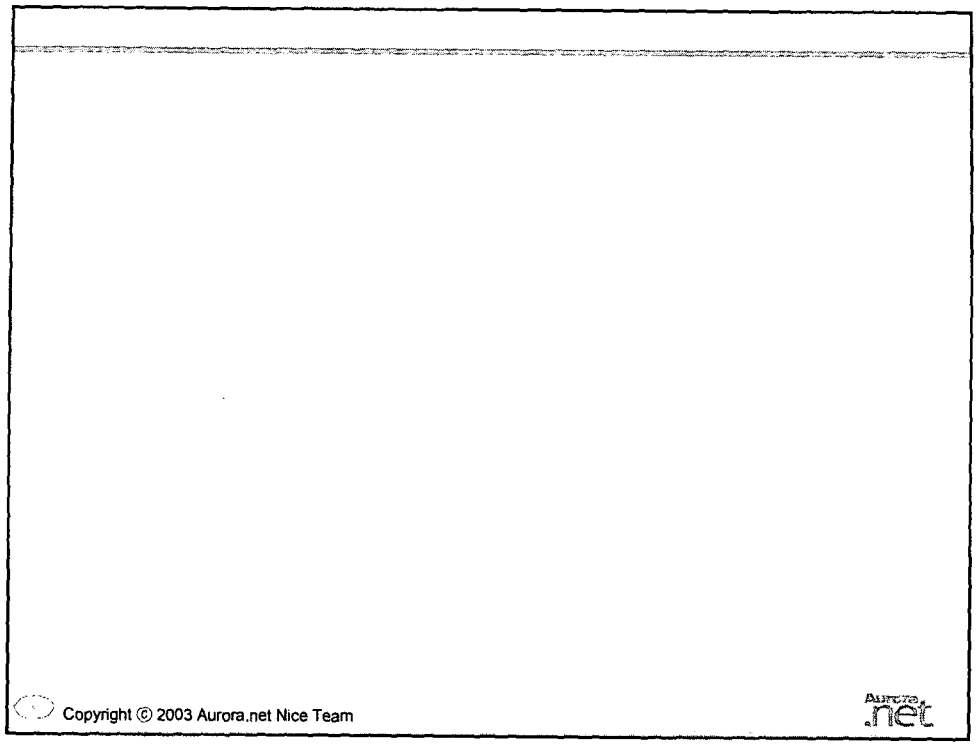


목 차

I Chapter CBD 기술동향 및 닷넷개발방법

1.CBD 시장 및 기술동향 3
2.MSF(Microsoft Solution Framework) Overview 8

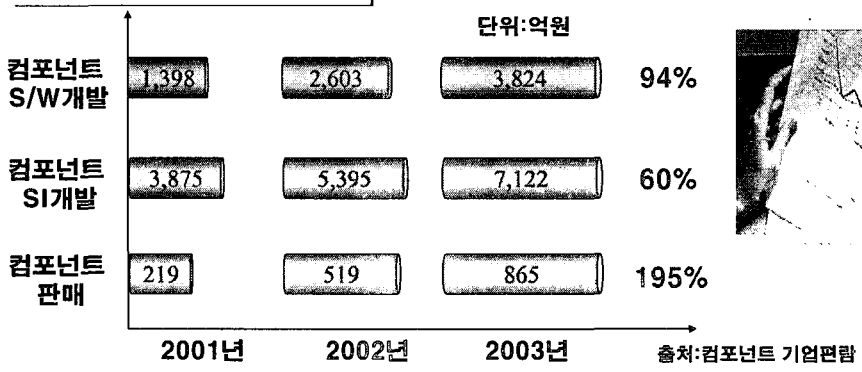
II Chapter MSF/CD 개발 방법론

1.Traditional C/S & CBD 16
2.Understanding MSF/CD 29

III Chapter Aurora.net 제품소개

1.Aurora.net 제품 소개 55
2.Aurora.net 제품 Demo 65

CBD 시장동향 – 평균 성장률



국내 76%의 소프트웨어 업체가 컴포넌트 기술도입 및 검토예정
국내 컴포넌트 시장은 연평균 142% 증가
CBD 개발을 위한 엔지니어 양성이 시급(.net 기반 C# 인력)

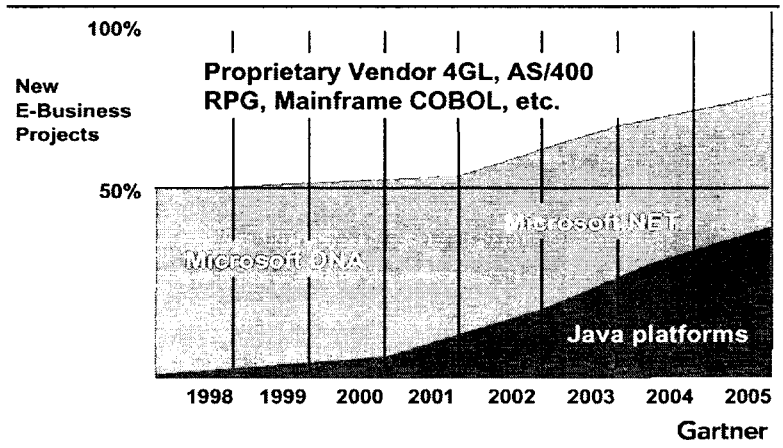
CBD 기술동향

사상	Build	Buy	Buy vs. Build	Portfolio Assembly
개발 방법론	구조적 방법론	정보공학 방법론	객체지향 방법론	컴포넌트기반 방법론
연대	• 1960~1980 년대	• 1980~1990 년대	• 1990~2000 년대	• 1990~현재
주요모형	• 프로세스(기능)	• 데이터	• 객체	• 컴포넌트
주요기술 환 경	• 메인 프레임 환경 • 단위업무 전산화 • 소프트웨어 공학 태동	• 전사규모의 정보시스템 • 클라이언트/서버 • 사용자 중심개발 • 통합 CASE	• 인터넷 이용확산 • 다양한 S/W요구증대 • UML • 분산 객체기술 • Round-Trip CASE	• 비즈니스 기능단위 컴포넌트 • .NET 기반기술 • C# 언어 • 웹서비스 기술 • 기존 시스템 재활용
언어 / DB	• COBOL,Fortran,C • H-DB, N-DB, R-DB	• Visual Basic, • Power Builder, Delphi • H-DB, N-DB, R-DB	• C++, C#, Java • R-DB	• C#, COM+ • R-DB
특 징	• 학습용이 • 보편적 활용	• 안정된 개발방법론	• 실세계 개념 모형 • UML 사용확산	• 적기개발 • 개발 비용 감소

