CBD 프로세스가 갖춰야 할 기본 요소들

Computer Associates Paul Allen
쿨소프트웨어 코리아 김경주

1. 서 론

CBD(Component Based Development)를 위한 여러 개발 프로세스들이 이미 소개되었지만, 경험 있는 전문가들에 의해 더 좋은 프로세스들이 탄생되기 위해 지금 이 시간에도 진통을 겪고 있을 지도 모른다. 최상의 프로세스는 이것이다라고 이 글을 통해 소개하는 것보다, e- Business를 실현하기 위해 사용되는 CBD 접근방식에 사용될 프로세스가 기본적으로 갖춰야 할 요소들을 포괄적으로 제시함으로써, 이 시대에 당면한 복잡한 문제를 해결하는데 도움을 주는 것이 보다 합리적이라고 생각한다. 이러한 의도에 입각하여, Perspective라는 CBD방법론으로 익히 잘 알려져 있으며, 현재는 Computer Associates사의 Principal Component Strategist인 Paul Allen이 그의 저서 "Realizing e-Business with Components"[1]에서, 25년간의 대규모 비즈니스 시스템 구축 경험을 바탕으로 제시한, e-Business를 위한 CBD 프로세스 프레임워크에 대해 알아보고자 한다.

오늘날의 조직은, 일반적으로 사람들이 생각하고 있듯이, 깨끗한 화이트 슬레이트를 사용하여 처음부터 프로젝트를 시작하는 그런 사치스러움은 갖고 있지 않다. 그런데, 많은 조직들은 선형적인 생산 라인의 방법으로, 단일 솔루션(point solutions)의 개발에 적합한 프로세스를 사용해오고 있다. 이것은, 부서와 부서간에, 혹은 인터넷을 통하여 조직적인 경계를 초월해서 행해지고 있는 e-Business 프로세스를 지원하기 위한 복합적인 솔루션(hybrid solutions)을 원하는 오늘날의 요구사항과는 잘 맞지 않는다. 가능한 많은 기능들을 재사용하거나 취득함으로써 이러한 복잡한 비즈니스 문제를 해결해야 할 필요성이 대두되고 있는 것이다. 그런데, 전통적인 소프트웨어 엔지니어링 프로세스들은 이것이 가능하도록 설계되어지지 않았다. 따라서 새로운 도전에 보다 잘 맞는 새로운 접근방법들이 필요하며, 이러한 접근법들은 비즈니스 요구사항에 대응하여, 선형적인 소프트웨어 개발 방식에서 배워 온 교훈에 바탕을 두고 만들어 져야 할 것이다.[2]

이 글에서는, 대부분의 방법론에서는 매우 부족한, 실질적인 컴포넌트 지향의 프로세스 프레임워크에서 사용되는 컴포넌트 기반 모델링 테크닉들을 배치하기 위한 컨텍스트를 설정하고 있다. 대부분의 현재 프로세스들은 실제의 엔터프라이즈 환경에 적용하기에는 너무 과도하게 자세하다. 이 글의 목표는, 이 글에서 소개되는 모델링 테크닉들을 적용할 수 있는 실현 가능한 프로젝트 관리 프레임워크를 제공하는 것으로 패턴, 체크리스트, 힌트 그리고 팁들에 초점이 맞춰져 있다.

1) published by Addison Wesley Longman, ISBN: 0 201 67520 X, Pearson Education 2001

2) 예를 들어, James Martin(Martin, 1991)과 여러 사람들에 의해 착안되었듯이, rapid application development(RAD)는, Joint Application Development(JAD), 프로토타이핑 그리고 타임 박싱(time-boxing)과 같은 많은 유용한 테크닉들을 제안하였다. 더 최근에는, Dynamic Systems Development Method (DSDM) 컨소시엄이 RAD와 CBD의 DSDM과의 접목을 위해 실제 가이드를 소개했다.

2. CBD 프로세스 프레임워크

e-Business 시스템의 개발은, 비즈니스 프로세스 컨설턴트, 소프트웨어 설계자, 레거시 전문가, 그래픽 디자이너 그리고 서버 엔지니어들과 같은 서로 다른 영역에서의 서로 다른 종류의 스페셜리스트들의 공동 작업을 통하여 이루어진다. 우리는 이런 다양한 스킬셋을 다룰 수 있는 잘 정리된 프레임워크가 필요하며 그것을 도와줄 수 있는 트랙기반의 패턴을 여기서 소개하고자 한다. 또한, 우리가 만들고자 하는 여러 종류의 산출물에 대한 일반적인 이해를 갖는 것도 중요하다. 이 글에서 CBD 프로젝트에 잘 맞는 여러 종류의 산출물들이 설명될 것이다. 그 다음, 트랙 기반의 패턴과 산출물로 구성된 전체적인 프로세스 프레임워크 내에서 각각의 모델링 테크닉이 융통성 있게 적용될 수 있을 것이다.

2.1 트랙 기반의 패턴

트랙 기반의 패턴은 CBD를 잘 컨트롤할 수 있고 또 수행되어질 역할에 따른 구성원들의 조직에 잘 맞춰진 프레임워크를 제공한다.

그림 1에서 볼 수 있듯이, 솔루션 조립과 컴포넌트 공급을 병행으로 수행하는 방식을 고려하는 것이 도움이 된다. 이것은 종종 '트윈 트랙 개발(twin track development)' (Allen and Frost, 1998)이라고 불려지기도 하는데 소비자(Consumers)와 생산자(Producers)는 신속한 비즈니스 솔루션 공급과 높은 품질의 컴포넌트를 얻고자 하는 각각의 요구사항에 맞는 별도의 프로세스를 따른다. 솔루션을 조립해서 궁극적으로 e-Business 솔루션을 제공하는 사람들은 컴포넌트 공급 트랙(Component Provisioning Track)에 의해 만들어진 컴포넌트들을 수확하기 위해 노력할 것이다. 동시에, 컴포넌트 공급자들은 컴포넌트로 성장시키기 위해 필요한 솔루션이라는 씨를 찾고자 노력할 것이다.

