특집

CBD(Component Based Development)현황과 전망

최 성[1] 윤 태 권[2]

목 차

1.CBD(Component Based Development)정의

1.1 CBD정의

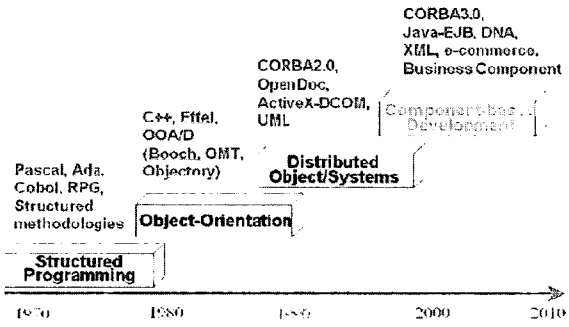
CBD란 재사용 가능한 소프트웨어 모듈 컴포넌트를 생성 및 조립 생산, 선택, 평가 및 통합으로 구성하여 더 큰 컴포넌트를 생성하거나 완성된 어플리케이션 소프트웨어를 구축하는 개발기법이다. 프로그래밍 위주의 전통적인 정보공학 개발방법론과는 달리 이미 만들어진 테스트 완료된 컴포넌트를 기본 아키텍처와 설계도에 따라 조립하는 방식의 새로운 개발형태로서, 컴포넌트의 오퍼레이션 및 상호 오퍼레이션을 정의하는 명세의 개발, 객체나 컴포넌트들로부터 컴포넌트의 구축, 컴포넌트들을 이용해 어플리케이션을 조립 등 세가지 활동을 필요로 한다. CBD는 대체로 UML과 같은 객체모델링언어와 EJB, COM 등과 같은 컴포넌트 플랫폼 기술을 사용해 구현하는 경우가 많지만 집을 지을 때처럼 특정 재료보다는 아키텍처 설계와 디자인이 중요하다. 소프트웨어를 레고 블럭처럼 만들어서 사용자가 원하는 모양으로 조립해서 사용한다. 소프트웨어 컴포넌트를 조립해 새로운 애플리케이션을 만들 수가 있으므로, 개발기간을 단축할 수 있고, 기존의 컴포넌트를 재사용함으로써 생산성과 경제성을 높일 수 있다. 컴포넌트를 만드는 데 소요되는 시간은 CBD 케이스 툴의 자동화 기능(워자드)을 사용하면 컴포넌트를 만드는 개발기간을 단축시킬 수 있다. 비즈니스 로직이 틀릴 경우에는 컴포넌트를 재조합하거나 새로운 컴포넌트를 끼워넣기만 하면 된다. 컴포넌트로 시스템을 만들 때의 장점은 개발하려는 시스템의 전체를 컴포넌트화 할 수 있는 여러 개의 단위로 분리해 개발하므로 복잡성을 단순화시킬 수 있고 초기 개발을 실질적으로 수행할 수 있도록 해준다. 컴포넌트는 캡슐화 되어 있어 로직상의 에러나 런타임 에러 등의 범위를 컴포넌트로 한정할 수 있어 유지보수가 용이하다.

1.2 CBD의 등장배경

80년대에 떠오른 소프트웨어 위기와 그 해결책으로 생각되었던 객체지향 개념은 소프트웨어 모듈의 독립성은 보장하지만, 재사용성에 대한 소프트웨어 모듈로서 수정이나 재컴파일 과정없이 클

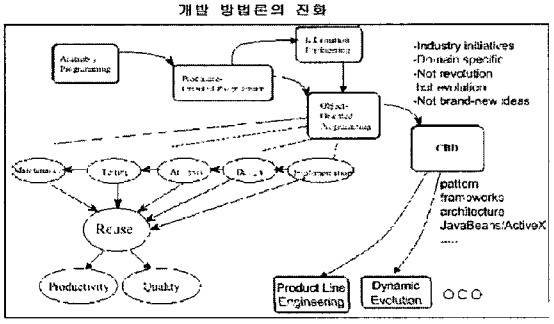
1) 남서울대학교 컴퓨터학과 교수
2) 한국소프트웨어 컴포넌트콘소시움 사무국장

라이언트 모듈 호출만으로 사용할 수 있도록 하지는 못한다. CBD는 구조적방법론->정보공학적방법론->패키지통합방법론, 객체지향 방법론, 클라이언트/서버 방법론 ->모델기반방법론->컴포넌트 기반 방법론으로 발전되었다. CBD는 새로운 컴포넌트 기술이 아니라 객체지향 개발이나 데이터 모델링, 분산 시스템 등을 포함한 전통적인 개발 접근법이나 기술들의 집합체다.



(그림 1) 소프트웨어 개발 패러다임

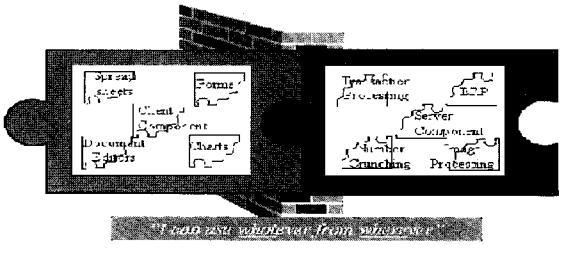
정보화 사회가 고도화 될수록 소프트웨어의 라이프 사이클이 단축됨에 따라 저비용에 의한 소프트웨어 개발기간의 단축이 절실하게 요구되고 있다. 그리고 인터넷의 보급으로 다양한 플랫폼 환경 하에 다양한 소프트웨어 획득이 가능해져 이들 이질적인 소프트웨어 컴포넌트들을 효과적으로 분류, 유통 및 조립하는 기술이 필요로 하게 되었으며, 또한 소프트웨어의 'Plug and Play' 기술의 발전으로 소프트웨어 부품 조립기술 등장과 이기종 컴퓨터간 연동 기술의 발전으로 인터넷 상의 다양한 소프트웨어 부품을 기종에 상관없이 이용할 수 있게 하였다. 컴포넌트 기반 개발은 인터넷 시대의 무한 경쟁에서 살아남기 위한 수단이라고 할 수 있다. 남보다 더 빠르게, 더 질이 좋은 소프트웨어를 만들어 시장을 선점하고 새로운 비즈니스 모델에 신속히 대처할 수 있는 소프트웨어를 만들기 위한 노력의 일환으로 컴포넌트 기반 개발에 대한 관심이 높아져 가고 있다. 그러므로 CBD는 어플리케이션 개발의 신속성, 용이성, 플랫폼 독립성 등에 의해 개발비용의 절감, 제품의 시장진입의 신속성, 생산성 향상 및 품질 향상을 기할 수 있다.