Internet Web service

Ⅰ. CBD 시장 및 기술 동향

CBD 기술동향



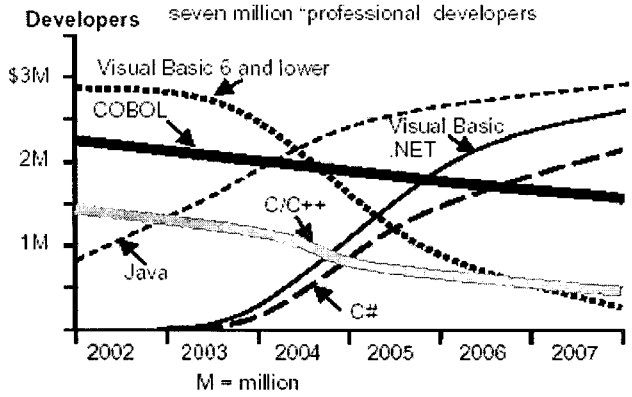
Source: Gartner Research

CBD 기술동향

Figure 3

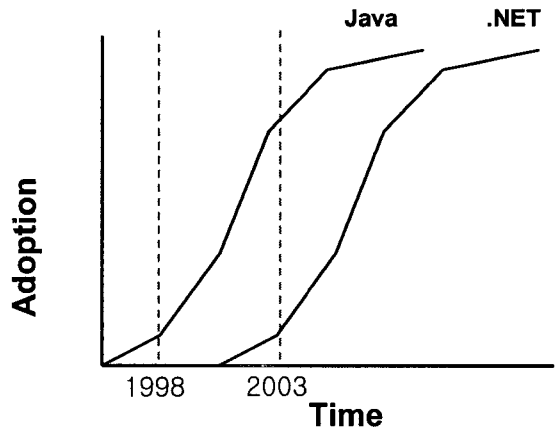
Number of Professional Developers Worldwide

From an estimated worldwide market size of seven million "professional developers



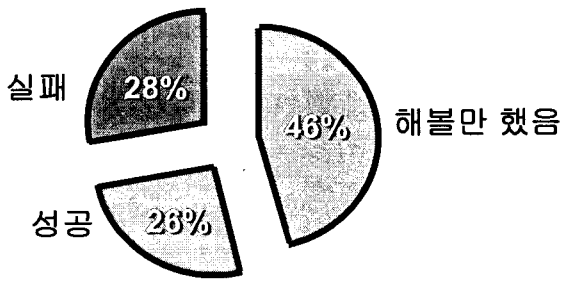
Ⅰ. CBD 시장 및 기술 동향

CBD 기술동향



II. MSF Overview

◆ 프로젝트 실패율



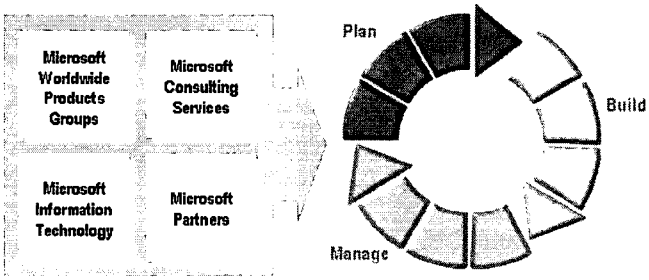
Application Development Projects

◆ 실패의 근본 원인

- ❑ 목표와 기능(function)의 분리
- ❑ 비즈니스와 기술(technology)의 분리
- ❑ 공통된 언어와 프로세스의 결여
- ❑ 한 팀으로서 의사소통하고 행동하지 못함
- ❑ 변화에 유연하게 적응하지 못하는 프로세스들

성공적인 솔루션을 제공하는데 있어, 사람의 측면을 더 깊게 살펴보면, 우리는 솔루션이, 문제를 겪는 사람들과 문제를 해결하려는 사람들 간의 인공적인 벽 때문에 어려움을 겪는 다는 사실을 보게 된다.

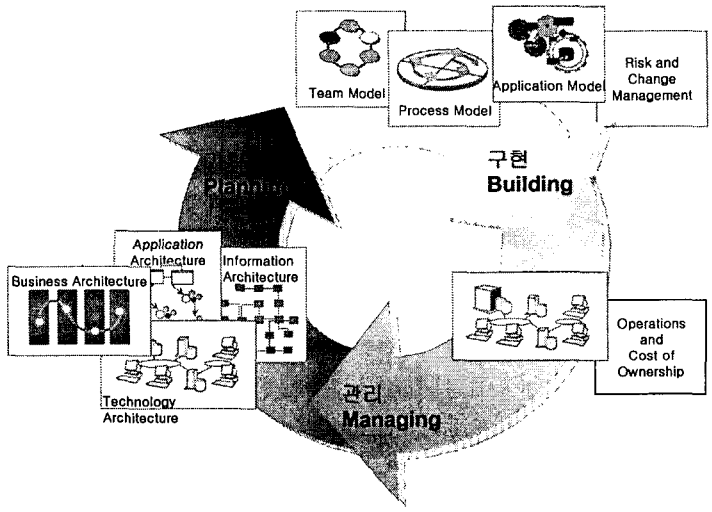
◆ MSF 탄생 배경



MSF는 마이크로소프트의 제품 개발과 IT 조직 내에서 최고의 실천 사례(best practice)에 기초를 두고 1994년에 만들어졌으며, 마이크로소프트 컨설팅 서비스(MCS) 조직 내에서 일관성과 효율을 촉진하는 표준화된 교육으로 개발되었다.