그림 1에서 보면, 우리는 트윈 트랙 프로세스가 서로 다른 방법에 의해 기동됨을 알 수 있다. 솔루션 조립 트랙은 시기 적절한 비즈니스 솔루션, 예를 들면, 인터넷을 통한 직접 판매와 같은 솔루션을 공급하고자 하는 필요성에 의해 시작된다. 조립은 쓸만한 컴포넌트를 찾는 작업과 필요하다면, 컴포넌트 공급 트랙에 새로운 컴포넌트의 필요성을 제시하는 등의 작업까지도 포함한다.

대조적으로, 컴포넌트 공급트랙은 구체적인 솔루션 요구와는 관계없이 기동된다. 즉, 비즈니스 인프라구조 컴포넌트(예를 들면, 공통적으로 필요한 상품 룰 엔진 컴포넌트)와 같은 재사용 가능한 컴포넌트에 대한 요구와 같이, 비즈니스 재사용에 대한 필요성이 제기됨에 따라 독립적으로 기동되어진다.

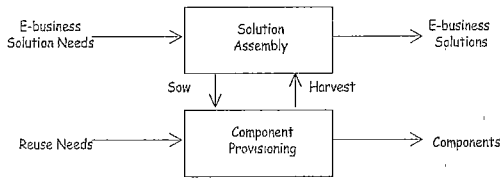


그림 1 트윈 트랙 프로세스

기존 시스템, 팩키지 그리고 데이터베이스의 재사용에 대한 압박은 갈수록 심해진다. 통합(Integration) 프로젝트들(Linthicum, 2000)은 데이터와 프로세스의 중복을 제거하고 종종 다양한 기술에 의해 구현된 기존 시스템과 팩키지 패밀리간의 일관성을 가져가기 위해 EAI용 소프트웨어를 사용한다. 그림 2에서는, 레거시 소프트웨어를 이용하기위해 재사용에 대한 요구사항(Reuse Requirements)이 어떻게 효과적으로 필터링 되어지는지를 볼 수 있다.

트랙 기반의 패턴은 아무것도 없는 상태에서

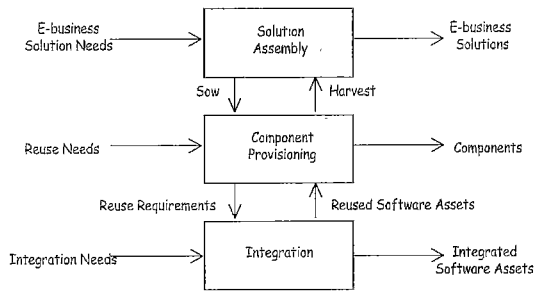


그림 2 트윈 트랙 프로세스와 통합(The twin track process and Integration)

시작되지는 않는다. 그 전에 e-Business 프로세스 향상 단계가 있다. e-Business 프로세스 향상(e-Business Process Improvement) 단계는, 그림 3에서 보여주는 바와 같이, CBD를 위한 적절한 비즈니스 컨텍스트를 제공한다. 전체적인 e-Busienss 향상 계획은, CBD를 이용하여 e-Business로 전환하고자 하는데 있어서 매우 중요하다. 이 e-Busienss 향상 계획은 아키텍처 계획과 비즈니스 모델에 대한 방향을 제시하며 e-Business 솔루션을 위한 구체적인 프로세스를 이해하는데 포커스를 둔다.

프로세스는 계속 진화하고 진행중임을 주목하여야 한다. 소프트웨어 프로젝트로부터 나온 결과는 e-Business를 통한 경험을 고려한 재평가를 위해 e-Business 프로세스 향상 작업으로 피드백 되어진다. 변화(Change)에 대한 요구가 반드시 관리되어야 하며 이와 유사하게, 컴포넌트들은 아키텍처 계획과 관련하여, 점차적인(progressive) 정련(refinement) 프로세스를 통해 평가되어진다. 아키텍처 계획은 여러 프로젝트들이 참고할 수 있는 커다란 밑그림을 제공하는 상위 수준의 엔터프라이즈 컴포넌트 아키텍처를 만들어 낸다.

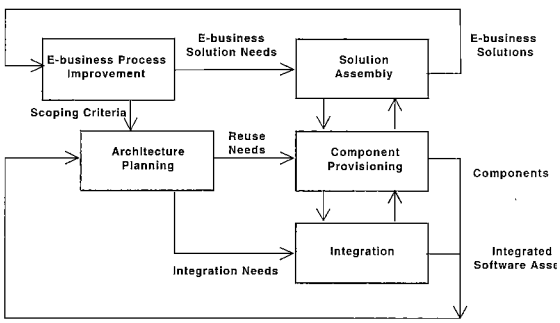


그림 3 트랙 기반의 프레임워크

트랙 기반의 프레임워크를 뒷받침해주는 것은 그림 4에서 보여지는 바와 같이, 효과적인 인프라구조물들이다. 이러한 컴포넌트들은 COM+, EJB, CORBA와 같은 컴포넌트, 인터넷 표준, 형상관리에 대한 지원도 포함한다. 요즘의 일부 컴포넌트 관리 툴은 모델 중심의 접근방법과 조화를 이루어 저장소 내에 컴포넌트 정보를 갖고 있는 컴포넌트 목록을 제공하기도 하고 컴포넌트 검색, 설치 그리고 등록 등을 할 수 있는 기능까지도 제공한다.

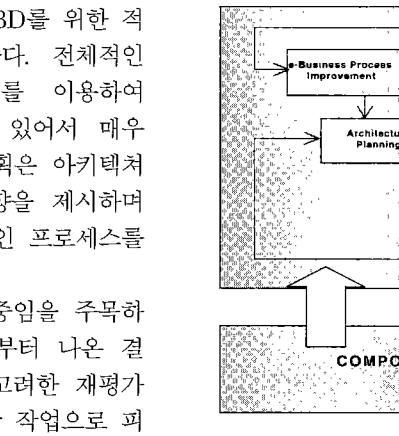


그림 4 트랙 기반의 프레임워크와 컴포넌트 인프라 구조

2.2 산출물(Deliverables)

실제 프로젝트의 라이프 사이클과는 관계가 거의 없는, 관행적으로 사용되는 타스크들을 나열하는 실수를 범하게 되는 소프트웨어 프로세스들이 많다. 이렇게 되는 이유는, 이런 종류의 프로세스가 '무엇(what)'을 이해하기 전에 '어떻게(how)'에 초점을 두기 때문이다. 그러므로, 테크닉과 세부 작업을 고려 하기 전에 우리는 어떠한 산출물을 만들고자 하는가에 더 포커스를 맞출 필요가 있다. 이 글의 후반부에서, 각각의 산출물 종류별로 어떤 종류의 모델들이 있을 수 있는지를 보게 될 것이다.