(그림 2) 소프트웨어 개발방법론의 진화

1.3 CBD의 능력

컴포넌트 기반 개발의 능력은 개발자로 하여금 기개발된(prefabricated) 'Plug-and-Play' 의 컴포넌트들을 '결합하고 조화(mixing and matching)' 시킴으로써 어플리케이션을 더 빠르고, 더좋고, 더 저렴하게(faster, better, cheaper) 개발 가능하게 한다. (그림 3)과 같이 컴포넌트웨어는 엔터프라이즈 어플리케이션(Enterprise Application)과 데스크톱 어플리케이션(Desktop Application)간의 통합의 장벽을 제거시킨다. 컴포넌트웨어는 운영체제, 네트워크 프로토콜, 언어, 개발도구, 인터페이스 스타일 등의 경계선에 걸쳐 접근 및 상호운영이 가능하다.



(그림 1) 엔터프라이즈 어플리케이션 및 데스크탑 어플리케이션간의 통합의 장벽을 제거

(그림 3) 통합2장벽 제거

I-Kinetics는 엔터프라이즈 어플리케이션 컴포넌트가 필요로 하는 6개의 기본 능력을 정의하였다.

1.3.1 환경 독립(Environment Independence)

컴포넌트는 언어, 운영체제, 네트워크, 개발 환경간에 상호 운영될 수 있어야 한다.

1.3.2 위치 투명성(Location Transparency)

사용자들은 동일 머신내에서 수행되는 하나 또는 복수의 프로세스들이나 네트워크/운영체제들간에 수행되는 복수의 프로세스들로부터 하나의 컴포넌트를 투명하게 액세스할 수 있어야 한다.

1.3.3 구현과 분리된 인터페이스(Interface Separate from Implementation)

인터페이스는 컴포넌트의 기능성을 명세하며, 어떤 구현, 플랫폼 및 환경 의존성을 드러내지 않는다.

1.3.4 자기 설명적인 인터페이스(Self-describing Interface)

컴포넌트는 인터페이스 명세(Interface Specification)를 설명하는 능력을 제공한다. 명세는 최소한 method 및 property 명세를 포함해야 한다. 이것은 컴포넌트 소프트웨어가 run-time시에 재사용을 위해 필수적인 능력을 결합할 수 있는 필수적인 기능이다.

1.3.5 플랫폼 바이너리 독립(Platform Binary Independence)

컴포넌트를 어느 플랫폼상에도 설치하고 사용할 수 있는 능력을 의미한다.

1.3.6 보편적인 응용컴포넌트 프레임워크 (Universal Application Component Framework)

컴포넌트는 새롭고 잘 정의된 어플리케이션을 생성하기 위해 하나 이상의 컴포넌트와 통합될 수 있어야 한다.

2. CBD의 국내외 개발현황

세계적으로 소프트웨어의 라이프사이클이 짧아지고 차세대 정보기술(IT) 패러다임인 웹서비스에 대한 관심은, e비즈니스 모델의 빠른 변화 · 발전에서 비롯되는 현상이다. 따라서 '보다 우수하고 안정적인 SW를 얼마나 빨리 개발하느냐'가 SW기업의 성패를 좌우하는 시대가 열렸다. 궁극적으로 SW 품질관리에 대한 열풍이 일고 있다. 국내도 사각지대였던 SW 품질관리에 대한 관심이 증폭되면서 CMM(Capability Maturity Model), SPICE(Software Process Improvement Capability determination)등 국제품질 인증 바람이 불고 있는데다 SW 납기단축, 비용절감을 위한 노력이 본격화되고 있다.

'왜 CBD(Component Based Development)인가.' 답은 국산 SW경쟁력 강화를 통한 수출증대의 지름길이기 때문이다. 특히 CBD는 다국적 IT 기업들이 운용체계(OS), 개발도구, 플랫폼, 애플리케이션 등 SW 전분야의 헤게모니를 장악한 상황에서 우리나라의 후발 IT기업들에 상대적으로 많은 기회를 제공할 것으로 보인다. 이는 컴포넌트가 SW시스템에서 독립적인 업무와 기능을 수행하는 모듈로서 '교체가 가능하다'는 특징에서 비롯된다. SW를 조립식으로 탈착 할 수 있기 때문에 후발 IT기업들도 자바(J2EE), 닷넷(.NET), CCM(CORBA Component Model) 등의 표준환경을 활용해 다양한 CBD SW를 개발해 시장에 선보일 수 있는 것이다. 또 CBD가 이미 개발해놓은 전산자원과 기술개발소스를 다시 사용할 수 있다는 점에서 국산 SW 개발시간과 비용을 크게 줄여놓을 것으로 보인다. 구체적으로

CBD는 건축용 블록으로 집을 만드는 것처럼 프로그램 논리(logic)를 독립적으로 구성, 컴퓨터 하드웨어구조(아키텍처)와 설계도에 따라 프로그램을 맞춰 나가는 SW개발방법론이다. 이를 통해 SW개발자들은 '모든 것을 만들어야 하는 개발방법론'에서 탈피해 기존에 개발된 SW모듈을 재사용해 쉽고 빠르게 시스템을 구축 · 관리 · 유지할 수 있게 된다. 이에 따라 지난 10년간 1만2000여개의 벤처기업이 탄생하고 70억달러 이상의 외자유치가 이루어지는 등 외형이 성장했음에도 불구하고 해외경쟁력이 취약한 국산 SW산업의 차세대 승부처로서 CBD가 빠르게 부상하고 있다.

정부도 CBD SW가 국내 SW 연간생산액인 83억달러의 3% 불과한 수출비중을 높이는 열쇠가 될 것으로 보고 지원에 나서고 있다. 실제 정통부는 오는 2007년까지 5년간 960억원을 CBD분야에 투입해 국내에서 진행되는 SW 및 시스템통합(SI) 프로젝트의 50%를 CBD체계로 구성한다는 목표를 세웠다.

또 공공기관의 정보시스템 구축작업에서 CBD SW의 사용을 의무화하는 방안이 추진되고 은행권을 중심으로 CBD 시스템 구축열기가 확산되는 등 시장활성화에 대한 기대가 높아지고 있다. 결국 CBD가 국내 SW산업의 경쟁력을 좌우할 '품질과 개발속도'라는 두 마리 토끼를 잡을 수 있는 SW개발방법론으로 빠르게 부상할 전망이다.