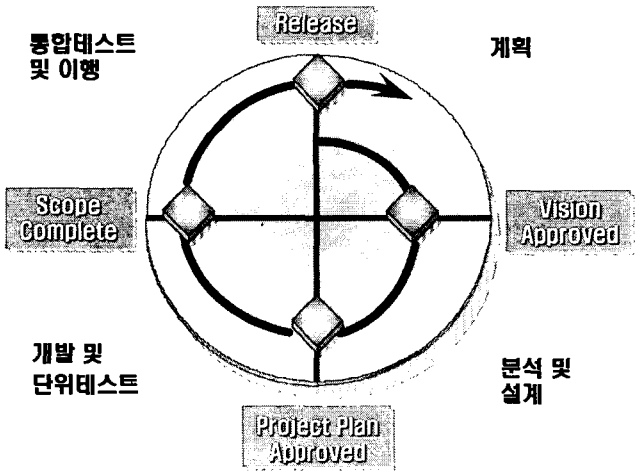
II. MSF Overview

◆ IT Life Cycle



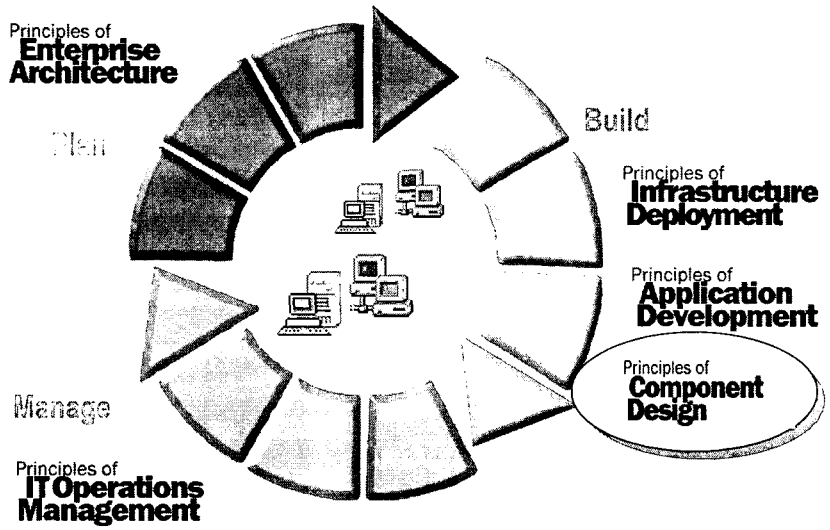
II. MSF Overview

◆ Project Life Cycle



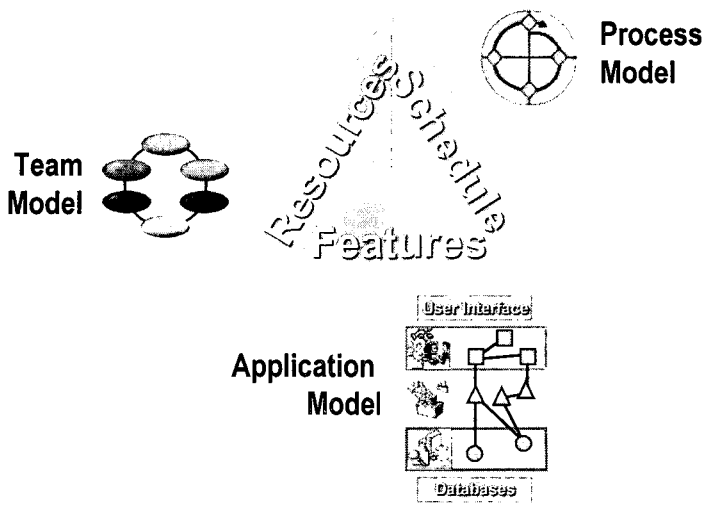
II. MSF Overview

Microsoft Solutions Framework

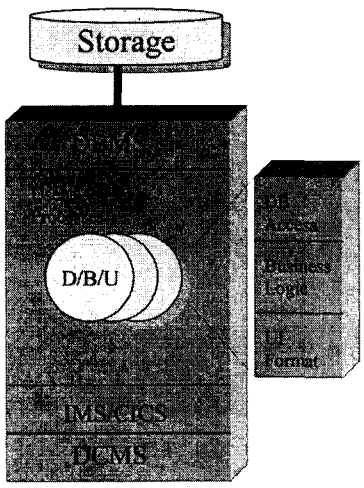


II. MSF Overview

Making Project Trade-Offs



Application Architecture Review : 1 Tier

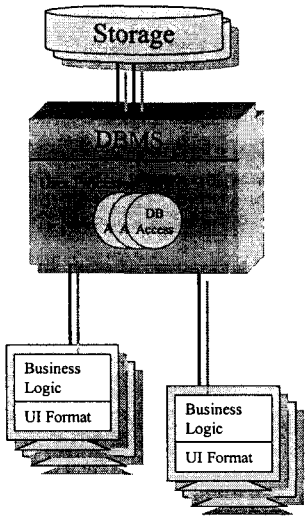


- Mainframe :
  중앙 집중 처리 방식
- Risk의 집중
  -> 시스템 운영조직 필요 : SP & OP
  -> 별도의 DBA 조직 필요 : RDB
- 이 기종간 Interface의 한계
- 설치 및 운영비용의 과다
- 이동 Client를 위한 대책 부재

-> 서버에 집중된 부하 및 위험요소의 분산 필요

I . Traditional C/S & CBD

Application Architecture Review : 2 Tier

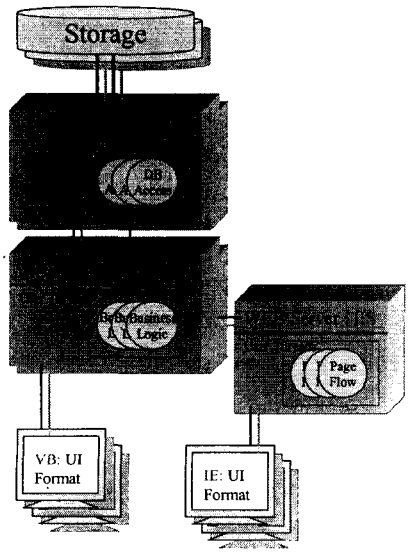


- 전형적인 Client/Server
- 부하 및 Risk의 분산
  -> 시스템 운영조직 필요 : SP
  -> 별도의 DBA 조직 필요 : RDB
- Client유형별 별도의 System 개발 필요
  : PB, VB, WEB
- 각 Client의 H/W, S/W 요구사항 증가
  : Memory, Disk, CPU & Softwares
- 각 Client 의 관리 요구 증대
  : Install, Upgrade, Configure
- 동시 사용자 확장의 한계
  : Trusted Multi-Users

-> B/L의 공유 및 최적의 Client관리를 통한 TCO의 절감 필요

I . Traditional C/S & CBD

Application Architecture Review : 3 Tier



- 확장 Client/Server
- 동일 Business Component (Logic)를 통한 다양한 유형의 Client 지원
  -> Application Logic과 Presentation의 분리 : '일 자체' 와 '결과의 표현' 분리
  (Object Modeling에 의한 설계 필요)
- 각 Client요구사항에 맞는 사양의 최적화
- 각 Client 의 관리 요구의 최소화 :
  IE/WinForm
- 다수 사용자 확장을 위한 Application Framework
  -> Larger Number of Concurrent User Support
- 일관된 기술구조를 통한 효율의 극대화

-> 효율적인 자원관리를 통한 TCO(Total Cost of Ownership)의 극대화

What's Object Design In 3 Tier Application System?

- Imagine you establish a company !
  - You know what your company is for
  - You know what kinds of work should be done
- You will find to do your business
  - what kind of people are required
  - what services they will have
  - what other service tools should be needed to support your people's work
- You will recruit people and buy tools with the minimum cost

  & Start your business

What's Object Design for 3 Tier Application System?

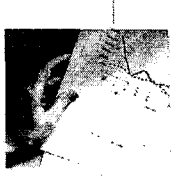
- Company = System
  People, Tool = Component
  People/Tool's Service/Function = Component's Method
  Office = Server
- How the work is made against the customer?
  Thru the service combination of one or several people
  =
  Thru the method combination of one or several components
- In 2 tier system, the work itself, the process 'From Start To End' is the program, so each process require each corresponding program
  , which does not look like the real life

Difference of Modeling Approach / Modeling Sequence

- Traditional C/S
  1. Data Modeling
     Process Modeling using Decomposition
  2. Grafting Data Model on Process Model
  3. UI design, Physical database design
  4. Program Design depending on UI & DB
  5. Development
- CBD
  1. Process Modeling using Use case/Scenario
  2. Service Modeling : Logical object model
     Database design, UI design
  3. Physical Component Modeling
  4. Design Cooperating Interface Mechanism
     between Component & UI, Component & DB
  5. Development

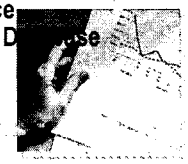
Difference of Modeling Approach / Process Modeling

- Traditional C/S
  1. Process Analysis by Top_down Decomposition
  2. Process Relation Analysis by Event Dependency
- CBD
  1. Key Business Process Analysis Based on
     Business Context
  2. Process Mutual Interaction Analysis
     by Event and Corresponding Actor's Activity



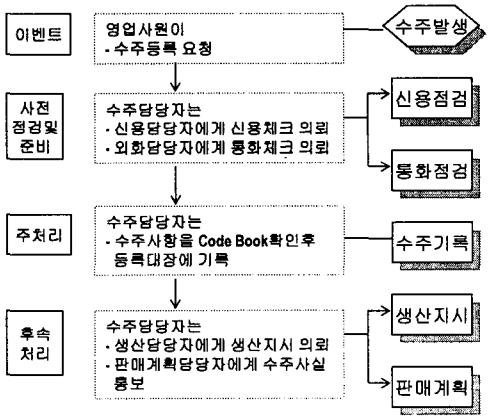
Difference of Modeling Approach / Characteristics

- Traditional C/S
  1. Business Process Analysis from System designer's perspective
  2. Data Model Centric Design
  3. Logical process Analysis, and Based on current events Program Design
     -> Dependency of Program to UI and Database is relatively high
- CBD
  1. Business Process Analysis from User perspective
  2. Service Model Centric Design
  3. By UI & Database design according to Service, Dependency of Business component to UI & D[illegible]se is relatively low