산출물들은 프로젝트에 대해 목표를 제공하고, 프로젝트의 작업 진도를 파악하고, 소프트웨어 납품에서의 공통적인 이해를 확인하기 위해 필요하다. 'n'개의 산출물 세트와 이에 따른 테크닉들이 있을 수 있으며, 이러한 테크닉들은 우리가 문화와 산업 형태 별로 서로 다른 조직에서 발견해낼 수 있는 것들일 수도 있다. 그러나 여기서, 모델링 테크닉들에 대한 앞뒤 컨텍스트를 제공하기 위해 일반적인 산출물 세트를 제공한다.-이러한 테크닉들은 우리 자신의 조직에 맞게 조정해서 사용할 수 있다. 전형적인 CBD 프로젝트에서는 다음과 같은 다섯 종류의 산출물[3]이 발견되

어질 수 있다.

1) E-Business 프로세스 향상 계획(E-Business Process Implementation Plan)
2) 소프트웨어 요구사항(Software Requirements)
3) 컴포넌트 아키텍처(Component Architecture)
4) 행위 명세(Behavior Specification)
5) 구현(Implementation)

구현은 여러 가지의 더 자세한 산출물, 즉, 이 글의 범위에서 벗어나는 상세한 산출물들을 포함한다.

e-Business 프로세스 향상 계획 사이클은 신속하게(몇 달이 아니라 몇주 내에) 완성되어야 하며 비즈니스와 IT 양쪽을 대표하는 중견 관리자의 참여가 있어야 한다. 비즈니스 모델링은 현업 사람들에 의해 쉽게 이해되어 지는 방식으로 비즈니스 요구사항을 이해하기 위해 사용되어진다. 그러나, 동시에 어떤 새로운 소프트웨어를 만들기 위해 최소한의 번역이나 재작업만을 통해 기능적이거나 비기능적인 요구사항으로 바로 사용될 수 있도록 만들어 져야 한다. e-Business 프로세스 향상 계획은 소프트웨어 요구사항 프로젝트와 아키텍처 계획(엔터프라이즈 아키텍처)에 대한 기준을 잡기 위한 비즈니스를 포함해야 한다.

소프트웨어 요구사항 문서는 광범위한 개발 일정이 짜여질 수 있도록 충분한 정보와 함께 제안된 소프트웨어의 범위를 담고 있다. 소프트웨어 요구사항은 사용자 참여와 정확도 검증을 위해 프로토타이핑 되어질 수 있다. 요구사항을 만족시키기 위해 사용될 수 있는 기존의 분석 패턴과 소프트웨어 자산들도 평가되어 진다. 후자는 기존의 시스템, 데이터베이스 그리고 이용 가능한 소프트웨어 패키지, 컴포넌트 그리고 인터페이스를 포함한다. 이것은 재사용의 기회를 초기에 발견하고 요구사항을 잘 이해하고 통합할 필요가 있는지 등을 평가하기 위해 중요하다.

컴포넌트 아키텍쳐 문서는 프로젝트와 엔터프라이즈, 이 두 단계로 작업 되어 진다. 기반이 되는 프로젝트와 컴포넌트 아키텍쳐는 프로젝트 경험과 소프트웨어 납품을 고려하여 점차적으로 정련되어 진다.

행위 명세[4]는 요구된 소프트웨어 행위에 대한 완전한 그리고 정확한(precise) 정의를 제공한다. 내부 설계 내용은 제외되고 외부적으로 보이는 소프트웨어 행위가 설명되어 진다. 그러나, 내부 설계에 대한 제약사항은 언급되어 진다. 그러한 제약 사항은 인터페이스 의존성, 비 기능적인 요구사항, 불변사항 등의 형태로 나타난다.

행위 명세에는 두 가지 형태가 있다. **인터페이스** 명세는 소비자의 관점에서 인터페이스의 행위를 명시한다. 이들 명세는 인터페이스에 의해 제공되는 서비스들에 대한 계약적인 명세와 그 인터페이스가 관여하는 정보에 대한 목록을 포함한다('인터페이스 타입 모델').

컴포넌트 명세는 컴포넌트 공급자(provisioner)가 소프트웨어 단위[5]로써 구현하고자 계약하는 인터페이스 집합에 대한 행위를 명시한다. 이 문서는 공급자들에게 제약을 가하는 인터페이스 의존성, 불변사항 그리고 비 기능적인 요구사항등을 포함한다. 그 어떤 구현도 해당 컴포넌트 명세에 따라야 하며 인터페이스와 컴포넌트 명세는 목록에 등록되어야 하고 컴포넌트 관리 도구를 사용하여 공개되어야 한다.

2.3 통합(Integration)

통합 프로젝트는 보다 전술적인 레벨(tactical level)로 존재할 수 있으며, 독립된 레거시 시스템의 재사용 작업을 포함한다. 이런 프로젝트들은 간섭적이지 않으며(non-invasive), 단지 컴포넌트 공급에 사용될 수 있도록, 랩핑되거나 적용될 수 있는 기존 서비스를 식별 해내는 작업을 포함한다. 반면, 레거시 개선(renewal) 프로젝트는 상대적으로 간섭적이며 컴포넌트 공급에 사용

3) 종종 관련된 산출물 각각을 반영하는 별도의 단계(stages)들로 프로젝트를 분리하는 것이 편리할 때가 있다.

4) 여기서 우리는 행위 명세에 대한 컴포넌트 관련 측면에 포커스를 맞출 것이다. 전통적인 행위 명세문서는 솔루션(예를 들면, '온라인 혹은 배치', '싱글 프로세서 혹은 분산 프로세서')에 대한 스크린 레이아웃, 리포트 포맷, 예제, 스크린 검증 룰과 비기능적인 요구사항과 같은 전통적인 '외부 설계' 관련 정보를 포함할 수도 있다는 것을 주목하는 것이 중요하다.