소프트웨어(SW) 부품화, 즉 컴포넌트기반개발(CBD)에 대한 노력은 SW산업의 고민으로부터 탄생했다. 하드웨어가 자동화된 대량의 생산시스템과 지속적인 연구개발투자에 힘입어 생산능력이 비약적으로 발전하고 있으나 SW는 아직 수작업 개발방식에 머무르고 있기 때문이다. 또한 SW는 날로 다양해지는 고객의 기능적 요구에 민첩하게 대응하기에도 어려운 상황이다. 이에 따라 부품화된 SW 컴포넌트가 대안으로 등장, 쉽고 편리하게 양질의 SW를 조립 · 생산하는 시대를 열고 있다. 특히 차세대 IT 패러다임인 웹서비스를 구현하기 위한 객체지향적인 CBD방법론이 대세로 자리잡고 있다.

2.1 해외 CBD동향

미국은 대통령 정보기술(IT)자문위원회를 통해 CBD(Component Based Development)를 선도개발기술과제로 선정, SW개발 및 유지보수 자동화를 위한 연구를 진행중이다.

유럽연합(EU)도 정보통신 RTD(Research, Technology and Development) 5차 기본계획에 CBD를 포함시켰으며 일본이 지난 97년 12월부터 정부기관 · IT업체 · 대학을 중심으로 CBOP(Consortium for Business Object Promotion)를 구성해 SW 재사용(CBD)을 위한 비즈니스 라이브러리를 구축하기 위해 힘을 쏟고 있다.

다국적 IT기업인 IBM은 아예 현금 21억달러를 투자해 CBD분야의 선두주자인 래쇼날소프트웨어를 전격 인수, 관련분야의 시장구조를 크게 바꿔놓을 것으로 예상된다. SW개발 자동화도구 전문업체인 래쇼날소프트웨어는 CBD의 모태인 객체지향엔지니어링 분야의 세계 1위 기업으로 IBM의 SW 모델링툴, 형상관리솔루션의 시장입지를 크게 넓혀놓을 것으로 보인다. 이로써 IBM(래쇼날소프트웨어), 오라클, 컴퓨터어소시에이츠, 투게더소프트, 텔레로직AB, 머랜트, 시레나(SERENA)소프트웨어 등 SW 모델링 도구 · 설계 · 형상관리 분야의 춘추전국시대가 도래할 전망이다. 이같은 시장경쟁구도에 힘입어 세계 SW 컴포넌트 시장규모는 지난해 55억달러, 올해 85억달러, 내년 130억달러, 2004년 200억달러 등 매년 52% 이상 성장할 것으로 예측되고 있다.

2.2 국내 CBD동향

금년 국내 SW 컴포넌트 시장규모가 1000억원

에 육박하면서 대중화의 토대를 마련할 것으로 분석된다. 특히 올해를 기점으로 국내에서 CBD에 관심을 갖거나 제품을 개발한 기업이 100개에 달하고 CBD 상용제품 수가 300종을 넘어설 것으로 전망되고 있다.

특히 은행들이 CBD 준거(레퍼런스)사이트로 등장하면서 기업 전반으로 고객이 확산되는 경향이다. 실제 국내 IT벤처기업인 모스텍이 한미은행 · 국민은행 · 하나은행의 국제금융시스템을 CBD로 완료한 데 이어 아토정보기술이 수자원공사 · 주택공사 · 강원랜드 등에 자바(J2EE) 기반 CBD시스템을 공급했다. 이네트, 화이트정보통신, 나모인터랙티브, 링크웨어 등도 CBD 솔루션을 잇따라 출시하고 삼성SDS, LGCNS, 현대정보기술, 포스데이타 등 대형 시스템통합(SI)기업들이 다양한 정보화 프로젝트를 CBD 방법론에 근거해 수행하는 등 국내 IT산업계의 SW 컴포넌트 바람몰이를 주도하고 있다. 또한 공공기관의 정보화 프로젝트에 CBD가 기본 시스템으로 채택되는 사례가 늘어나는데다 정부(정보통신부)가 CBD기술을 IT 전영역으로 확대하기 위해 내년부터 2007년까지 960억여원의 예산을 투입할 예정이어서 국내 IT산업계의 CBD 열기는 더욱 뜨거워질 전망이다. 이는 컴포넌트(component)가 플랫폼이나 운용체계(OS)에 따른 표준기술에서 공개 시스템 표준으로 확대되고 웹서비스를 정착시키는 데 크게 기여할 것이다. 컴포넌트기반개발(CBD:Component Based Development) 방법론을 활용한 소프트웨어 및 시스템이 정보기술(IT)산업계 전반으로 확산되고 있다. 또 세계 SW산업계에서 제품 자체, 시스템 프로세스에 대한 품질을 동시에 끌어올려야 하는 경쟁환경이 형성되면서 CBD가 'SW 부품화를 통한 속도전의 기재'로 떠올랐다. 이에 따라 국내외 IT기업들의 SW 컴포넌트 기술경쟁과 시장주도권 다툼이 더욱 뜨거워질 것으로 풀이된다. 특히 세계 7대 SW 강국을 꿈꾸는 우리나라로서는 올해 85억달러, 내년 132억달러, 2004년 202억달러 등 연평균 50% 이상의 성장을 거듭할 세계 SW 컴포넌트 시장에서의 성패가 관건으로 등장하고 있다.

2.3 다국적 IT기업의 CBD적용 공세

IBM, 썬마이크로시스템즈, 마이크로소프트 등 대형 IT기업들이 차세대 IT 패러다임인 웹서비스의 기술 · 시장 주도권을 차지하기 위한 방법으로 CBD를 적용한 프로그램을 속속 선보이고 있다. 썬마이크로시스템즈는 기존에 발표한 제품 · 기술 · 서비스를 CBD방법론을 통해 기업의 엔드 투 엔드(end-to-end)시스템에 적합한 형태의 시스템 구조로 묶어내는 'N1'를 제시했다. N1은 시스템의 구성요소를 가상화(virtualization)하고 자동화함으로써 전반적인 운영을 단순화해 경제동향에 걸맞은 최적의 시스템 환경을 제공한다.

IBM도 'e비즈니스 온 디맨드(on demand)전략'을 수립하고 컴퓨팅 · 확장성표기언어(XML) · 리눅스 · 그리드 · 웹서비스 등의 요소기술들을 통합(CBD)한 시스템 운영환경을 제공한다. 마이크로소프트의 경우에도 자사의 객체지향 방법론인 컴포넌트디자인원칙(Principles of Component Design)을 포함하는 MSF(Microsoft Solutions Framework)를 내세우고 있다. 관련기업들은 각자의 CBD방법론을 한국의 전산환경에 걸맞도록 변형 · 표준화하고 위험관리(Risk Management)기능을 포함시켜 시장을 선점해나갈 계획이다.

