



UML 기반의 ERP 개발방법론

박 화 규[†] 백 중 명^{**}

◆ 목 차 ◆

1. 서 론
2. 제안 ERP 개발방법론
3. 결 론

1. 서 론

성공적인 ERP 시스템을 개발을 위해서는 합리적이고 다양한 모델링 표현력을 갖는 방법론을 활용하여야 한다. 이러한 개발 방법론은 ERP의 구축을 위하여 중요한 도구가 되며 타 방법론들과 Benchmarking 되어 환경에 맞는 차별성이 있어야 한다. 따라서 개발 방법론은 기본 구조를 바탕으로 다양한 업무영역에 대하여 높은 Modeling Power를 갖고 업무중심의 접근에 대한 적용원리가 합리적이고 유연해야 하며, CASE Tool 및 체계적 Documentation을 위한 지원도구가 있어야 한다.

이러한 방법론에는 ERP공급회사나 컨설팅 회사가 자체적으로 개발하여 보유하는 것을 활용할 수도 있고, 범용적인 개발 방법론을 사용할 수 있다. 즉, ERP를 자체적으로 개발하는 경우와 ERP 패키지를 도입(Packaged ERP System) 하는 경우(In-House System, Home-Grown System)로 분류되는데 전자의 경우는 대표적으로 SAP R/3의 ASAP(Accelerated SAP), Oracle Manufacturing의 AIM(Application Implementation Methodology) 및 Knowledge Ware의 ADW(Application Development

Workbench), PriceWaterhouse의 SMM(System Management Methodology)등이 있으며 후자의 경우, J. Martin의 정보공학 방법과 Rumbaugh가 제안한 OMT(Object Modeling Technique)외에 현재 OMG의 표준 방법론으로 자리잡은 UML(Unified Modeling Language)등이 있다. 본 논문에서는 UML1.1기반의 ERP개발 방법론 절차에 대해 기술한다.

1.1 UML 방법론

UML은 크게 4부분으로 나누어진다. 첫번째가 사용자의 요구사항 정리에 필요한 Use Case모델링이며 두 번째가 현실세계의 복잡한 문제들을 추상화하여 풀어야 하는 문제의 범위를 개념적이고 현실적으로 표현하는 Class와 Object 모델링이다. 세번째는 소프트웨어의 레고 블럭식 조립이 가능하도록 하는 개발 Component의 모델링이며, 네번째는 개발 후의 소프트웨어 컴포넌트를 복잡한 분산처리나 Client/Server 환경에 물리적으로 어떻게 배치할 것인지에 요구되는 Implementation 다이어그램이다.

1.2 UML Meta 모델

UML의 메타 모델은 UML자체의 분석/설계 모델을 의미한다. 즉, 개발하고자 하는 ERP 시스템을 위해 모델링을 할 때, 여기서 ERP시스템이

[†] 정회원 : 한국전자통신연구원 연구원

^{**} 정회원 : 한국전자통신연구원 정보통합연구팀장

UML에 해당되며 모델링이 된 결과물이 바로 UML 메타모델이 되는 것이다. UML 메타 모델의 모델링 구성 요소는 다음과 같다.