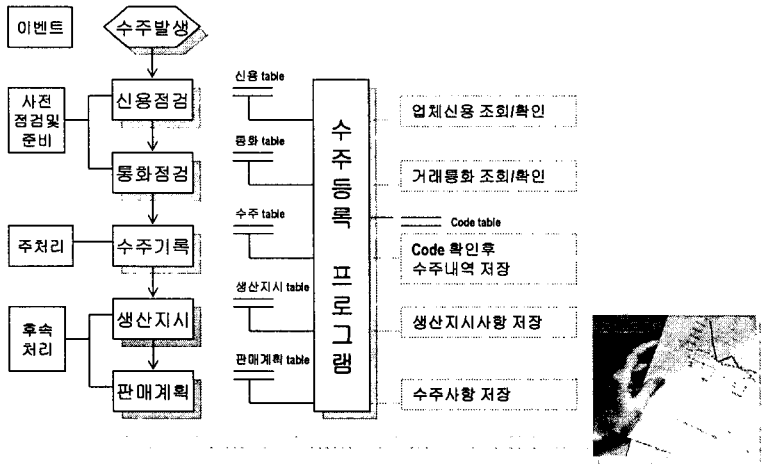
Traditional CS vs CBD : 실제 업무처리 Flow

Supposing a scenario : 수주등록



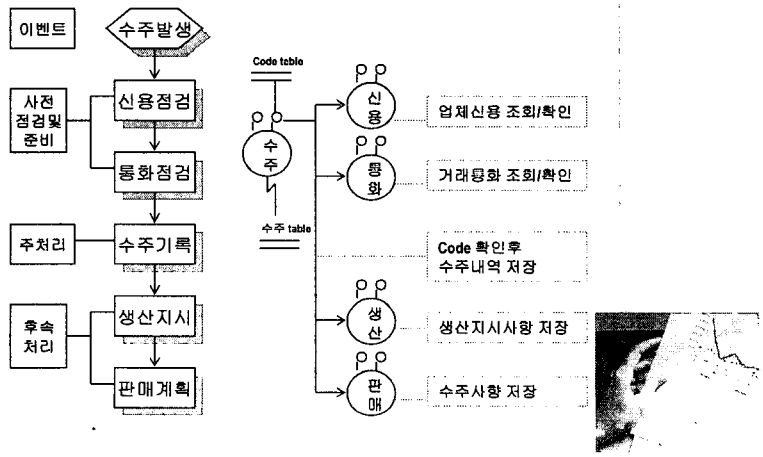
Traditional CS vs CBD : Traditional C/S Application

Supposing a scenario : 수주등록

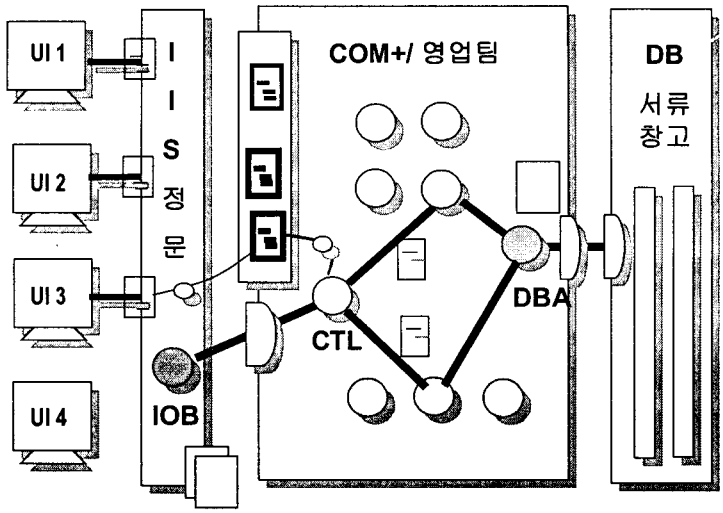


Traditional CS vs CBD : CBD Application

Supposing a scenario : 수주등록

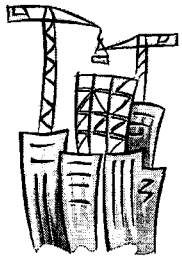


Architectural View of 3 Tier System



소프트웨어 개발과정

- 요구사항 수집(*Requirements gathering*)
- 분석(*Analysis*)
- 설계(*Design*)
- 구현(*Construction*)
- 검사(*Testing*)
- 배포(*Deployment*)
- 유지보수(*Maintenance*)



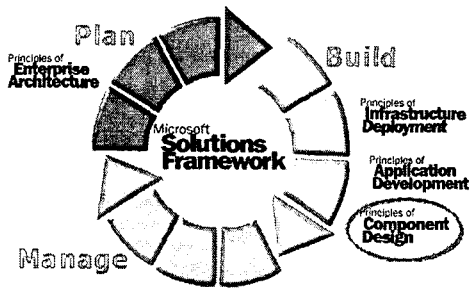
전통적인 개발방법은 ...

- 구조적 또는 정보공학 분석/설계 기법을 따르는
- 시간에 맞춰 끝나기는 하지만 급변하는 비즈니스 환경 덕에 재작업이 필요하게 되는
- 개발보다 유지보수에 더 많은 자원이 소요되는
- 업무가 변경되면 프로그램을 변경해야 하고, 자료처리 방식도 변경해야 하는
- 업무 프로세스보다는 데이터 중심으로 설계하는
- 개인의 경험이나 능력(Skill)에 의존하는
- 분석가 또는 프로그래머 중심으로 프로젝트를 진행하는
- 팀 프로젝트의 경우 팀간의 정보교류 부족으로 통합의 문제가 발생하는
- 재사용성을 강조하지 않는
- 설계산출물의 피드백을 허용하기는 하지만 이전 단계의 잘못된 작업으로 인하여 다음단계에 영향을 미치는

경향이 있다

Ⅱ. MSF/CD 개발 방법론

MSF/CD 개발방법론



Aurora.net은 MSF/CD (Microsoft Solution Framework / Component Design) 개발 방법론을 적용하여 시스템을 개발 시 소프트웨어를 분석 / 설계하는데 사용하는 비주얼 모델링 도구라 정의할 수 있다. 기본적으로 모델/뷰(Model/View)구조에 근거하여 소프트웨어 시스템을 설계하고 프로그램코드를 자동으로 생성해 가는 반복 개발(Iterative Process)의 지원을 위한 제반 환경을 제공한다.

