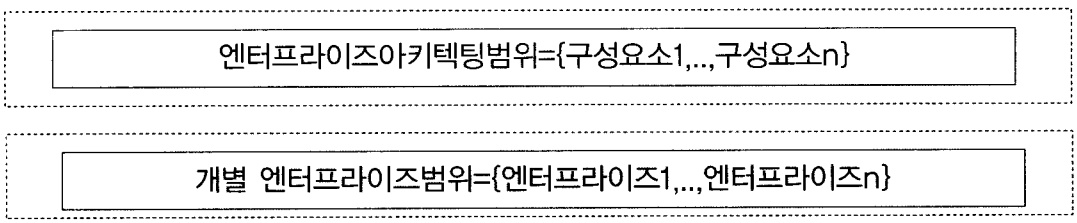
1단계는 <그림 6>과 같이 엔터프라이즈아키텍팅의 범위에 포함될 엔터프라이즈들(E_s)를 결정하는 것이다. 범위결정 시 E_s와 E_i간의 재사용 최대로 하기 위해 엔터프라이즈아키텍팅 대상의 구성요소 {$E_s.e_1$,..,$E_s.e_n$} 와 엔터프라이즈아키텍팅에 의해 개발된 컴포넌트를 재사용할 엔터프라이즈 범위 내에 있는 개별 엔터프라이즈(E_i)의 구성요소{E_ie_1,..,E_ie_n}도 고려한다. 예를 들면 엔터프라이즈아키텍팅을 위한 대상(범위)을 조직1, 2 및 조직3라고 한다면 엔터프라이즈 범위는 E_s={조직1(E_1), 조직2(E_2), 조직3(E_3)}와 같고 엔터프라이즈아키텍처 구성요소를 기능, 데이터, 응용, 정보기술로 분할한다면, 엔터프라이즈범위 구성요소 $E_s.e$는 {기능($E_s.e_1$), 데이터($E_s.e_2$), 응용($E_s.e_3$), 정보기술($E_s.e_4$)}로 분할하고 이와 관련된 개별엔터프라이즈의 구성요소E_i는{기능($E_i.e_1$), 데이터($E_i.e_2$), 응용($E_i.e_3$), 정보기술($E_i.e_4$)}와 같이 설정한다.

2. 도메인 분할 단계

이 단계는 <그림 7>와 같이 엔터프라이즈아키텍팅 대상을 도메인으로 분할한다. 도메인은



〈그림 7〉 도메인 분할



〈그림 6〉 엔터프라이즈 범위

1단계에서 설정된 엔터프라이즈아키텍팅 대상을 고려하여 상호배타적이며 플랫폼에 독립적으로 분할하며 이러한 방식은 앞으로의 아키텍팅 단계에 지속적으로 적용된다. 즉 E_s를 $E_s.e_1$, $E_s.e_1$ 및 $E_s.e_n$으로 분할하고, 엔터프라이즈아키텍팅 범위 내에 있는 개별 엔터프라이즈(E_i)를 위해서는 E_s을 E_1, E_2 및 E_n으로 분할한다.

예를 들면 엔터프라이즈아키텍팅을 위한 도메인은 기능($E_s.e_1$)도메인, 데이터($E_s.e_2$)도메인, 응용($E_s.e_3$)도메인, 정보기술($E_s.e_4$)도메인으로 분할하고, 개별 엔터프라이즈를 위한 도메인은 조직1(E_1)도메인, 조직2(E_2)도메인, 조직3(E_3)도메인으로 분할한다.

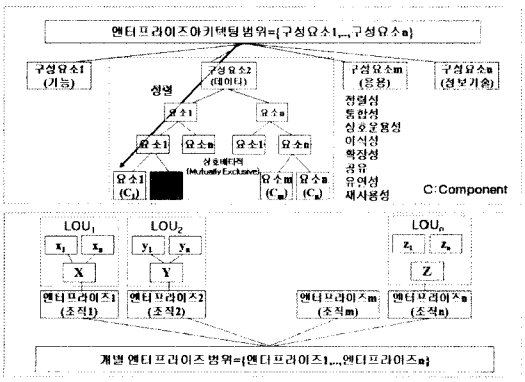
3. 분할계층도 작성

두 번째 단계는 <그림 8>과 같이 2단계 분할한 도메인을 기반으로 분할계층도를 개발하는 것으로, 첫째, $E_s.e_1$도메인 분할계층도, $E_s.e_2$도메인 분할계층도 및 $E_s.e_n$도메인 분할계층도를 개발하고, 둘째, 엔터프라이즈아키텍팅에 의해 개발된 컴포넌트를 재사용할 수 있는 논리적운영단위(LOU:Logical Operating Unit)를 기반으로 개별 엔터프라이즈(E_i)의 분할계층도를 개발한다.