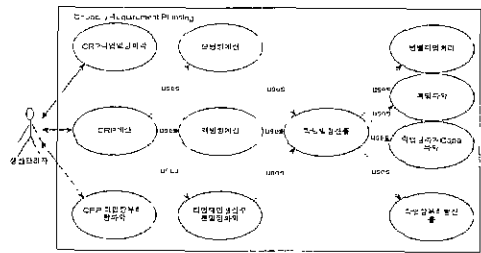
- **Class**: 공통된 구조와 행동을 가진 객체들의 개념적인 집합체
- **Use Case**: 상호 연관된 객체에 의해 시스템의 기능을 완성하는 단위
- **Pattern**: 업무 처리에 사용되는 일반적인 형태 혹은 폼
- **State**: 한 객체가 시간과 공간 그리고 외부 사건에 대해 정적이면서 동적의 상태의 임의의 값을 갖기 위한 조건
- **Interface**: 한 객체의 공용부분
- **Component**: 논리적이면서 물리적인 재사용 요소
- **Node**: 완성된 소프트웨어 시스템이 운영되는 하드웨어 시스템
- **Package**: 모델링 구성요소의 논리적/물리적 집합체
- **Note**: 부연 설명을 위한 요소
- **Association Relation**: 두 구성 요소간의 개념적인 관계
- **Generalization**: 일반화된 모델링 구성 요소와 구체화된 구성 요소간의 분류(taxonomic)를 나타낸 관계
- **Dependency Relation**: 한 모델링 구성요소(Client)가 다른 구성요소(Server)를 이용하는 관계
- **Refinement Relation**: 점차로 추상화 수준을 구체화하는 관계
- **UML 자체를 사용자가 확장할 수 있도록 하는 Mechanism**: 명세(Spec.)서, 스테레오 타입(Stereotype), 제한(Constraint), 태그 값(Tag Value)

위의 모델링 구성요소들은 상세한 설명을 위한 명세부분, 추상화 수준에서 편리한 분류(Classification)를 위한 스테레오 타입, 모델링의 범위를 구체화하기 위한 제한 그리고 부연 설명을 위한 태그 값들로 구성된다.

이러한 구성요소를 이용해 ERP 모델링시 도출되는 다이어그램은 9개로 구성되며 다음과 같은 형태와 특성을 갖는다. 이는 본 방법론의 분석과 설계단계에서 사용되어 진다.

9개 UML Diagram

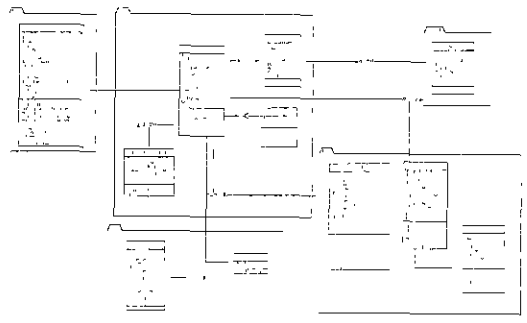
a. Use Case Diagram



(Fig 1) Use Case Diagram

- ▶ Use case는 기능적인 요구를 정의
- ▶ ERP 각 단위 모듈의 외부 Actor와 System 이 제공하는 여러 개의 Use Case를 연결
- ▶ ERP 시스템이 제공하는 기능성을 설명

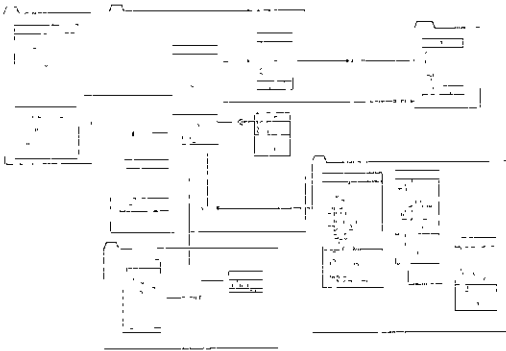
b. Class Diagram



(Fig 2) Class Diagram

- ▶ 클래스들은 정적인 구조로 ERP 시스템에서 다루어지는 “things”으로 표현됨
- ▶ 클래스들은 연관된, 특수화된, 또는 패키징된 방법으로 다른 것들과 각각 관계있음