- MSF의 개발 방법론 종류

MSF/CD (Component Design, 컴포넌트 설계 방법론) -> Aurora.net

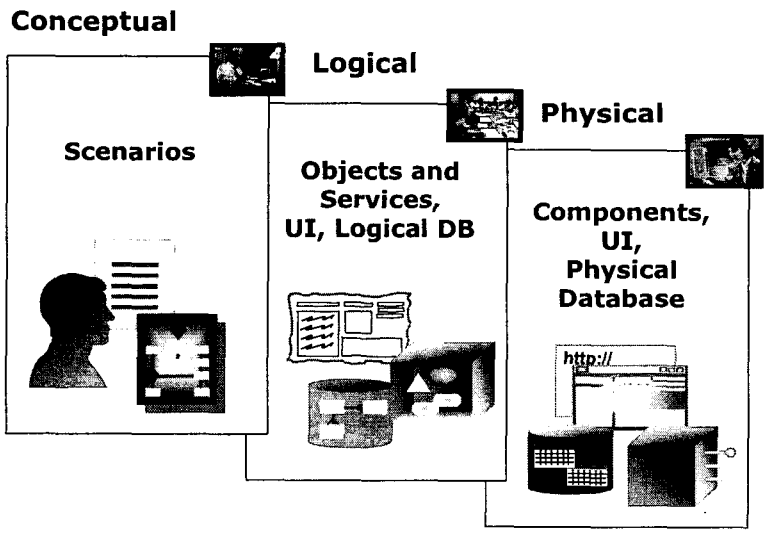
MSF/AD (Application Development 응용 프로그램 시스템 개발 방법론)

MSF/ID(Infrastructure Deployment 시스템 기반 기술 구축 방법론)

MSF/EA (Enterprise Architecture 기업 전략 기반 시스템 아키텍쳐 계획 수립 방법론)

Ⅱ. MSF/CD 개발 방법론

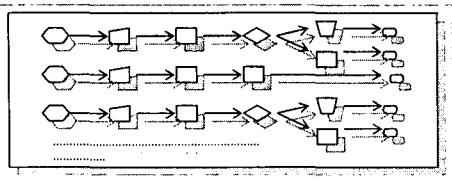
Design 3 단계



Apply Design Concept

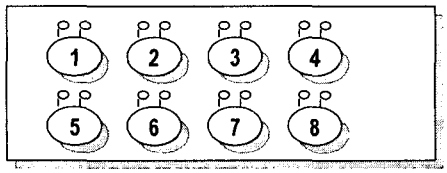
Conceptual Design (Process Modeling)

Scenario :
- Business Context Diagram
- Workflow Process Diagram
- Task Sequence Diagram



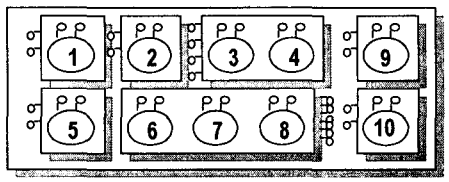
Logical Design (Object Modeling)

Object/Service :
- Object/Attribute
- Object Interaction Diagram
- Service Generalization
- Class Diagram
-> UI Scratch, ERD



Physical Design (Component Modeling)

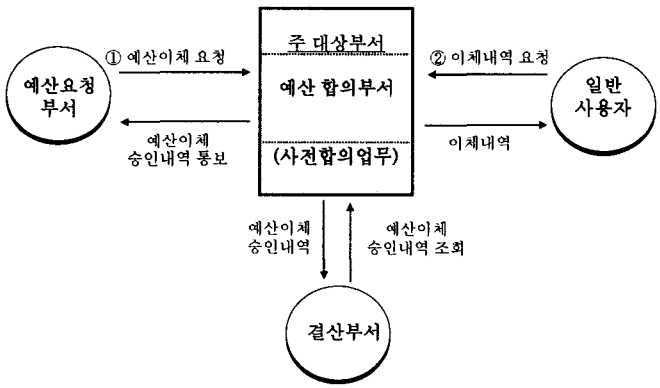
Component :
- Component Interaction D
- Application Structure
- Component Specification
-> UI Design/UI Spec, Physical DB



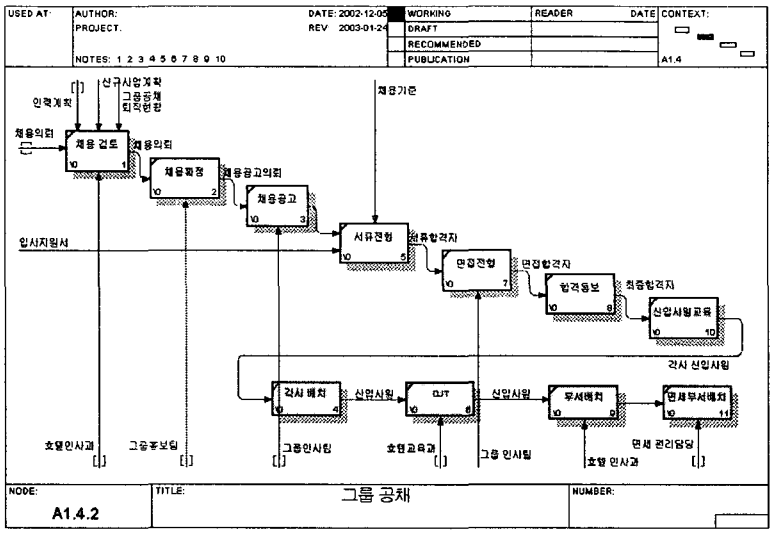
Conceptual Design

- **Business Context Diagram**
- **Workflow Process Diagram**
- **Use Case Scenario / Task Sequence Diagram**

Business Context Diagram



Workflow Process Diagram



Use Case Scenario

급여	XX 회사 업무 재구축		최초시행	1999.7.
	개발경로	MSF / CD		
	스테이지	Conceptual Design	작성자	홍길동
	테 스 크	Use Case Scenario	확인	
UseCase Name	구 분	Step/Description		비 고
연말정산 확정처리	사전준비사항	1. 연말정산 계산 내역서를 준비한다.		
	업무처리절차	1. 연말정산 환급금을 산출한다. 환급금 = (연말정산 결정세 - 기납부 결정세) + (연말정산 주민세 + 기납부 주민세) 2. 연말정산 내역으로 근로소득원천징수부(국세청 기준 자료), 전산매체(카트리지 테이프)를 작성한다.		
	사후준비사항	1. 환급금을 1월 급여에 적용하기위해 개인별 환급금 내역을 급여담당자에게 준다. 2. 개인별 환급금 내역을 재무담당자에게 준다. 3. 개인별 연말정산 내역을 회계담당자에게 준다.		

Logical Design

MSF 어플리케이션 모델에 따라 서비스 기반 조직 원리들을 적용하고, 솔루션의 구조 및 솔루션의 각 부분들 사이의 관계를 설계하는 것

- 정의

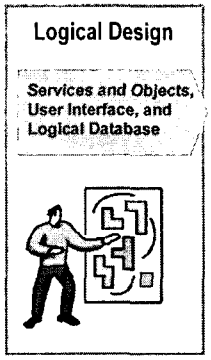
  프로젝트 팀의 관점에서, 솔루션의 각 부분들의 조직과 구조와 계통적 배열(syntax), 상호작용에 기초를 두고, 솔루션을 묘사하는 프로세스

- 산출물

  서비스(services)와 속성(attributes), 관계(relationships)를 갖는 오브젝트집합과 상위수준의 유저 인터페이스 설계, 논리적 데이터 베이스 설계

- 설계 관점

  프로젝트 팀의 관점



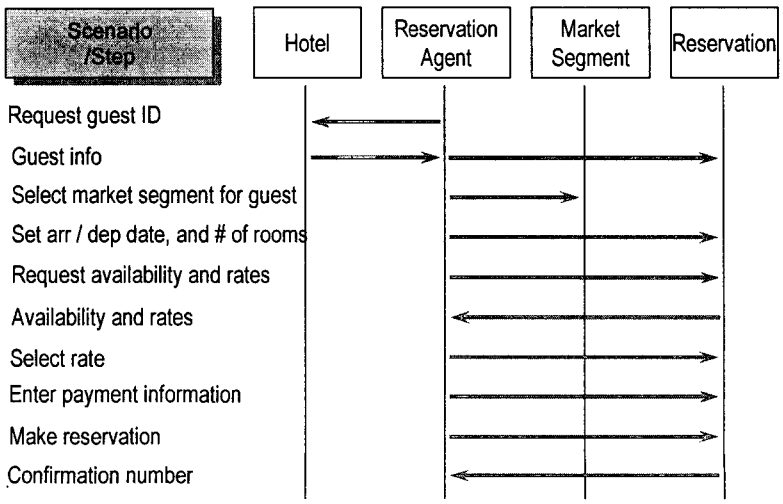
Logical Design

- **Object 추출 및 정리**
- **OID (Object Interaction Diagram)**
- **Services Generalization**
- **Pseudo Code**
- **UI Scratch**
- **ERD**