5) 즉, '컴포넌트 구현' 형태를 말한다.

되어질 인터페이스를 뽑아내기 위한 준비로 레거시 시스템의 리엔지니어링 부문을 포함한다.

다른 통합 프로젝트는 보다 전략적이다. 그러나, 액티비티의 기본 패턴(계획, 요구사항, 아키텍쳐, 명세 그리고 구현)은 똑같다. 전략적인 통합 프로젝트는 컴포넌트 공급 프로젝트에서 사용될 수 있도록 통합적인 서비스를 제공할 수 있어야 한다. 레거시 모델들은 기술적이거나 전략적인 통합 작업에 도움이 될 수 있도록 사용될 수 있다. 모델링 기능이 EAI 미들웨어와 연결될 수 있는 도구가 이 작업에서는 매우 유용하다.

2.4 트랙기반 패턴에서의 산출물 구성

그림 5는 산출물과 트랙기반 패턴과의 관계를 보여준다. 소프트웨어 요구사항, 프로젝트 아키텍쳐, 행위 명세 산출물은 솔루션 조립, 컴포넌트 공급, 통합 트랙 중 임의의 트랙에서 발생하는 프로젝트 컨텍스트에 적용된다. 그러나, 나중에 다시 보겠지만, 사실상의 스코핑은 라이프 사이클에 걸쳐 정규적인 포인트에서 발생하며 결과적으로 혼합(Hybrid) 프로젝트가 된다. 그러므로, 예를 들어, 하나의 솔루션 조립 프로젝트는 별도의 더 적은 조립, 공급, 통합 프로젝트들로 나눠질 수 있다.

예를 들어, 소프트웨어 요구사항 테크닉은 비즈니스 프로세스 향상 프로젝트내에서 사용될 수도 있다. 이러한 소프트웨어 요구사항은, 요구사항과 차이가 나는 디자인을 미리 발견해내는 수단으로서, 프로토타이핑과 연결될 때 특별히 유용하다. 이것은 또한, 비즈니스 프로세스 향상과

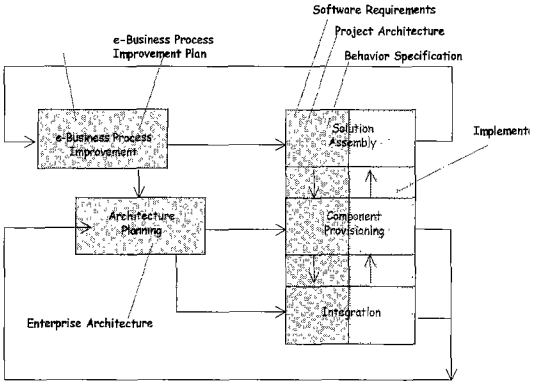


그림 5 트랙 기반 프레임워크에서의 산출물

솔루션 조립간의 차이가 흐려지는 그런 e-Business에 특히 유용하다.

3. 재사용 체크포인트

전통적인 프로세스에서 나올 수 있는 '우리가 소프트웨어를 직접 제작할 것인가 아니면 구매할 것인가?' 라는 선택보다는, 이제, CBD 프로세스에서의 질문에는 다음과 같은 것들이 있을 수 있다.

- 어떤 컴포넌트를 구입하고 또 어떤 것들은 직접 개발할 것인가?
- 비즈니스 요구사항에 잘 맞는 비즈니스 컴포넌트를 어떻게 스코핑하고 식별해낼 것인가 ?
- 80%의 솔루션을 20%의 시간과 비용으로 획득하는 것이 적절한가?
- 당면한 문제를 해결하기 위해 재사용되고 통합될 수 있는 소프트웨어 팩키지가 있는가 ?
- 당면한 문제를 해결하기 위해 재사용되고 통합될 수 있는 레거시 소프트웨어 자산이 있는가 ?
- 당면한 문제해결을 돕기 위해 사용될 수 있는 레거시 모델이 있는가 ?
- 당면한 문제를 해결하기 위해 확장할 수 있는 컴포넌트 프레임워크[6]가 있는가 ?
- 컴포넌트 구현을 아웃소싱할 수 있는 기회가 있는가?
- 가장 적합한 제품을 어떻게 찾아내고 또 그것을 어떻게 통합할 수 있을 것인가 ?

이 시점에서, 표 1에서와 같이, 트랙 기반 프레임워크에 사용될 수 있는 재사용 체크포인트를 만들기 위한 가이드라인을 고려해보는 것이 좋다.

4. CBD 프로세스

전통적인 시스템과는 달리, e-Business 시스

6) 컴포넌트 프레임워크는 사용자에 의해 확장되거나 특화되어질 수 있도록 설계된 컴포넌트 인터페이스와 의존성의 집합체이다. 사용자는 단지 한 개의 인터페이스만을 특화하기 위해 선택할 수도 있다. 반면, 전체 프레임워크는 사용자 시스템의 아키텍쳐적인 기초로서 사용되어질 수 있다.