c. Object Diagram

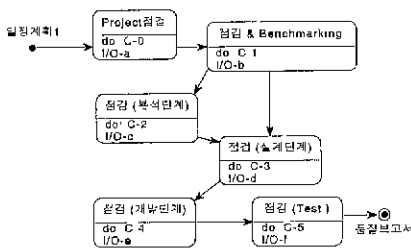


(Fig 3) Object Diagram

- ▶ 클래스의 여러 오브젝트 인스턴스를 나타내는 대신에 실제클래스를 사용
- ▶ 클래스 다이어그램에서 2가지 예외사항을 제외하고는 동일한 표기법을 사용
 - 오브젝트의 이름에 Underline
 - 관계있는 모든 인스턴스를 표현

d. State Diagram

Project관리



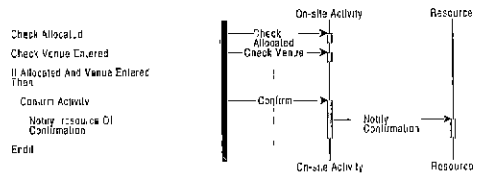
(Fig 4) State Diagram

- ▶ 클래스 설명을 보완
- ▶ 클래스의 객체가 갖을 수 있는 모든 가능한

상태를 모델링

- ▶ 모든 클래스들을 그리지는 않는 단지 여러 개의 잘정의된 상태와 클래스의 행위가 다른 상태에 의해 어떤 영향을 미치고 변화를 주는지에 대해 정의

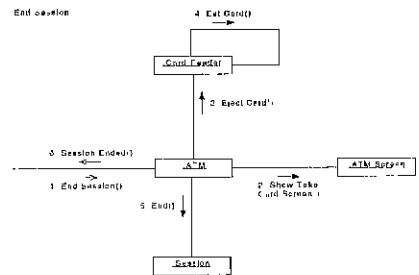
e. Sequence Diagram



(Fig 5) Sequence Diagram

- ▶ 여러 개의 객체들 사이에 동적인 협력사항으로 Use Case에서 분기된 객체 사이에 메시지를 보내는 순서를 표기
- ▶ 수직선상의 여러 개의 오브젝트들로 구성
- ▶ 시간은 아래방향으로 진행을 하고 순서나 함수로 시간이 지나가는 것에 따라서 오브젝트들 사이에 메시지의 교환
- ▶ 메시지는 수직적 오브젝트 라인 사이에 메시지 화살표를 갖는 라인

f. Collaboration Diagram

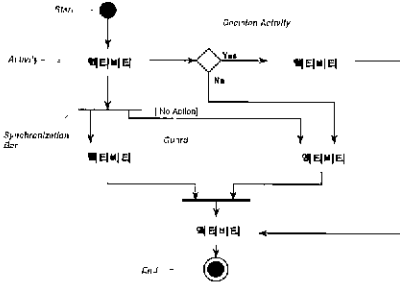


(Fig 6) Collaboration Diagram

- ▶ 시퀀스 다이어그램처럼 동적인 협력사항 표기
- ▶ 오브젝트와 그들의 관계를 보여준다

- ▶ 만일 시간 또는 순서가 강조해야 할 중요한 특징이라면 시퀀스 다이어그램을 선택 하고 내용이 중요한 경우에는 콜래버레이션 다이어그램을 선택

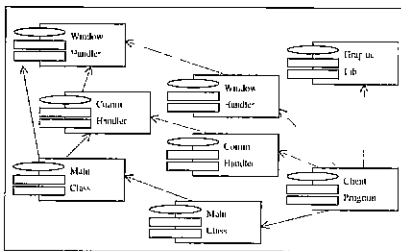
g. Activity Diagram



(Fig 7) Activity Diagram

- ▶ 전형적으로 연산자로 수행된 액티비티를 설명하기 위해 사용
- ▶ 수행되기 위한 액티비티의 명세를 포함함 액션상태로 구성
- ▶ 액션상태의 병렬수행의 결정과 조건표기