Object 추출 및 정리

Use Case 명	Object 명(동사)	Class 명	Attribute 명
고객등록/변경	고객정보	고객	
고객등록/변경	변경사항	고객	
고객등록/변경	고객등록	고객	
고객등록/변경	기등록고객	고객	등록구분
고객등록/변경	신규요청고객	고객	등록구분
고객등록/변경	성명	고객	성명
고객등록/변경	주민등록번호	고객	주민등록번호
고객등록/변경	주소	고객	주소
고객등록/변경	신용정보	고객신용정보	
고객등록/변경	신용불량자	고객신용정보	신용등급
대금지급요청	대금지불요청여부	대금청구접수내역	대금지불요청여부
대금청구접수	정상청구여부	대금청구접수내역	정상청구여부
대금청구접수	대금청구요청내역	대금청구접수내역	
대금청구접수	대금지급요청	대금청구접수내역	
대금청구접수	수량	반주	반주수량
대금청구접수	단가	반주	반주단가
대금청구접수	일자	반주	반주일자
대금청구접수	반주 번호	반주	반주 번호

Object Interaction Diagram

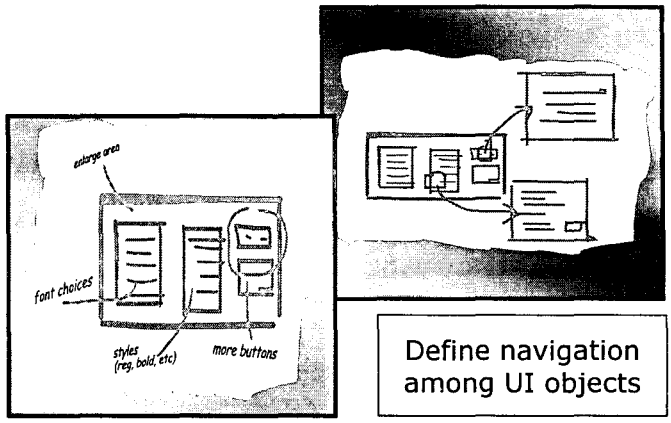


Service 일반화

CLASS	서비스명	Param-in	Param-Out	Use Case Name
거래은행	TransferReq	Name,Account,금액	Status	집행
거래처	Inquiry()	요청항목들, 조건 명세	요청항목값들	지불방식결정,집행,결재
거래처	QueryAccount	BuyerName,CompanyID	Bank,Account	집행
거래처	QueryGrade	BuyerName	Grade	지불방식결정
거래처	QueryRisk	BuyerName	Credit,Bankrupt	결재
결재	Approve	BuyerName	Status	결재
결재	Inquiry()	요청항목들, 조건 명세	요청항목값들	
결재	PayTypeRegister	BuyerName,CashFlag(지불방식)	Status	지불방식결정
결재	Query	뭘	Confirm 대상들	결재
고객	CheckExists	주민번호	결과메시지	고객정보등록/변경
고객	CheckExists	조건명세	결과메시지	고객정보등록/변경
고객	CheckExists	고객명	등록여부	주문접수
고객	Inquiry()	요청항목들, 조건 명세	요청항목값들	집행
고객	Insert	고객정보	확인 메시지	주문접수
고객	Modify	주민번호,성명,주소,조건	결과메시지	고객정보등록/변경
고객	Modify	Set of (항목=항목값), 조건명세	결과메시지	고객정보등록/변경
고객	QueryAccount	CustomerName,SocialID	Bank,Account	집행

Copyright © 2003 Aurora.net Nice Team

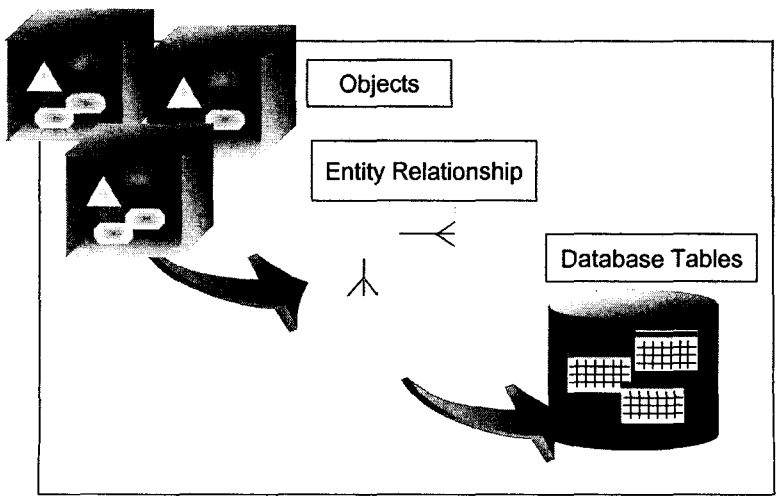
UI Scratch



Define navigation among UI objects

Define UI objects and interactions

Database Design



Physical Design

구현과 성능 고려 사항들을 포함한, 실세계 기술 제약들을 논리적 모델에 적용하는 것

- 정의

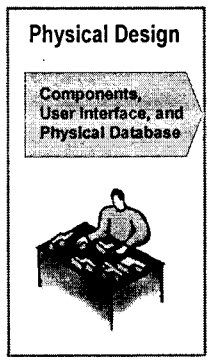
  개발 팀의 관점에서 솔루션의 컴포넌트와 서비스와 기술을 묘사하는 프로세스