표 1 재사용 체크 포인트

산출물(Deliverable)	솔루션 조립(Solution Assembly)	컴포넌트 공급(Component Provisioning)
e-Business 프로세스 향상 계획 (e-Business Process Improvement Plan)	해당 비즈니스 문제가 이전에 다뤄진 적이 있는가? 비즈니스 템플리트, 공통 비즈니스 모델 등을 찾아본다.	
엔터프라이즈 컴포넌트 아키텍춰 (Enterprise Component Architecture)	전체 엔터프라이즈 아키텍춰의 일부로서, 정당한 비용으로 기존의 소프트웨어 자산[7]을 재사용 혹은 포괄하고, 프레임워크를 구축할 수 있는 기회가 있는가? 발견된 소프트웨어 자산이 아키텍춰에 적합한지를 고려한다. 여러 가지 아키텍춰 패턴을 고려한다. 의존성을 전검하고 전체 가능성을 조사한다.	
소프트웨어 요구사항 (Software Requirements)	해당 소프트에어 문제가 이전에 다뤄진 적이 있는가? 사용가능한 기존의 모델을 찾아보고, 그 문제해결에 도움이 될 수 있는 프레임 취크도 찾아본다. 그리고 그 후보들을 나열한다.	해당 문제가 보다 일반적인 용어를 기술될 수 있는가? 유사한 프로젝트와의 비교연구를 수행한다. 공통적인 면을 일반화 할 수 있는 기회를 찾아낸다. 분석작업이 다양한 컨택스트를 충분히 제공하는지를 확인한다.
프로젝트 컴포넌트 아키텍춰 (Project Component Architecture)	당면문제를 해결하기 위해 기존의 소프트웨어 자산 혹은 프레임워크가 사용될 수 있는가? 재사용될 자산이 아키텍춰에 적합한지 고려한다. 의존성을 점검하고 전체 가능성을 조사한다.	소프트웨어 자산을 일반화하거나 프레임워크를 구축할 수 있는 더 많은 기회가 있는가? 범용적인 컴포넌트 혹은 프레임워크가 아키텍춰에 적합한지 고려한다;아키텍춰 패턴을 고려한다. 의존성을 점검하고 전체 가능성을 조사한다.
행위 명세 (Behavior Specification)	해당 문제를 해결하기 위해 기존의 인터페이스가 사용될 수 있는가? 인터페이스를 확장하고 특화시킨다.	범용적인 인터페이스가 제공될 수 있는가? 인터페이스를 일반화시키고 분석 패턴을 고려한다.
구현 (Implementation (Internal Design & Acquisition))	해당문제를 해결하기 위해 기존의 구현이 사용될 수 있는가? 구현 설계를 재사용한다. 구현을 구매한다. 구현을 아웃소싱한다. 런타임 서비스를 이용한다.	변화를 관리할 수 있도록 설계가 충분히 유연성을 갖고 있는가? 유연성을 지원할 수 있는 구현을 설계, 구매 혹은 아웃소싱한다. 테스트계획이 다양한 컨택스트를 지원할 수 있는지 확인한다.

템의 또 다른 핵심은, 빠른 속도로 변화한다는 것이다. 예를 들어, 한 온라인 업체의 담당자는 매달 전자 중개 사이트를 새로운 버전으로 업그레이드하고 증가하는 접속량을 관리하기 위해 기초가 되는 인프라구조를 계속 발전시키고 있다. 그러므로, 프로세스 가이드의 필요성과 신속한 솔루션 제공 간의 균형이 잘 잡혀야 한다. 비록 불행하게도 종종 많은 제약을 가하는 경우가 있지만, 효과적인 프로세스는 과도하게 관료적인 방법으로 제약을 가하지 않고, 최상의 실전경험을 만들어낼 필요가 있다.

7) 소프트웨어 자산은 기존의 소프트웨어 시스템, 데이터베이스 혹은 소프트웨어 팩키지 등이 될 수 있다. 이들은 내부적으로 개발된 기존의 컴포넌트가 될 수도 있고 외부로부터 제공된 것들일 수도 있다.

훌륭한 프로세스는 도움이 되도록 만들어져야 하며 방해가 되어서는 안된다.

우리가 필요한 것은 "최적의(optimizing)" 프레임워크(Highsmith, 1999)라기 보다는 "적응력이 있는(adaptive) 조정 가능한" 프레임워크이다: "모델이 정해진 패턴에 포커스를 맞출 필요가 있는 프로세스라기 보다는 적응 가능한 프로세스여야 한다. 우리는 20세기의 "명령에 의해 컨트롤되는(Command-Control)" 모델에서 21세기의 "지도력과 협동이 어우러진(Leadership-Collaboration)" 모델로 옮겨가야 한다." 이 섹션에서 우리는 적응력이 있는 접근방법을 도와줄 수 있는 성공적인 소프트웨어 프로세스에 대한 몇몇 주제, 즉 반복적이고 점진적인 개발, 복합적인 개발 그리고 격차 분석(Gap Analysis) 등에 대해 살펴 볼 것이다. 이것은 여러 부류의 테크닉을 매핑하기 위한 컨텍스트를 제공한다.

4.1 반복적이고 점진적인 통합

반복적이며 점진적인 프로세스들은 객체 지향 개발 프로젝트의 특성이며 이에 대해서는 이미 책들에서(Kruchten, 1998; Jacobson et al., 1999) 잘 기술되어 있다. 산출물 또한 그림 6에서 볼 수 있는 바와 같이 반복적이고 점진적으로 발전하며 동시에 중요한 변환점이 있다.

e-Business를 위한 CBD는 단일 솔루션 중심도 아니고 개발(Development) 프로세스도 아니다. 이것은 계속되는 변화 관리(Change management)와 통합을(Integration) 위한 프로세스이다.

변화하는 비즈니스 요구사항과 조화를 이뤄가며 아키텍춰가 발전한다는 것을 포용할 수 있는 프로세스라야 한다. 우리는 단지, 비즈니스가 견딜 수 있는 만큼만의 아키텍처를 설계할 수 있다, 재사용을 위한 수확과 씨를 뿌리는 작업은, 이미 설명되었듯이, 요구사항부터 구현까지의 프로세스 전체를 통해 행해진다. 맞춤 설계부터 아웃소싱까지, 기존 시스템 통합부터 컴포넌트 구매까지, 프레임워크 확장부터 서비스 이용까지 다양한 구현 옵션들이 관련되어 진다.

e-Business 프로세스 향상 계획은 엔터프라이즈 아키텍처에 대한 스코핑 기준을 설정하고 소프트웨어 요구사항 프로젝트를 위한 비즈니스를 제공한다. 피드백은 e-Business 솔루션의 구현에서부터 정규의 절차대로 행해지며 평가되어 진다.

소프트웨어 요구사항, 명세 그리고 프로젝트 아키텍처의 정련작업은 근본적으로 반복적이며 안정된 시점에서 사실상의 스코핑을 하게 되며 더 상세하게 파고들기 위한 새로운 컨텍스트를 제공한다. 엔터프라이즈 아키텍처는 프로젝트 아키텍처를 지배하는 전체적인 컨텍스트를 제공하며, 프로젝트 아키텍처는 그 컨텍스트내에서 진화해간다.

소프트웨어 요구사항이 일단 안정되면 프로젝트는 점진적인 조립을 통한 구현으로 바로 옮겨갈 수도 있다. 이것은 종종 'fast-track'이라고도 알려져 있으며 RAD의 발전된 형태이다. 그렇지 않을 경우에는, 몇몇 요구사항의 부분 집합으로 나눠져서 다음 작업이 행해진다.