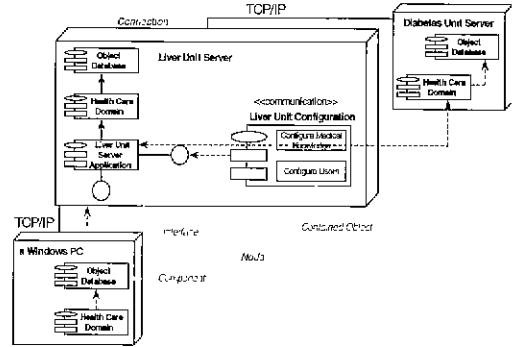
h. Component Diagram



(Fig 8) Component Diagram

- ▶ 코드 컴포넌트에 바탕을 둔 코드의 물리적인 구조로 컴포넌트는 논리적 클래스 또는 클래스 자신의 구현에 대한 정보를 포함
- ▶ 실질적인 프로그램 작업에서 사용

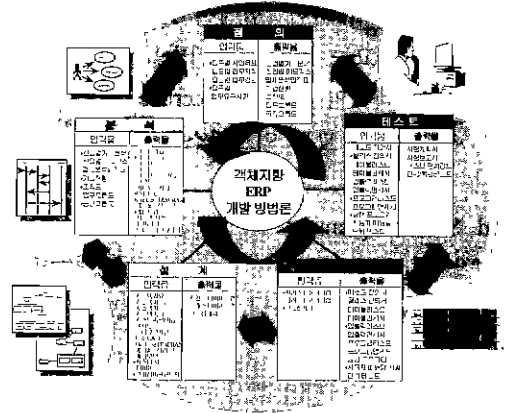
i. Deployment Diagram



(Fig 9) Deployment Diagram

- ▶ 시스템의 하드웨어와 소프트웨어의 물리적인 구조로 실질적인 컴퓨터와 디바이스(노드)와 컴포넌트 사이의 종속성을 표현

Fig. 10은 이러한 구성요소를 바탕으로 단계별 입력 및 산출물을 갖는 ERP시스템의 객체지향 개발의 예를 나타낸다.



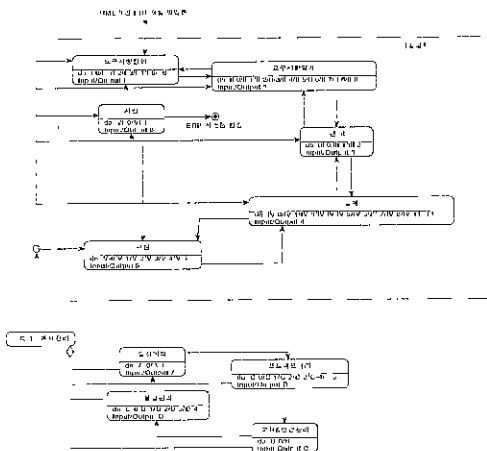
(Fig 10) ERP모델링 단계별 입력 및 산출물 예시

2. 제안 ERP 개발방법론

본 UML기반의 추진 방법론은 ERP 시스템 구축을 계획하는 기업이 자기기업에 적합한 시스템을 선정하고 이를 활용하여 ERP시스템을 구축하는

절차적 방법론이다. 이러한 추진과정은 A, B, C를 구성하는 Project/Quality 통제관리 활동과 I, II, III, IV, V, VI을 구성하는 요구사항분석부터 테스트에 이르는 개발절차 및 활동으로 분리되어 있고, A, B, C와 I, II, III, IV, V, VI는 상호 밀접한 Interaction을 갖으며 추진된다.

다음 Fig. 11 State Diagram은 본 방법론의 상위 레벨의 View를 설명한다. 추진방법론 구성은 일정계획의 3단계, 교육 & 변경관리의 3단계, 프로젝트 관리 및 품질관리의 11단계외에 요구사항정의 및 명세의 17단계, 분석의 3단계, 설계의 13단계, 구현6단계 및 시험의 2단계로 구성되어 있고 각각은 Segment로 분리되어진다. 이는 Windows CASE도구로 SELECT Ver.7.0을 활용하고, 각 문서 형식 및 산출물은 MS-Office를 이용한다. 세부적 흐름을 나타내는 Fig. 12 ~ Fig. 21은 Child State Diagram으로 Fig. 11 상위 뷰의 각 단계를 세분화한 것이다.