- 산출물

  특정 플랫폼에 대한 컴포넌트들과 사용자 인터페이스 설계 모음, 그리고 physical 데이터베이스 설계

- 설계 관점

  개발자 관점

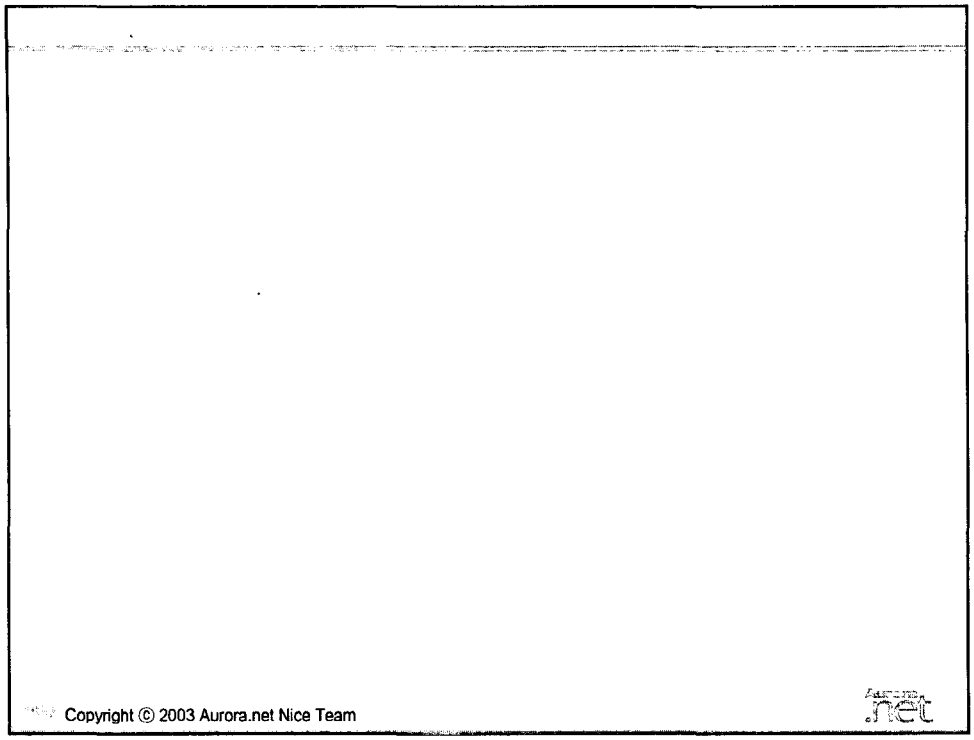


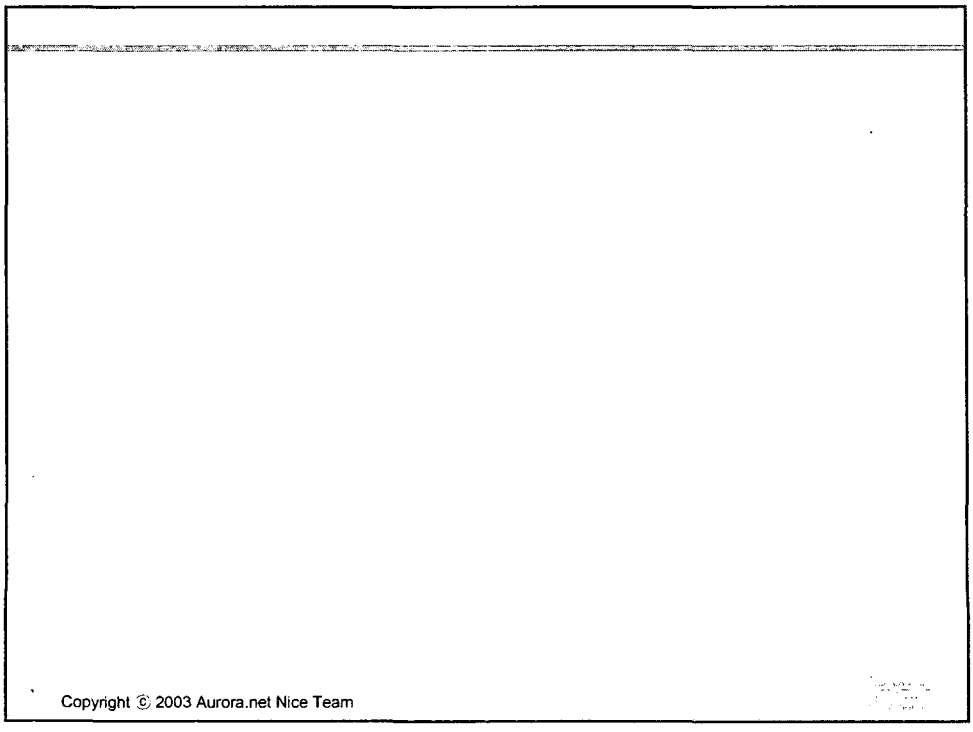
II. MSF/CD 개발 방법론

Physical Design

- **CID (Component Interaction Diagram) 1차, 2차**
- **Application 구조도**
- **Component Specification**
- **UI Specification**

Copyright © 2003 Aurora.net Nice Team


Copyright © 2003 Aurora.net Nice Team


Copyright © 2003 Aurora.net Nice Team

Application Structure

System	Application	Component	Class	Service	Parameter		Logic
					In	Out	
회계	회계관리	회계처리	회계전표	Inquiry	String StrContent,String StrCondition, Optional RS	RS	
	ACT	A_C001	clsACTAcctSlip	InqSijae	String StrContent,String StrCondition, Optional RS	RS	
				InqSlip	String StrContent,String StrCondition, Optional RS	RS	
				InqSlipAcct	String StrContent,String StrCondition, Optional RS	RS	
				InqBatchSlip	String StrContent,String StrCondition, Optional RS	RS	
				AccountProc	String StrContent, Array AcctAry	String StrResultMessage	
		회계마감	조작자마감정보	Register	String StrContent	String StrResultMessage	
		A_C002	clsACTOpClsInfo	RegisterOpClose	String StrContent	String StrResultMessage	
				CancelOpClose	String StrContent	String StrResultMessage	
				Inquiry	String StrContent,String StrCondition	RS	
				CheckClose	String StrCondition	String StrResultMessage	
				InqOpClose	String StrContent,String StrCondition	String StrResultMessage	

Component Specification

SYSTEM		호텔관리
Application		CheckIn
Component		Reservation
Class		clsReservation
service	public	private

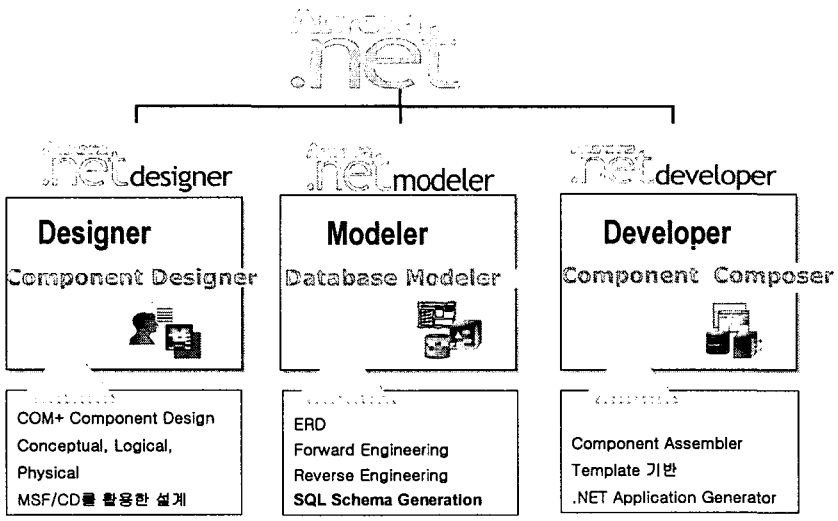
Check	Public Check
	connect to DB using CommDB
	IF table.rows >0 then
	return "Y"
	ELSE
	return "N"
	END IF
	End Public	

Inquiry	Public Inquire Connect to DB using CommDB select 예약항목들모두 (*) from input parameter (예약번호 or 고객명) End Public	