명세와 프로젝트 아키텍춰의 정련작업은 또 다시 반복적으로 행해진다. 일단 이 두 가지가 안정되면 프로젝트는 공급 전략에 따라 점진적인 구현으로 옮아간다.

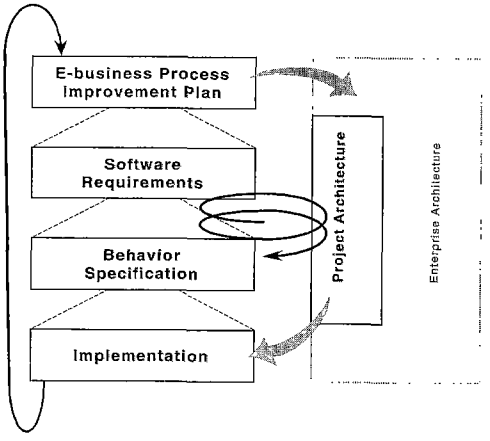


그림 6 CBD 액티비티의 반복 및 스코핑

4.2 복합적인 통합(Hybrid integration)

CBD의 주요 특징인 복합적인 통합을 프로세스가 어떻게 지원하고 있는지를 살펴보자. 예를 들어, 복잡한 프로젝트의 첫번째 반복작업에 의해, 그림

7에서처럼, 두 가지의 서브 프로젝트(P1과 P2)를 야기시키는 소프트웨어 요구사항문서가 생성되었다고 가정해 보자. P1은 이 중 하나의 요구사항을 해결하기 위해 fast-track assembly를 수행하는 RAD 프로젝트이다. P2는 아키텍쳐를 더 자세히 조사하고 컴포넌트를 명세화 하기 위한 명세 프로젝트이다. P2는 다음과 같은 5개의 컴포넌트로 구현될 프로젝트라고 하자[8].

1) P2.1은 또 하나의, 조립 프로젝트이다. 그러나 이번에는 사용자에게 바로 직면하는 그런 기능이라기보다는 프로세스 컴포넌트의 형태로, 인터페이스 집합을 제공한다.
2) P2.2는 레거시 통합 작업과 컴포넌트를 구현하기 위해 어댑터[9]를 만드는 작업을 포함한다.
3) P2.3은 작은 레거시 시스템에 대한 직접적인 랩퍼인다.
4) P2.4는 조직이 직접 구현하기로 결정한 컴포넌트의 내부설계 작업을 포함한다.
5) P2.5는 아웃소싱 프로젝트이다.

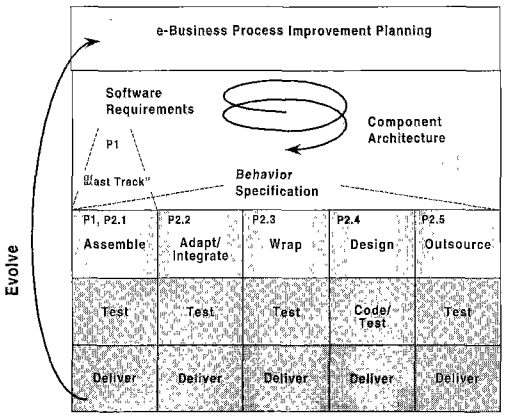


그림 7 CBD 프로세스내에서의 복합적인 구현

이렇게 해서 컴포넌트가 구현되면, 더 많은 인터페이스들이 추가될 수도 있고 혹은 기존 인터페이스가 확장될 수도 있다. 예를 들어, fast-track assembly는 컴포넌트 상태로 발전되거나 어댑터에 추가되는 더 많은 인터페이스로 발전될 수 있다. 이것은 CBD의 중요한 특징이다 : 이것이 바로 진화론적인 접근(evolutionary approach) 방식이다. 이와 같은 혼합된 구현방식은 오늘 날의 소프트웨어 개발에서 점차적으로 증가는 추세에 있다.

여기서 우리가 짚고 넘어가야 하는 것은, e-Business 시스템에 대한 CBD의 적용은 각각의 인터페이스를 분석과 설계의 단위로서 다룬다는 것이다. 이제, 인터페이스가 점진적이며 병행적인 개발에서 자연스러운 인도(delivery) 메커니즘을 제공한다는 것이 분명해졌다.

인터페이스는 또한 e-Business 솔루션을 발전시키기 위한 매우 편리한 수단이다. 초기 증분(increments)은 주로 솔루션에 대한 큰 그림만을 전달하기 위해 설계되는 다소 불완전하고 임시적인 인터페이스를 가질 수도 있다. 요구사항이 명확해짐에 따라 새로운 그리고 보다 완전하고 안정적인 인터페이스가 소개되어 진다. 클라이언트 시스템은 새로운 버전의 인터페이스가 준비되는 대로 이 인터페이스쪽으로 스위치될 수 있다. 더 이상 사용되지 않는 인터페이스는 단계적으로 제거될 수도 있다. 우리는 또한 인터페이스의 수가 증가함에 따라 형상 관리 이슈를 다루어야 한다.

4.3 격차 분석(Gap analysis)

CBD의 중요한 특징은 제시된 요구사항과 이미 존재하는 컴포넌트간의 차이를 평가하기 위한 격차 분석이라는 것이다. 격차 분석은 컴포넌트 공급 전략을 낳는다 : 이전 섹션에서 보았듯이, 서로 다른 구현 옵션들을 포함하도록 권장한다.

격차 분석은 제시된 요구사항을 재조정하거나 솔루션을 보다 저렴한 비용으로 신속하게 전달하기 위해 요구사항과 타협하는 작업을 포함할 수도 있다는 것을 이해하는 것이 중요하다. E-Business 향상 계획의 일부로서 팩키지의 사용이 필수적인 경우라 하더라도, 격차 분석은, 있을 수 있는 팩키지의 단점을 상위 수준에서 이해하고 특별히 현업 사람들의 기대치를 관리하기 위해서라면 여전히 중요한 작업이다. 팩키지는

8) 보다 전형적인 증분(increments)은 컴포넌트의 각 인터페이스에 대해 트리거 되어질 것이다.
9) 어댑터는 비컴포넌트 소프트웨어, 흔히 레거시 시스템을 호출하는, 그리고 받아들인 정보를 변형하여 그것의 인터페이스를 통해 새로운 기능을 제공하는 컴포넌트이다. 어댑터는 종종 새로운 컴포넌트와 레거시 소프트웨어 사이에 있는 컴포넌트 계층을 형성하며 레거시 마이거레이션을 용이하게 해준다.