2.4 국내 IT기업들의 CBD도전

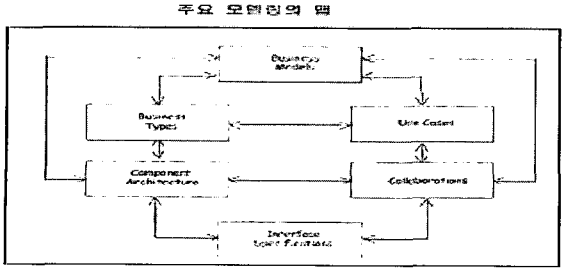
정보통신부가 지난해부터 SW품질인증제를 시행하고 CMM(Capability Maturity Model), SPICE(Software Process Improvement Capability determination) 등 국제 SW품질인증의 중요성이 부각되자 국내 IT기업들도 SW품질과 생산성(개발속도)을 향상시키기 위해 CBD

기술개발에 팔을 걷고 나섰다.

특히 은행 · 보험 · 증권 · 통신 · 건설업을 중심으로 정보화 프로젝트를 컴포넌트 기반으로 수행하는 기업이 늘어나면서 SW 컴포넌트 개발업체들의 입지가 넓어지고 있다. 이는 개방성과 확장성을 겸비한 CBD를 기본 시스템으로 채택함으로써 시스템 사용영역을 웹으로 확장하고 기존 정보자산을 재활용할 수 있다는 인식이 확산된 결과다. 실제 한미은행 · 국민은행 · 하나은행이 국제금융 외신시스템을 구축하면서 국산 SW 컴포넌트를 채택했으며 SK텔레콤 · 수출입은행 · 신한은행 · 기업은행 · 서울은행 · 삼성물산 · 현대중공업 · 롯데제과 등도 CBD 수요처로 떠오르고 있다. 이에 따라 300여종의 국산 SW 컴포넌트가 등장하고 완성된 CBD제품을 제3자에게 판매하는 유통(채널)사업도 고개를 들기 시작하는 등 국내 CBD시장은 올해 1,000억원을 돌파한 후 연평균 130%씩 고속성장해 오는 2004년 4,600억원대를 형성할 것으로 예상된다.

3. CBD의 주요 모델링

e-Business 시스템 개발을 CBD로 할 때 가장 핵심적인 모델링 개념을 설명한다. 이 내용은 Paul Allen의 컴포넌트 모델링을 중심으로 설명한다. 그리고 컴포넌트 기반 접근 방법에 대한 자세한 내용은 다루지 않고 개념, 표기법, 정의 정도만 언급한다.



(그림 4) Modeling Map

이 모델링을 기술하기 위해서는 객체지향의 UML(Unified Modeling Language)의 분석 및 설계 모델링에 대한 표기법과 의미를 이용한다. UML 의 표현 기법을 기본으로 하되 지원하지 못하는 부분 특히 비즈니스 모델링에 대한 표기법은 Penker와 Eriksson이 제안한 것을 이용하여 설명하려고 한다. 그리고 UML Diagram을 이용하여 표기하기 보다는 "Thinking Tool"로서 유즈케이스 정제 작업과 인터페이스 반응에 대한 맵을 사용해야 한다. UML에서 말하는 다이어그램(Diagram)보다는 맵(Map)으로 이용하는 것이 중요하다. 여기에서는 객체지향 기법을 다루는 것이 아니므로 우리는 비즈니스 분석, 아키텍처, 그리고 사양에 초점을 두고 설명하려고 한다.

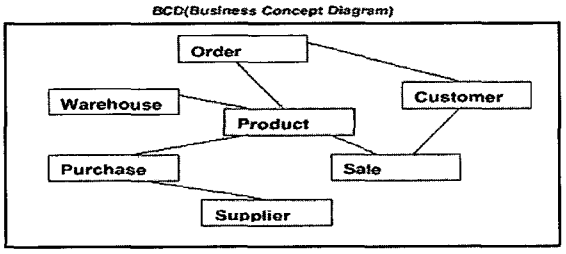
3.1 Business Modeling

비즈니스의 전제조건으로서 룰(규칙), 목적, 문제로 나눌 수 있다. 룰은 비즈니스 관점에서의 비즈니스 정의와 제약 사항이다. 그리고 비즈니스 관점으로 지시적인 목적 및 문제로 나눌 수 있다. 비즈니스의 전제조건들 즉, 룰, 목적, 문제에 대해 가장 중요한 것은 입증 가능성의 유무다.

- 비즈니스 룰에 대한 표현은 개념적인 제약 사항 혹은 조건으로 표현할 수 있다.
  (Payment OK = Order OK + Money OK, Hierarchy 다이어그램으로 표현)
- 비즈니스 목적 (예 : 2003년 1월 수주량은 70% 증가시켜 꼭 달성해야 한다)과 문제 (판매는 현재 70% 다운될 것으로 예상)는 비즈니스 프로세스와 연관되어 표현된다. 이는 프로세스 흐름 다이어그램 (PFD : Process Flow Diagram)으로 표현된다. 비즈니스 프로세스란 고객의 가치를 높일 수 있는 비즈니스 능력에 관련된 조직이다. 이 능력은 태스크와 같은 친숙한 의미로 실체화 될 수 있고, 비즈니스 프로세스에 따른 실행부서 (그룹)들간의 협업으로

표현된다.

- 비즈니스 능력은 하나 혹은 많은 서비스들이 제공되어 지고, 그 결과로 제공되는 접점지점에서 필요한 S/W 를 요구하게 된다. 비즈니스 프로세스 실행 부서 (High-Level)의 비즈니스 프로세스는 세분화된 비즈니스 프로세스 (Low-Level)의 항목 분석을 필요로 한다. 비즈니스 프로세스를 분류하면 고객 프로세스, 유지 프로세스, 기능 프로세스 세 부분으로 나누어 볼 수 있다.



(그림 5) Business Modeling의 BCD

3.2 Business Type Modeling

비즈니스 타입 모델링은 비즈니스 개념 모델링을 찾기 위한 것 보다는 BCD에 대한 기술과 내부적인 관계에 집중되어진다. 도메인 레벨의 정보 요구 사항들은 실행단계와는 다르게 독립적으로 설계되어져야 한다. 일반적으로 BTM (Business Type Modeling)은 컴포넌트 아키텍처 기반을 설계하는 데 사용된다.