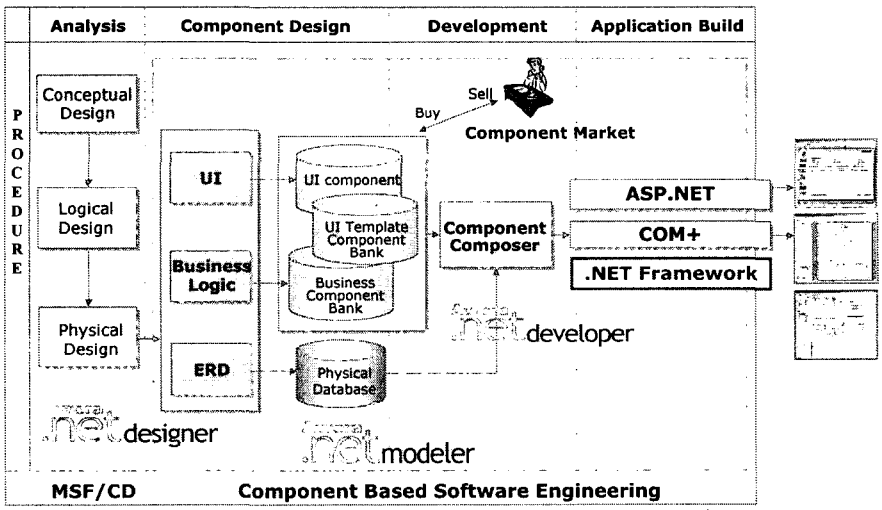
UI Specification

UI Layout	Spec.
일일 마감 기준일자 : XXXX-XX-XX (default:영업일자) 거래구분 : X (1.마감, 2.취소) < 입기장 조회 > 계정코드 계정과목 당일잔액 차변건수 차변금액 대변건수 대변금액 전일잔액 ------ ------ ------- ------- -------- ------- ------ -- -------	기준일자 = System 일자 (Default) [등록] Edit Request_string RM = call 총계정.Register(Request_string) Display RM [취소] Edit Request_string call 총계정.Cancel(Request_string) Display RM [조회] : 당일에 입출이 발생한 기표계정만 조회 RS = call 회계전표.Inquiry RS - > Display

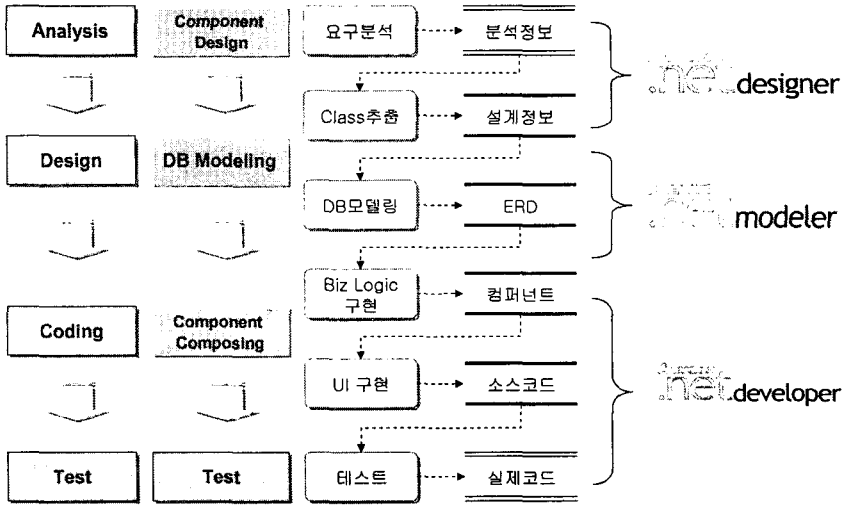
Aurora.net 제품 소개



Aurora.net 개념 구조도



Aurora.net을 이용한 시스템 개발 절차



Aurora.net Designer는 MSF/CD 방법론을 탑재하여 CBD(Component Based Development)를 지원하는 차세대 통합형 Component Design 솔루션

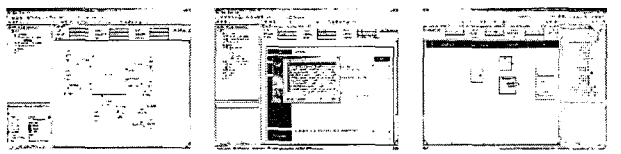
Aurora. net Designer (CBD, Component Design Solution based on MSF/CD)

◆ 기능

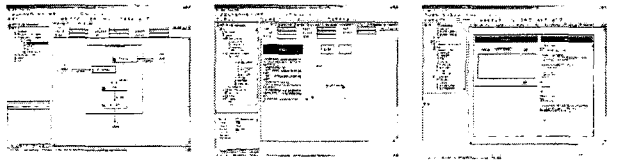
- ❑ **Conceptual Design**
  - Business Context Diagram
  - Workflow Process Diagram
  - Use Case Scenario
- ❑ **Logical Design**
  - Object Extraction
  - Object Interaction Diagram(OID)
  - Service Generalization
  - UI Scratch
  - Pseudo Code for Services
  - Entity Relation Diagram
- ❑ **Physical Design**
  - Component Interaction Diagram(CID)
  - Application Structure Chart
  - Component Specification
  - UI Specification
  - Physical Database

◆ 주요 화면

▸ **Conceptual Design** ▸ **Logical Design** ▸ **Physical Design**



Business Context Diagram | Object Extraction | Component Interaction Diagram



Workflow Process Diagram | Object Interaction Diagram | UI Specification

Ⅰ. Aurora.net 소개

Aurora.net Modeler는 Database를 필요로 하는 모든 프로젝트 및 개발 System 운영에 사용될 수 있는 솔루션으로서 DB Modeling 및 Remodeling을 하는 솔루션

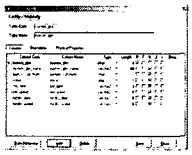
Aurora.net Modeler (Data Base Modeling Solution)

◆ 기능

- ERD(Entity Relation Diagram)
- Forward Engineering
- Reverse Engineering
- User Security Management
- SQL Schema Generation
- Data Dictionary
- Report(Document)

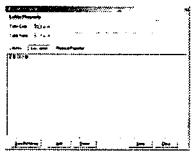
◆ 주요 화면

▸ ERD 제작



Edit Attribute

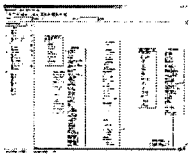
▸ Edit Attribute



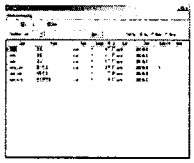
Edit Description

▸ SQL Generation

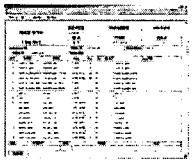
SQL Schema Generation



물리적 ERD 생성



Data Dictionary



Report(HTML,EXCEL,WORD)

Ⅰ. Aurora.net 소개

Aurora.net Developer는 .NET Framework에서 운영되는 소스(C#, ASP.net 등) 및 COM+ Component를 Generation하는 Component 기반의 솔루션

Aurora. net Developer (C# Development Tool based on .NET)

◆ 기능

❑ Project Management
Project Registration
User Information
Application Environment Setup

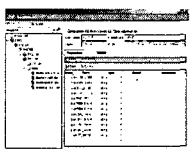
❑ Database Modeling
Import Database
(Database, Script, ERD)
Database Administration

❑ Application Building
COM+ Management
Program Items
Component composer

❑ Component Repository
System UI Template
Template Assembly
System Component

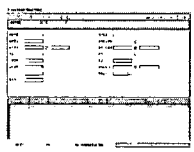
◆ 주요 화면

▸ Application Building



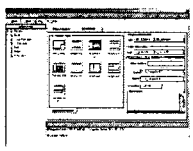
COM+ Management

▸ Component Building

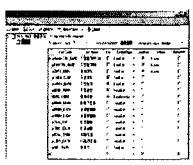


Class Based UI Builder

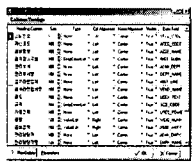
▸ Component Repository



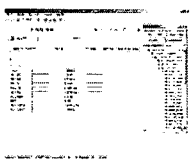
UI Template



Program Items



Column Design



Component Composer

II. Aurora.net Demo

Aurora.net Demo

aurora.net designer

aurora.net modeler

aurora.net developer