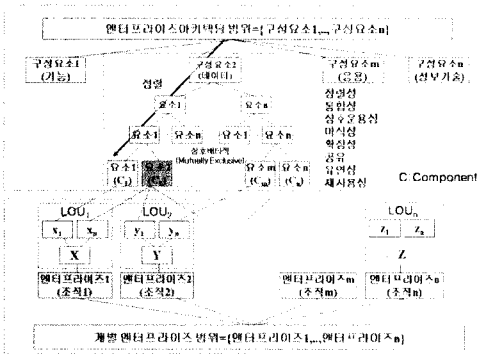
여기서 논리적운용단위는 논리적인 업무 수행단위로 엔터프라이즈의 기능은 논리적운영단위에 의해 수행되어지며 프로세스, 정보시스템 등을 예로 들 수 있다. 예를 들면 엔터프라이즈범위의 도메인을 위한 계층분할도는 기능($E_s.e_1$)분할계층도, 데이터($E_s.e_2$)분할계층도, 응용($E_s.e_3$)분할계층도, 정보기술($E_s.e_4$)분할계층도와 같고, 개별 엔터프라이즈를 위한 계층분할도는 조직1(E_1)분할계층도(프로세스), 조직2(E_2)분할계층도(시스템), 조직3(E_3)분할계층도(데이터)와 같이 분할한다.

4. 재사용 단계

세 번째 단계는 <그림 9>와 같이 엔터프라이즈아키텍팅 범위(E_s)를 기반으로 한 엔터프라이즈아키텍팅에 의해 개발된 컴포넌트들을 엔터프라이즈아키텍팅 범위 내에 있는 각각의 엔터프라이즈(E_i)가 재사용하는 것이다. <그림 9>에서 조직1의 LOU_1과 조직2의 LOU_2가 요소2(C_2)를 재사용하였다.



〈그림 8〉 계층분할도



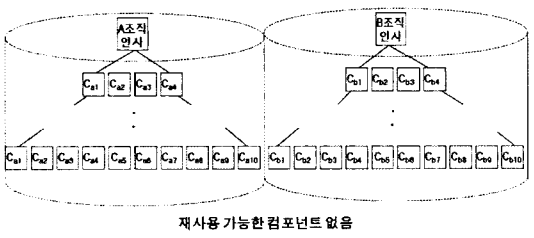
〈그림 9〉 시스템 재사용단계

5. 적용사례

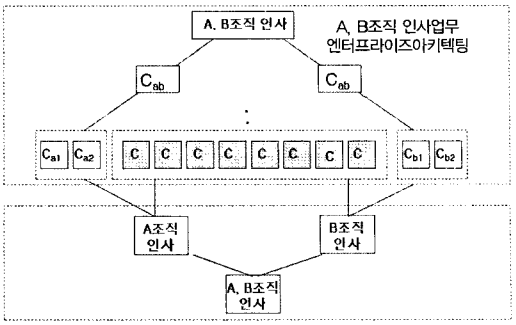
<그림 10>과 <그림 11>은 엔터프라이즈아키텍팅 범위(E_s)가 {조직A, 조직B}이고, 각 엔터프라이즈아키텍팅 대상이 조직의 인사업무($E_s.e_i$=인사업무)인 사례를 엔터프라이즈아키텍팅에 의해 컴포넌트를 재사용한 것과 시스템아키텍팅에 의해 재사용을 전혀 할 수 없는 것을 비교한 것이다.

<그림 10>에서 보는 바와 같이 조직A의 인사업무와 조직B의 인사업무는 시스템을 기반으로 하는 아키텍팅으로 인사업무 간에 존재하는 재사용가능한 구성요소를 식별할 수 없어 재사용으로 향상될 수 있는 품질을 향상 할 수 없다.

반면 <그림 11>에서는 엔터프라이즈아키텍팅을 위해서 1단계로 엔터프라이즈아키텍팅 범위를 {A조직, B조직}으로, 엔터프라이즈아키텍팅 대상은 {A조직인사업무, B조직인사업무}로 결정했다. 2단계로 조직에 독립적인 인사업무 분할계층도를 개발하였다. 이러한 계층분할의 결과로 엔터프라이즈를 구성하는 컴포넌트를 C_{a1}, C_{a2}, C_{b1}, C_{b2}, 와 8개의 C로 계층 분할하였다. 3단계는 A조직 및 B조직이 개발된 인사업무의 컴포넌트를 사용하여 시스템을 구성하는 단계이다. 조직1의 인사시스템과 조직2의 인사시스템이 엔터프라이즈아키텍팅에



〈그림 10〉 시스템기반아키텍팅



〈그림 11〉 엔터프라이즈아키텍팅에 의한 인사업무 컴포넌트 품질 향상 사례

의해 개발된 인사업무 컴포넌트를 재사용한다.