(Fig 11) 방법론의 상위 뷰

그림에서와 같이 ERP개발영역은 여러 개발 주기를 거쳐 시스템이 개발되는데 이 개발 프로세스의 특징은 어느 한 시점에서 현재 진행중인

ERP 프로젝트가 분석, 설계, 구현을 동시에 수행하고 있다는 것이다. 하나의 개발 주기 진행 중에 미비한 점이 있다면 새로운 개발 주기를 출발시켜 시스템의 기능성을 정의하고 이를 설계하고 필요한 구현을 완성해서 지속적인 시스템의 버전을 만들어 나가는 방법이다.

2.1 프로젝트 통제관리 영역

프로젝트 통제관리는 프로젝트에 참여하는 구성원의 담당업무 및 산출물에 대한 일정관리와 프로젝트 수행에 대한 계획대비 실적에 대한 진도체크 및 교육체계 실행 & 변경관리에 대한 영역으로 단계별 발생되는 문제점에 대해 해결방안 모색하며 프로젝트 각 단계별 산출물들의 품질을 보증한다.

일정 계획 및 프로젝트관리는 프로젝트가 정해진 시점에 완료될 수 있도록 하기 위한 Activity정의, 순서화, 활동기간의 추정, Gantt Chart 기반의 일정과 이에 따른 Critical Path를 통제한다.

품질관리는 품질 계획수립, 통제, 보증, 개선을 통한 품질 정책, 목적, 책임을 결정하는 관리기능으로 기능성, 신뢰성, 사용성, 효율성, 보수성, 이식성의 관점에서 수행된다.

교육 & 변경관리는 BPR과 ERP시스템 구축 활동과 병행하여 기술적, 기능적 부분의 관점에서 수행하며, 특히, 변경관리는 프로젝트 전반에 걸쳐 발생되는 변경항목에 대해 분석하고 대응하기 위한 목적을 갖는다. 다음은 각 활동 단계별 산출물과 이에 대한 간략한 설명이다.

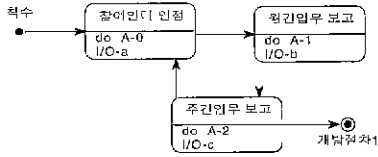
1) 일정계획

A-0. 참여인력 일정표: 프로젝트에 참여하는 구성원의 담당업무에 대한 일정관리

A-1. 월간 업무현황 보고서: 프로젝트 수행에 대한 월간 보고를 통하여 계획 대비 실적에 대한 진도체크와 문제점에 대한 해결

방안 모색

일정계획

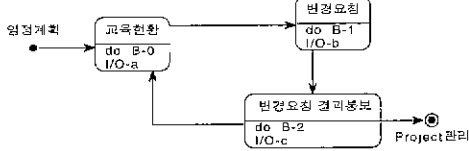


(Fig 12) 일정계획

A-2. 주간 업무현황 보고서: 프로젝트 수행에 대한 주간 보고서로써 계획 대비 실적에 대한 진도체크와 문제점에 대한 해결방안 모색

2) 교육 & 변경관리

1 [교육 & 변경관리]

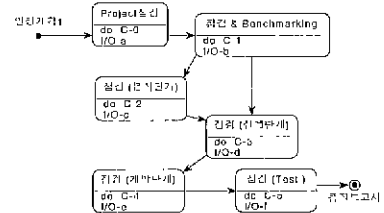


(Fig 13) 교육 & 변경관리

- B-0. 교육현황: 프로젝트 수행에 도움이 될 수 있는 기술 습득이나 관련 교육에 대한 교육현황 정리
- B-1. 변경 요청서: 변경요청서는 개발의 초기단계부터 시스템의 운영지원 단계까지 프로젝트 전단계에 걸쳐서 진행된다. 변경요청은 해당관리자의 변경에 대한 승인이 있는 후에 시행되어야 하며 변경 내용들의 변경상태를 보증
- B-2. 변경요청 결과 통보서: 과제별로 제시된 변경요청서에 대한 해당관리자의 승인처리이며 변경 내용들의 변경상태를 보증