비즈니스에 의해 이상적으로 요구되는 것들과는 대립하여, 비즈니스 컴포넌트의 통합을 위한 컴포넌트 아키텍쳐 구성을 결국에는 심각하게 제약할 수도 있다.

격차 분석은, 보통 잘못 이해되고 있듯이 명세 단계에서 단지 한 번 수행되는 것이 아니며, 프로세스 전반에 걸쳐 추상화 레벨을 감소시키면서 적용되어 진다. CBD는 소프트웨어의 전체 라이프 사이클에 영향을 미친다.

5. 테크닉과 산출물(Techniques versus deliverables)

인터페이스 기반 접근을 위한 기본 스텝으로 다시 돌아오자. 각 스텝은 반복적이고 점진적인 모습으로 적용되어지는 서로 다른 테크닉들에 의해 행해진다.

· 비즈니스 모델링(Business modeling) : 전략, 조직, 프로세스 그리고 정보에 관한 비즈니스 요구사항을 이해한다.

· 유즈 케이스 모델링(Use case modeling) : 소프트웨어 행위에 대한 비즈니스 컨텍스트를 식별하고 범위를 잡는다.

· 비즈니스 타입 모델링(Business type modeling) : 소프트웨어에 의해 관리될 비즈니스 개념을 식별하고 그 정의를 기술하며 인터페이스 후보들을 식별하는 것을 돕기 위해 발전되어진다.

· 상호작용 모델링(Interaction Modeling) : 요구된 행위를 실현하기 위해 소프트웨어 개념간의 동적인 상호작용을 점검한다. 소프트웨어에 의해 제시될 행위를 식별하고 나누며 인터페이스에 책임을 할당하고 그 인터페이스 후보를 발전시킨다.

· 아키텍쳐 모델링(Architecture modeling) : 소프트웨어 의존성을 이해하고 인터페이스의 책임을 조정한다. 기술적인 가능성도 검증한다. 인터페이스를 컴포넌트로 그룹화 하고 소프트웨어를 배치 가능한 단위로 나눈다.

· 명세 모델링(Specification modeling) : 행위와 정보에 관하여 인터페이스를 구체화 한다. 컴포넌트로 구현되어질 인터페이스 그룹들에 대한 명세 작업을 컴포넌트 명세 형태로 수행하며 제약사항도 포함시킨다. 목록에 그 컴포넌트 명세를 등록하고 공개한다.

· 내부설계 모델링(Internal design modeling) : 컴포넌트 구현을 직접 하거나 취득한다.

그림 8은 테크닉들간의 전체적인 관계를 그림으로 보여주고 그 테크닉들이 전형적인 프로젝트 산출물과 어떻게 관련을 맺고 있는지를 보여 준다. 내부설계 모델링은 범위에서 벗어나므로 이 그림에는 나타나지 않았다.

지금 부터는 그림 8과 관련하여, 테크닉들이 어떻게 함께 작업될 수 있는지에 대한 개요를 설명 하고자 한다.

비즈니스 모델은 비즈니스 타입과 유즈 케이스

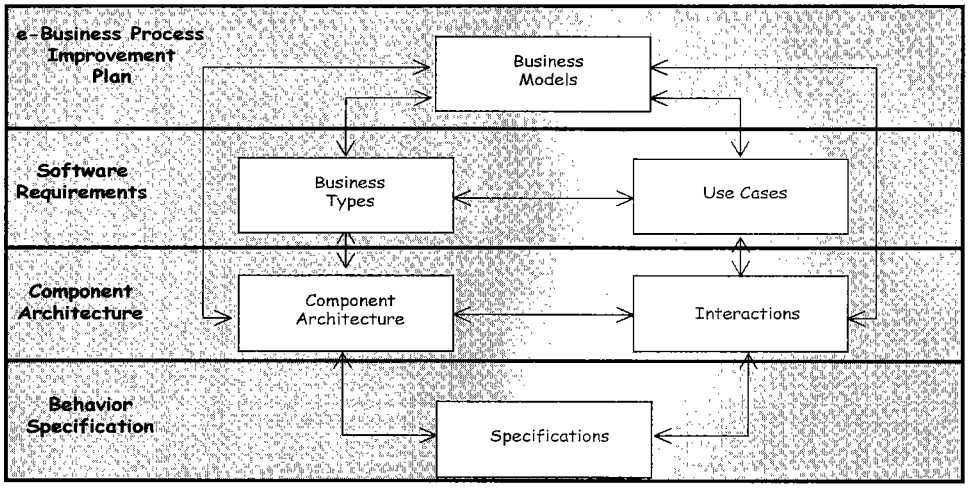


그림 8 전형적인 프로젝트 산출물에 관련되는 테크닉들

를 모델링하기 위한 컨텍스트를 제공한다. 비즈니스 프로세스 흐름 다이어그램은 유즈 케이스를 식별하는데 도움이 된다. 비즈니스 개념 모델은 비즈니스 타입을 식별해내기 위한 초기 작업이다. 비즈니스 모델은 컴포넌트 아키텍처의 범위를 잡는데 도움을 주고 상호작용 모델링에 대한 컨텍스트를 설정한다. 역으로, 그림 8에 나타나 있는 4가지의 소프트웨어 관련 테크닉, 즉 비즈니스 타입 모델링, 유즈 케이스 모델링, 컴포넌트 아키텍춰 모델링, 상호작용 모델링의 결과로서 나온 정보들은 비즈니스 모델을 개선할 수도 있다.

유즈 케이스와 비즈니스 타입 모델링사이는 양방향이다. 둘 다 소프트웨어 요구사항을 이해하는데 도움이 된다. 비즈니스 타입 모델은 유즈 케이스를 지원해야만 한다. 유즈 케이스는 새로운 비즈니스 타입, 속성, 연관관계 등을 발굴하는데 도움을 주어야 한다. 동시에, 비즈니스 타입 모델은, 비즈니스 정책, 비즈니스 룰, 정보 요구사항 등과 같은, 유즈 케이스에 의해 나타나지 않는 요구 사항도 반영하여야 한다.