타입에 대한 정의는 기술적으로 구조적 기술에 대한 내용을 설명한다. 각 타입에 대한 객체 (인스턴스)를 구현하기 위한 방법을 정의하는 것은 아니다. 타입은 일반적으로 UML 기법의 데이터 구조(속성)와 행동(수행)을 구조적으로 표시하고 분류하여 정의한다. 그리고 분류된 객체(인스턴스)들은 각각 Identity을 가지고 있다. 다시 말해서 각각의 타입은 데이터 구조(속성)와 오퍼레이션으로 정의할 수 있다. 그러나 어떻게 오퍼레이션할 지에 대한 메소드는 정의하지 않는다. 타입들은 종속, 상속, 연결 형태의 관계성을 가지고 있으며, UML 의 스테레오 타입의 클래스로 표현하면 된다.

타입은 매우 클래스와 비슷하다고 말할 수 있다. 그러나 클래스는 C++, Java 같은 프로그램 언어로 특별히 구현 가능한 것을 가정하는 것이고, 타입은 기술적으로 뚜렷하지 않은 형태라고 말할 수 있다. 이와는 반대로 타입의 오퍼레이션은 클래스가 가지고 있는 메소드들을 정의한다. 타입의 가장 중요한 형태의 특징은 인터페이스를 기본으로 한 접근방법과 비즈니스 수용에 따른 구현 가능 여부가 초점이다.

타입은 일반적으로 클래스와 같은 형태로 실체화된다. 이것은 객체지향 언어의 클래스형태다. 만약에 관계형 데이터베이스와 같은 언어를 사용한다면 긍극적으로 같은 내용이 될 수도 있다. '비즈니스 타입' 이란 항목은 비즈니스 정책에 반영된 가장 일반적인 타입을 강조한 것이다. 예를 들면 구매처 혹은 공급자 같은 것이 될 것이다. 그리고 비즈니스 타입은 기술적인 형태의 추상화 (데이터베이스 서버, 프로토콜 등)는 아니다. 다시 말해서 비즈니스 타입은 일반적인 항목들은 사용치 않는다.

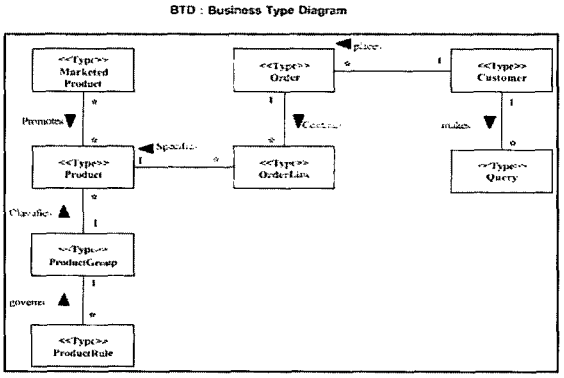
〈표 1〉 타입 표기법

타입별 표기법

〈〈Type〉〉 Type Name
Attribute Name 1 --- Attribute Name 2
Operation Name 1 --- Operation Name 2

주의할 점은 속성과 오퍼레이션들이 옵션이라는 사실이다. 그리고 가능하면 타입 다이어그램에서

는 은폐된 형태의 속성과 오퍼레이션들이다.
BTD(Business Type Diagram)는 비즈니스 타입과 관계를 나타내주게 된다. (그림 6)은 속성과 오퍼레이션이 은폐된 형태의 그림이다.



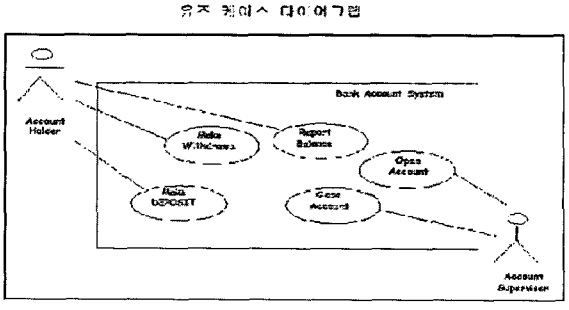
(그림 6) Business Type Modeling 중의 BTD

3.3 Use Case Modeling

유즈 케이스 모델링은 비즈니스 요구사항을 검증하고, 비즈니스의 조직 및 소프트웨어 개발 범위를 이해하기 위해 사용되는 것으로서 소프트웨어가 무엇을 할 수 있는지에 대해 외부적인 것에 초점을 두고 기술되어지며, 소프트웨어의 내부적인 메커니즘에 들어가기 전에 사용된다. 이것은 CBD에서 매우 중요한데, 비즈니스 서비스와 인터페이스에 대한 하나의 출발점으로서 사용된다. 유즈 케이스는 테스트 단계의 테스트 케이스로서 사용되고, 또한 프로젝트 수행시 소프트웨어 증가에 대응할 수 있는 메커니즘을 제공한다. 유즈 케이스는 주로 소프트웨어 요구사항을 획득하는데 주요한 수단이 된다.
유즈 케이스 모델링의 주요 특징은 다음과 같다.
- 시스템 사용자, 업무 영역의 전문가와 시스템 개발자 간의 의사 소통 수단 제공으로 요구사항에 대한 명확한 이해가 가능하다.
- "누가 (Who)" 시스템을 사용할 것인가?(시스템과의 상호작용)
- 시스템은 사용자를 위해 "무엇 (What)"을 해야 하는가?
- 사용자와의 상호작용을 위해 시스템이 제공해야 할 "인터페이스 (Interface)"는 무엇인가?
- 사용자의 모든 요구사항이 제대로 반영되었는지 검증 (Verification)하는 기회를 제공한다. 유즈 케이스는 사용자와 시스템간의 전형적 인터렉션으로 User-visible Function 들의 집합으로서 일종의 시나리오들의 집합으로 표현된다. 그리고 하나의 유즈 케이스는 액터들의 서브집합으로부터 형성된다. 다시 말하면 유즈 케이스는 액터가 해당 시스템을 통해 원하는 결과를 얻어내기까지 시스템 내부에서 이루어지는 이벤트들의 흐름이라 할 수 있다.

유즈 케이스의 주요 특징은 다음과 같다.
- 항상 시스템이 제공하는 특정 기능과 연관된 액터에 의해 시작
- 액터와 시스템 간의 상호작용을 표현
- Use Case들의 총합은 시스템 사용 시 가능한 모든 경우를 표현한 것임
- 단순한 기능의 열거가 아닌 이벤트들의 흐름
- 시작과 끝이 명확하며 액터에게 결과물을 제공해야 함

액터는 소프트웨어 시스템에 대한 사용자의 역할로 정의 할 수 있다. 액터는 사람, 그룹, 조직 유닛, 다른 하나의 소프트웨어 시스템, 장비가 될 수 있다.