3) 프로젝트관리

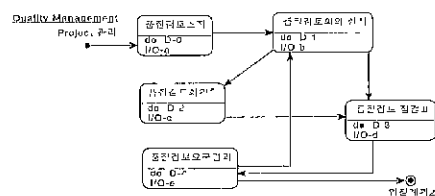
Project 관리



(Fig 14) 프로젝트관리

- C-0. 프로젝트 점검사항: 프로젝트를 원활히 수행할 수 있는지를 평가(일정, 조직 등)
- C-1. 점검 및 벤치마킹: 해당 분야의 시스템에 대한 선진 제품에 대한 분석 자료 유지관리
- C-2. 점검사항(분석단계): 현업무와 그에 따른 개발시스템의 요구사항에 대한 정리 및 총실성
- C-3. 점검사항(설계단계): 개발시스템의 기능들을 모형화 하는 단계로 요구분석내용의 총실 및 정확성 평가와 산출물 확인
- C-4. 점검사항(개발단계): 시스템 설계단계에서 정의된 모듈들의 수행내역을 프로그램으로 전환되고 구현한 것에 대한 평가
- C-5. 점검사항(시험단계): 각각의 프로그램을 구현하고 난후 각각의 독립적인 상태에서서의 평가 프로그램 모두를 구현후 통합하여 시스템 평가

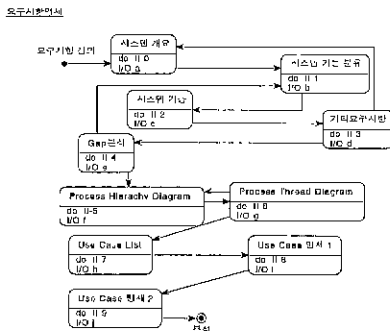
4) S/W 품질관리



(Fig 15) S/W품질 관리

- I-4. 대상기업 조직도: 대상적용 기업의 조직구조를 파악하기 위하여 대상기업 조직도 작성
- I-5. 대상기업 업무 흐름도: 대상적용 기업의 업무처리 절차 및 결재라인을 파악하기 위하여 작성
- I-6. 대상기업 공정 흐름도: 대상적용 기업의 제품생산에 관련된 공정흐름을 파악하고 제품이 제조 생산되는 과정을 알기 위하여 작성

II. 요구사항 명세(Requirement Specification)



(Fig 17) 요구사항 명세

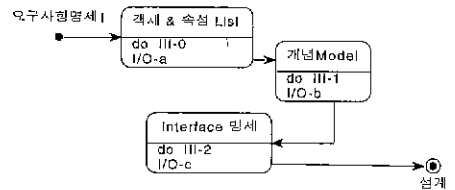
- II-0. 시스템 개요: 시스템의 개발의의, 개발목적, 개발범위, 업체적용 시 도입효과 등을 개략적으로 기술하여 시스템의 전반적인 내용을 파악함에 목적있음.
- II-1. 시스템 기능분류: 시스템의 기능을 사용자가 직접 작업하거나 시스템 내부에서 처리되거나 Option 사항으로 처리되는가를 구분하기 위하여 작성하고, 시스템기능에서 기능에 대한 분류방법으로 사용
- II-2. 시스템 기능: 업무를 타 기능과 중복되지 않게 최소단위로 분할하고 각 기능을 하나의 Method로 만들 수 있도록 한다.
- II-3. 기타 요구사항: 시스템과 관련하여 부가적으로 요구해야 할 사항이나 전체 조건들