비즈니스 타입 모델링은 아키텍춰 모델링을(그리고, 그 반대로도) 이끌어 내야 한다. 인터페이스는 초기에 비즈니스 타입 모델을 고려해서 정의되어진다. 하나의 인터페이스는 응집된 비즈니스 타입 집합을 관리하여야 한다. 인터페이스간의 의존성은 컴포넌트 아키텍처에 반영되어야 한다. 인터페이스는 상호작용 모델에서 정의되어지며, 이 상호작용 모델은 인터페이스에 대한 이해를 높이는데 도움이 된다.

유즈 케이스 모델링은 상호작용 모델링을 (그리고, 그 반대로도) 이끌어 내야 한다. 유즈 케이스는 하위 레벨의 유즈 케이스 스텝, 요구되는 타입 그리고 인터페이스들로 정련되어지며, 상호작용 모델에서 정의되어 진다(주의 : 사실상 유즈 케이스 모델링 없이 상호작용 모델링으로 바로 나아갈 수도 있다). 이런 방법으로 상호작용을 통해 생각하는 방식은 또한 유즈 케이스가 조정되게 하기도 한다.

상호작용 모델링은 아키텍처 모델링을 (그리고, 그 반대로도) 유도한다. 아키텍처는 상호작용 모델에 정의되어지는 인터페이스를 제공한다. 역으로, 상호작용(그리고, 유즈 케이스)은 아키텍처 상에서 움직이고 테스트되어지며, 이 작업 도중에 인터페이스의 역할이 조정되거나 의존성이 제기되거나 새로운 인터페이스와 그 의존성이 도출되어 진다.

상호작용 모델링은 명세 모델링을 (그리고, 그 반대로도) 유도한다. 상호작용을 지원하기 위해 필요한 오퍼레이션과 그 상호작용에 의해 필요한 속성과 타입들이 인터페이스 명세에 정의되어 진다. 상호작용 모델링은 또한, 컴포넌트 명세에 기록되는 제약사항과 의존성을 식별해내는데 도움이 된다. 역으로, 이런 방식으로 명세를 통해 생각하는 것은 또한 상호작용이 조정되도록 하기도 한다(예를 들면, 오퍼레이션이 다른 인터페이스로 옮겨지는 것 등이 될 수 있다).

아키텍처 모델링은 명세 모델링을 (그리고, 그 반대로도) 유도한다. 인터페이스 명세와 컴포넌트 명세는 아키텍처에서 식별된 의존성을 지원해야 한다. 역으로, 이런 방식으로 인터페이스를 통해서 생각하는 것은 아키텍처가 조정되도록 하기도 한다.

6. 결 론

이 글에서, 우리는 e-Business를 위한 효과적인 프로세스의 주요 특징들을 정리해 보았다. 이 글에서는 보다 포괄적으로 이 정보를 분류하고 구조화하려고 시도하지는 않았다. 그 이유는 이 글의 범위를 벗어나기 때문이기도 하고, e-Business를 위한 CBD가 매우 기술적이며, 창조적이며 상황에 맞게 적절하게 변화 가능한 프로세스이기 때문이기도 하다. 종종 이러한 사실을 간과하는 프로젝트를 위해 프로세스들이 만들어지기도 하고, 극도로 상세하게 또 엄격하게 개발 프로세스가 설명되어지기도 한다.

그럼에도 불구하고, CBD를 계획하고 다양한 컴포넌트 공급 루트와 스킬 셋을 관리하기 위해 어느 정도의 구조는 기본적으로 갖춰져야 한다. 프로세스 프레임워크와 산출물들은 재사용을 위해 우리가 보았던 체크리스트, 가이드라인 등과 함께 좋은 시작점을 제공한다. 산출물은 또한, 입력물과 출력물로서, 서로 다른 CBD 모델링 테크닉을 사용할 수 있는 컨텍스트를 제공하는데 도움을 준다.

참고문헌

[1] Allen, P. and Frost, S., Component-Based Development for Enterprise Systems: Applying The SELECT Perspective , Cambridge University Press-SIGS Publications, 1998.

[2] DSDM Consortium, DSDM Version 3 , Tesseract Publishing, 1997.

[3] Highsmith, J., Adaptive Management: Patterns for the e-Business Era , Cutter IT Journal, vol 12 no 9, Sept 1999.

[4] Jacobson, I., Booch, G., Rumbaugh, J., The Unified Software Development Process , Addison Wesley Longman, 1999.

[5] Kruchten, P., The Rational Unified Process: An Introduction , Addison Wesley Longman, 1998.

[6] Linthicum, D., Enterprise Application Integration , Addison Wesley Longman, 2000.

[7] Martin, J., Rapid Application Development , Macmillan, New York, 1991.

[8] Stapleton, J., DSDM-The Method in Practice , Addison Wesley Longman, 1997.

Paul Allen

Experience: 대규모 상용시스템에서의 프로젝트 매니저, 주요 통신회사의 방법론 고문, Yourdon, Inc사의 컨설턴트 매니저, Princeton Softech사의 방법론팀 리더, DSDM 타스크 그룹 의장

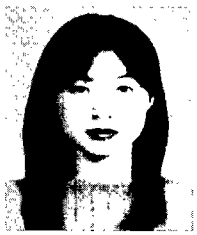
전문분야: Business IT alignment, 소프트웨어 프로세스, 컴포넌트 모델링 & 아키텍춰, e-Business로의 전환 관리 (e-Business transition management)

공저: Component-Based Development for Enterprise Systems Paul Allen and Stuart Frost, Cambridge University Press, Cambridge, UK, 1998

현재 Computer Associates사의 Principal Component Strategist

E-mail:paul.allen@ca.com

김 경 주



1990 경북대 컴퓨터공학과 학사

현재 쿨소프트웨어 코리아 CBD팀 차장

현재 연세대 산업대학원 전산학과 석사과정

관심분야: 소프트웨어 공학, 컴포넌트 기반 개발, 분산객체 컴퓨팅

E-mail:kyungju_kim@coolsoft.co.kr