(그림 7) Use Case Modeling의 다이어그램

3.4 Component Architecture Modeling

컴포넌트 아키텍처 모델링은 컴포넌트와 인터페이스의 범위, 의존성, 그리고 비컴포넌트의 소프트웨어 유닛과 연관된 의존성 등을 정의하고 조사하게 된다. 그리고 컴포넌트 아키텍처에 대한 패턴 요인, 설계 가이드 라인, 비기능적인 요구사항들, 아키텍처 정책을 포함하여 모델링한다.

3.4.1 컴포넌트 아키텍처 모델링 개념

컴포넌트 아키텍처 다이어그램은 소프트웨어의 유닛과 유닛들간의 의존성을 수립하는 것이다. 컴포넌트 아키텍처 모델링은 다음과 같이 세 가지 관점에서 모델링한다.

3.4.1.1 Specification Architecture

- Interface : 하나의 컴포넌트가 지원하는 인터페이스에 의해 제공되어지거나 요구되어지는 소프트웨어 서비스들의 집합인 스펙이다. 이 스펙에서 인터페이스 타입 모델의 구조와 제약사항에 대한 정보는 오퍼레이션에 의해 관리된다. 이 오퍼레이션은 인터페이스 타입모델과 파라미터에 의해 정의된다.
- Component Specification : 컴포넌트 스펙은 컴포넌트가 실체화된 것이다. 소프트웨어의 유닛에 대한 스펙으로 객체들의 집합에 대한 행동을 기술하고, 구현 유닛을 정의한다. 이러한 행동은 인터페이스의 집합으로 정의한다.

3.4.1.2 Implementation Architecture

- Component : 컴포넌트 스펙의 논리적인 실체화로서 독립적인 실행이 가능하다. 이것은 컴포넌트가 다른 독립적인 컴포넌트에 의해 교체와 인스톨을 할 수 있다는 것을 의미한다.
- Component Implementation : 컴포넌트에 대한 실행으로서 소스 코드, 데이터베이스 정의, 스크립트, 파라미터 등에 의해 구성된다. 보통 이것은 컴포넌트 배포 시 제공되지 않는다.
- Component Module : 하나 이상의 컴포넌트들의 물리적인 패키징의 부분으로 구성되는 파일이다. 컴포넌트 모듈은 하나 이상의 컴포넌트들로 패키지화 될 수 있다.

3.4.1.3 Deployment Architecture

- Installed Module : 일종의 인스톨된 컴포넌트 모듈로서, 런타임 실행환경에서 스펙 위치 및 레지스터에 대한 정보를 가지고 있다. 이들은 컴포넌트가 서비스하는 것에 의해 확인 할 수 있다. 하나의 컴포넌트는 컴포넌트 모듈에 의해서 인스톨된다.
- Installed Component : 복사된 컴포넌트다. 이 컴포넌트는 실행환경에서 배포된 것이며, 인스톨된 컴포넌트의 인스턴스 혹은 오퍼레이션을 실행함으로서 인스톨된 모듈에서 확인 할 수 있다.
- Component Object : 인스톨된 컴포넌트의 인스턴스다. 각 오브젝트 혹은 한 개의 데이터 값으로 확인 할 수 있는 집합으로 구성된다. 이들은 한 개의 컴포넌트를 상징한다.

3.5 Collaboration Modeling

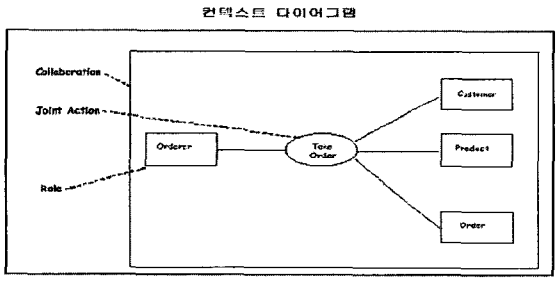
협업 모델링은 추상화 단계를 통해 타입들의 행동, 역할을 이해하고 확인하는데 도움을 준다. 이 모델링은 비즈니스 프로세스 모델링으로부터 구현 모델링까지 추상화 하는 데 이용한다. 협업 모델링할 때는 타당한 인터페이스들을 통한 행동(behavior)요소와 컴포넌트들 사이의 협업 타입들의 그룹이 기본적인 고려 사항이다.

Collaboration Modeling 개념

- Collaboration : 타입 오브젝트들의 역할을 수행하는 액션들의 집합으로 추상화 되어있다 .
- Action: 행동(behavior)들에 대한 분리된 조각이다 .
- Joint Action: 타입에 할당되어 있지 않은 인터페이스를 가지는 액션들로 유즈 케이스가 대표

적이다.

- Localized Action: "interface operation"라고도 하며 이 액션들은 인터페이스 형태로 타입에 할당되어 있다
- Role: 유즈 케이스 혹은 협업의 컨텍스트에 대한 책임들을 가지는 집합으로 타입의 한 표현이다.
- Refinement: 추상화 모델 분석과 정제된 상세 모델 생성의 액티비티다. 이것은 추상화에 대한 확인이고, 추상화에 대한 반대 개념이다.
- Abstraction: 정제된 상세 모델에서 덜 추상화된 상세 모델들로 구성된다.



(그림 8) Collaboration Modeling중의 Context Diagram

3.6 Interface Specification Modeling

이 모델링은 인터페이스를 자세하게 표기한다. 여기에서는 인터페이스 타입 모델을 작성하는 것이 주 목적이다. 즉 타입들의 콜렉션, 속성과 관계, 오퍼레이션 등으로 구성한다. 인터페이스 스펙 다이어그램은 인터페이스에 의해서 제공되는 오퍼레이션을 반드시 기술해야 한다. 각 오퍼레이션의 조건에 따른 Precondition, Postcondition을 표현해야 인터페이스, 컴포넌트 스펙에 적용할 Invariant를 적용해야 한다.

4. CBD의 전망

Gartner Group Report에 의하여 다음과 같은 미래의 컴포넌트 시장 구성이 예측되었다.

"소프트웨어 컴포넌트는 3개의 새로운 마켓 즉, 컴포넌트 마켓(Component Market), 컴포넌트 도구 마켓(Component Tool Market) 및 컴포넌트를 이용해서 개발되는 어플리케이션을 위한 마켓(a Market for Custom Applications Developed using Components)의 출현을 가져올 것이다."