그림에서 보는 바와 같이 A조직과 B조직에 독특한 컴포넌트C_{a1}, C_{a2}, C_{b1}, C_{b2}를 제외한 8개의 C 컴포넌트가 재사용되어 시스템의 품질이 향상되었다.

일반적으로 조직1, 조직2의 인사업무를 분석하면 많은 부분이 같을 것이다. 예를 들면, 조직1, 조직2의 인사업무는 자연의 법칙인 파레토 법칙에 의하면 80%는 동일한 부분이고 나머지 20%정도가 각 조직의 고유한 인사업무가 될 것이다. 그러므로 엔터프라이즈아키텍팅에 의해 인사업무를 개발하면, 80%의 동일한 컴포넌트를 식별하여 재사용할 수 있다는 것이다.

그러나 만약 각 조직별로 작은 범위로 아키텍팅 작업을 수행한다면, 재사용가능한 컴포넌트를 만들기는 어려울 것이다.

V. 결 론

정보기술아키텍처 및 엔터프라이즈아키텍처는 시스템 품질 요소인 상호운용성, 이식성,

확장성 및 재사용성 등을 향상하기 위한 방법 중에 최선의 방법으로 인식되고 있다.

본 논문에서는 엔터프라이즈아키텍팅 개념을 기반으로 시스템 컴포넌트의 품질을 향상할 수 있는 방법을 제시하기 위해 먼저 아키텍팅 개념을 기반으로 엔터프라이즈아키텍팅에 대하여 살펴보았다. 둘째, 시스템 컴포넌트의 품질 요소를 정의하기 위하여 시스템기반 상호운용성, 이식성, 확장성, 및 TCO개념을 시스템 컴포넌트개념으로 재정의 하였으며, 셋째, 이를 기반으로 시스템 컴포넌트의 품질요소인 재사용성을 정의하였고, 마지막으로, 엔터프라이즈아키텍팅에 의해 시스템 컴포넌트의 재사용성이 향상됨을 사례를 통하여 제시하였다.

향후 본 연구를 발전시키기 위해 도메인 컴포넌트의 분할계층도를 작성하기 위한 정형화되고 구체적인 방법에 대한 연구와 도메인 분류 방법에 대한 추가적 연구가 필요하다.

참고문헌

[1] Ian Sommerville, Software Engineering 6th Ed, Chap3, Addison Wesly, 2001

[2] Peter Bernus & L Nemes & Gunter Schmidt, Handbook on Enterprise Architecture, P630, Springer, 2003

[3] U.S. Customs Service, Enterprise Architecture Blueprint, 1999

[4] 이태공, 정보기술아키텍처 편람, 국방대학교, p461, 2003

[5] Institute of Electrical and Electronics Engineers. IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries, IEEE, 1990

[6] Gartner Group Glossary Term, http://www3.gartner.com/6_help/glossary/Gartner_IT_Glossary.pdf, 2004

[7] http://www.itapmo.org/board/include/download.php?no=3&db=seminarforum&fileno=1, 2008.10.8. 방문

[8] 이태공, 박성범, 이헌중, 정보기술 아키텍처, 기한재, 2000

[9] U.S. CIO Council, Federal Enterprise Architecture Framework V1.1, 1999

[10] Mary Ewing, The Economic Effects of Reusability on Distributed Simulations, Proceedings of the 2001 Winter Simulation Conference, 2001

저자소개

임 남 규

1990년 공군사관학교 전산학학사
2004년 미국 플로리다 대학 전산학석사
2008년-현재 아주대학교 NCW공학과 박사과정
1994년-2002년 공군 비행대대 편대원
2004년-2007년 공군작전사령부 작전전산소

주관심 분야 : Architecture/Domain/Application
Engineering Enterprise
Architecture Quality, SOA

이 태 공

1976년 공군사관학교 전자과학사
1986년 미국 해군대학원 체계관리석사
1991년 미국 웨인 주립대학 전산학박사
1995년-2007년 국방대학교 전산정보학과 교수
2007년-현재 아주대학교 정보통신대학원 교수

주관심 분야 : Architecture/Domain/Application
Engineering IT Strategy. Enterprise
Integration COT-Based S/W
Acquisition