위에서 언급된 3개의 마켓 중에서 첫째와 둘째는, 제품 마켓(Product Market)이고, 단지 셋째만이 서비스 마켓(Service Market)이다. 그러나 서비스 마켓은 제품 마켓보다 더욱 중요하게 될 것이기 때문에 다양한 서비스 마켓들이 출현할 것이다. 소프트웨어 컴포넌트 자체만으로 시장성이 약하기 때문에 서비스 마켓과의 결합을 통한 마케팅 전략이 요구된다.

4.1 컴포넌트 마켓 전망

여러 도메인에 걸친 경쟁력은 강하지 않기 때문에 수평적(horizontal) 또는 수직적(vertical) 도메인에 근거를 둔 컴포넌트 마켓이 생길 것이다. 소프트웨어 컴포넌트는 너무 'soft' 하여 Warehouse와 Distributor Chain과 직접적인 유사성을 갖지 않는다.

4.2 컴포넌트 도구 마켓

소프트웨어 컴포넌트 개발 및 사용은 전통적인 소프트웨어 개발보다 개발자에게 더 많은 것을 요구하고 있다. 그러므로 컴포넌트 기술(component technology)은 지원 도구에 관한 기대의 수준을 높여 상당수의 신규 도구들의 출현을 가져올 것이다.

4.2.1 컴포넌트 설계와 구현 도구(Component Design and Implementation Tools)

컴포넌트 설계는 컴포넌트 프레임워크(component framework)와 관련되는 컴포넌트

모델의 환경적 명세(environmental specifications)에 의존한다. 컴포넌트 기반 개발은 컴포넌트 시스템의 구축시에 요구사항을 빠르게 획득하기 위해서 RAD(Rapid Application Development) 방법을 사용해야 한다. 또한 컴포넌트의 프로토타입을 개발하고 구현하기 위해서 이러한 RAD 도구가 사용되어야 한다.

4.2.2 컴포넌트 테스팅 도구(Component Testing Tools)

컴포넌트의 테스팅은 컴포넌트 기술을 가장 필요로 하는 부분이다. 어려운 컴포넌트 테스팅의 보완을 위해 2 종류의 전략이 사용될 수 있다. 첫째의 전략은, 가능하면 에러를 피하는 것이다. 예를 들면, 안전하고 표현력 있는 랭귀지는 컴파일러나 분석 도구로 하여금 실질적인 에러를 발견하도록 한다. 좋은 경우로서, 주의 깊게 고안된 랭귀지는 대부분의 에러들을 제거할 수 있다. 탁월한 랭귀지는 메모리 deallocation의 명확한 개념을 갖지 않아 dangling references나 memory leaks가 간단히 제거된다. 더 좋은 예는 캡슐화와 invariants를 위반하는 부작용(side effects)을 제거해 주는 랭귀지 강화된 액세스 모드(Language-enforced Access Modes)와 가시성 규칙(Visibility Rules)이다. 두 번째 전략은, 컴포넌트 내의 오류가 logging 되도록 하여 컴포넌트 시스템 내의 오작동이 추적될 수 있도록 한다.

4.2.3 컴포넌트 조립 도구(Component Assembly Tools)

컴포넌트들은 컴포넌트 인스턴스를 연결하고 카스터마이징에 의해 조립된다. 모든 컴포넌트 인스턴스들은 조립시에 강력한 구축도구(builder tool)에 의한 visual한 표현을 사용함이 바람직하다. JavaBeans는 조립시간과 run-time사이를 명확히 구분하며, 컴포넌트 인스턴스들이 조립시간과 run-time 중에 다르게 보이고 행하게 해주는 컴포넌트 표준이다. 현재의 구축도구(builder tools)들이 간과한 중요한 측면은 조립 자체를 위한 자동화의 기능이다. 소프트웨어 조립은 각개의 부분들을 반복해서 조립할 필요가 없고 전체 조립한 제품을 증식할 수가 있다는 점에서 하드웨어 조립과 다르다. 제품의 구성 컴포넌트가 변경되거나 컴포넌트 버전이 변경 시 필요한 부분만 수정을 하면 된다.

4.2.4 컴포넌트 시스템 진단과 유지보수(Component System Diagnosis and Maintenance)

컴포넌트 테스팅과 관련하여 전체 컴포넌트 시스템이 진단될 수 있어야 한다. 시스템은 많고 다양하고 독립적인 벤더의 컴포넌트로 구성되기 때문에 진단은 복잡하다. 여러 벤더의 컴포넌트를 위한 진단 도구(diagnosis tool)와 진단 표준(diagnosis standard)이 만들어져야 하며, 컴포넌트 벤더는 진단도구에 통합될 수 있는 진단 컴포넌트를 만들어야 한다. 진단 후, 유지보수를 위해 운영 중인 시스템의 컴포넌트들, 그들의 인스턴스(instance)와 자원을 대체하는 것이 필요한데, CORBA System Management Common Facility가 그 예가 될 수 있다.

4.3 컴포넌트 서비스(Component Services)

4.3.1 컴포넌트 시스템과 프레임워크 아키텍트(Component System and Framework Architects)

컴포넌트 시스템이나 컴포넌트 프레임워크을 architecting하는 업무는 지극히 필요하기 때문에 독립적인 아키텍처 회사가 전문성을 가지고 서비스를 지원하게 될 것이다. 컴포넌트 시스템 아키텍트는 클라이언트에게 컨설팅을 제공하는 반

면, 컴포넌트 프레임워크 아키텍트는 open market에 기여할 것이다.

4.3.2 컴포넌트 조립 컨설턴트(Component Assembly Consultants)

소규모 시스템을 위한 컴포넌트 조립은 단순할지라도 필요한 컴포넌트의 수가 많은 대규모 시스템일수록 대규모 컴포넌트 저장소로부터 어느 컴포넌트를 선택할지는 어려운 문제다. 손쉬운 컴포넌트 조립의 솔루션을 제공하는 업체는 넓은 마켓을 확보할 것이다.

4.3.3 컴포넌트 형상관리(Component Configuration Management)

컴포넌트 저장소 내의 컴포넌트는 시간이 지남에 따라 진화되기 때문에 컴포넌트에 대한 버전 및 변경관리를 필요로 한다. 특히, 컴포넌트의 여러 버전이 여러 어플리케이션을 위해 사용될 때 효율적인 관리를 필요로 한다. 제품 계열(product line)의 개념을 기반으로 해서 시스템 버전을 쉽고, 신속하게 개발 및 릴리이스하기 위해서 형상관리가 중요한 역할을 한다. 컴포넌트 도구 마켓도 형상관리 지원을 필요로 한다.

4.3.4 컴포넌트 웨어하우스(Warehouse), 마켓팅 및 컨설팅

컴포넌트는 전통적으로 컴포넌트 제공자(생산자)와 사용자 사이에 조정을 필요로 하므로 컴포넌트 웨어하우스, 유통업자(Distributor), 마케팅 및 컨설팅회사가 모두 필요하다. 이러한 컴포넌트 관련 회사들은 인터넷을 통해 해당 서비스를 신속하게 제공할 수 있다.

5. 결 론

CBD기술을 보급함은 SW공학의 기본원칙을 따르는 것이다. 이 원칙은 SW생산성, 품질향상, 위험통제, 원가절감을 위해 반드시 필요한 것이다. 그러나 IT 기술이 가능케 하는 경영혁신 Model을 한국 산업의 현실에 맞도록 개발하는 것, 또한 해외 시장에 수출할 수 있는 SW 제품 및 부품을 기획할 수 있는 역량을 마련하는 것이 신기술 투자에 앞서야 될 선결 과제다. CBD는 이러한 새로운 Business Model의 개발, 기존 Business Process의 Reengineering 등 SW 설계 이전의 전략적 경영 기획을 강조하고 지원하는 방법임을 알아야 한다.

IT 기술의 변화 속도가 빨라짐에 따라 IT 기술에 대한 투자는 3년이면 그 효과가 감소하기 시작하는 데 반해 SW Process 및 SW Architecture는 기업 내에 축적되는 지식, 제도 및 문화로서 장기적인 핵심 역량으로 남는다. 부존자원이 빈약한 우리나라에서 SW산업은 국가경제의 전략산업일 수 밖에 없다. SW산업은 대표적인 지식집약형 산업으로 우수한 두뇌가 필요한 자원이다. 타 산업에서와 마찬가지로 SW 산업에서도 제조(Programming)능력보다는 설계(Architecture) 능력이 고부가가치 자원이며, 이는 무엇보다도 체계적인 교육의 강화를 통해 확보될 수 있다. CBD는 Architecture중심의 SW 구축 방법으로서 보급이 시급한 실정이다.

SW를 가르치는 분들의 학생 · 개발자들에게 세계수준의 SW공법, Architecture, 또는 Application Domain전문가로 육성함으로써 SW개발의 견고한 지식을 경쟁무기로 해외수출시장에 진출하겠다는 의지를 공고히 할 때, 우리 SW산업이 비약적 발전으로 세계에서 존경받는 진정한 IT강국이 될 것이다.

참고문헌

[1] 정보통신부, "S/W컴포넌트 기술개발계획", 1999.7.

[2] 정보통신부, 한국전자통신연구원, 한국소프트웨어컴포넌트컨소시움, "S/W컴포넌트활성화 2001년도 사업추진계획", 2001.6.

[3] 최성, "Together로 UML 배우기" 1권, 으뜸정보기술㈜간, 2002.6.,

[4] 최성, "Together로 객체지향 프로그램 배우기" 2권, 으뜸정보기술㈜간, 2002.9.

[5] Butlergroup, " Component-Based Development", Butlergroup Management Guides, Sept., 1998.

[6] Keith Short," Component-Based Development and Object Modeling", Sterling Software, Feb., 1997.

[7] Alan, W. Brown and Balbir Barn, " Enterprise-Scale CBD Building Complex Computer System From Components"' Sterling Software,1999

[8] Siegel, "CORBA: Fundamentals and Programming", John Wiley Press,1997

[10] Desmond F. D'Souza, Alan C. Wills: "Objects, Components, And Frame-Works with UML- the Catalysis Approach", Addison-Wesley, Reading, Mass.1998.

[11] D. Chappell, "On COM ActiveX vs. Java Beans" , Object Magazine. January 1998.

[12] Paul Allen, "Realizing e-Business with Components", Addison-Wesley, 2000

[13] Alan W. Brown, " large-Scale Component Based Development", Prentice Hall.2000

[14] George T. Heineman & William T Council, Component-Based Software Engineering", Addison-Wesley, 2001

[15] Peter Herzum & Oliver Sims, "Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise", John Willey & Sons, Inc, 1999

[16] John Cheesman & John Daniels, "UML Component : A simple Process for specifying Component-Based software", Addison-Wesley, 2000

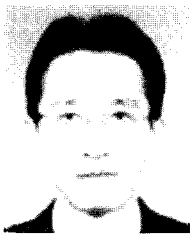
[17] Rosenblum, D.S. and Natarajan, R.," Supporting architectural concerns in component-interoperability standards" , IEE Proceedings-Software, Volume: 147 Issue: 6, pp.215 -223, Dec. 2000

[18] Rational Software Corporation, Rational Unified Process, http://www.rational.com/products/rup/index.jsp

[19] Computer Associates, CBD96, http://www.sterling.com/ cbdedge1/cbd96.htm

[20] CompuWare Corporation, UNIFACE Development Methodology, http://www.compuware.com/

저자약력



윤 태 권

1980년 성균관대학교 경상대학 졸업(경영학사)
1983년 1월 전국경제인연합회 입사 (공채)
1983년 5월 한국정보산업연합회 설립추진단 근무 및 이적
1997년 한국정보산업연합회 조사부장
1997년~1998년 정보통신부 SW유통협의회 위원
1998년~1999년 정보통신부 SW육성계획위원회 위원
1999년 정보통신부 SW경쟁력 강화 대책반 참여
1999년- 현재컴포넌트컨소시엄 운영위원회 간사
SW컴포넌트표준화포럼 운영위원회 간사
1999년 한국시스템통합연구조합 사무국장
2003년- 현재 한국소프트웨어 컴포넌트콘소시엄 사무국장

최 성

1975년~1994년 : 기업은행 전산개발부, 제주은행전산실장,
한국생산성본부 OA추진사무국장
1994년- 현재 남서울대학교 컴퓨터학과 교수
현) 한국정보기술전문가협회이사. 현정포럼이사,
정보화지도자포럼부회장
1999년 강원대학교 컴퓨터과학과 이학박사
1983년 연세대학교 산업대학원 전자계산학과 공학석사
관심분야 : 웹서비스, EA, EC/ERP, VR영상게임, 소프트웨어공학, 가상대학, CBD
이 메 일 : sstar@nsu.ac.kr