을 기술

- II-4. Gap 분석서: 시스템에 대한 기능을 정의 후, 대상적용 업체의 Business Process를 분석할 경우의 차이를 자세하게 기술
- II-5. Process Hierarchy Diagram: PHD List에는 PHD File Name, PHD Name, PHD Level, 비고로 구성되어 있고, PHD를 List-up하는데 목적이 있다.
- II-6. Process Thread Diagram List: PTD List에는 PTD File Name, PT Name, 비고로 구성되어 있고, PTD를 List-up하는데 목적이 있다.
- II-7. Use Case List: 서버 시스템별 존재하는 Use Case를 기술하여 Use Case를 List-up하여 관리하는데 목적이 있다.
- II-8. Use Case 명세서I: Use Case에 관련된 Actor, 목적, 개요, 유형, 참조기능으로 구성되어 있고 Use Case의 목적 및 행위를 기술하고, 이후 Sequence Diagram을 작성하는 지침이 된다.
- II-9. Use Case 명세서II: Sequence Diagram을 작성하기 전 단계에서 Actor와 Use Case의 상호 업무처리 절차를 순차적으로 간단하게 기술하고, Sequence Diagram을 작성하는 지침이 된다.

III. 분석 사양(Analysis Specification)

분석



(Fig 18) 분석단계

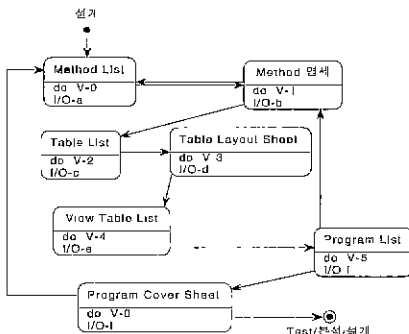
- III-0. 객체 & 속성 List: 객체 & 속성리스트는 작업을 반복 수행하면서 보완작업을 하

- IV-11. 세부코드 설계: 코드체계에 대한 자료를 기술하여 관리함으로써 프로그램 구현 및 업무를 처리하는데 효율을 높이기 위하여 작성
- IV-12. Message List: 개발자가 프로그램을 개발하거나 사용자가 시스템을 사용하는데 사전에 Error Message나 권장사항 등을 작성하여 작업의 효율을 높이기 위하여 작성

V. 구현 사양(Implementation Specification)

- V-0. Method List: Method List에는 Method Name, 비고로 구성되어 있고, Method를 List-up하는데 목적
- V-1. Method 명세: Class내의 Method에 대한 상세한 내용을 작성하여 이후 Business Process Logic이 변경되었을 경우 해당 Method만을 수정하기 위하여 작성
- V-2. Table List: Table List에는 Table ID, Table Name, 비고로 구성되어 있고, Table을 List-up하여 관리하는데 목적
- V-3. Table Layout Sheet: Table Layout Sheet에는 Attribute ID, Attribute Name, Data Type, Length, Key 등을 기술하여 관리하는데 목적

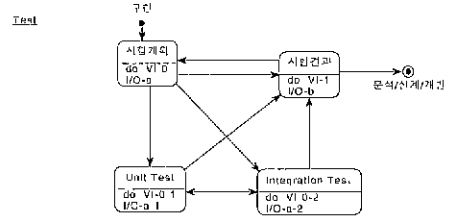
구현



(Fig 20) 구현 단계

- V-4. View Table List: View Table List에는 View Table ID, 내용으로 구성되어 있고, View Table을 List-up하여 관리하는데 목적
- V-5. Program List: Program List에는 Program ID, Program Name, 비고로 구성되어 있고, Program을 List-up하여 관리하는데 목적
- V-6. Program Cover Sheet: 시스템 운영 시 사용자가 Data를 입력하거나 Event를 발생시키면 시스템 내부적으로 자료가 처리되고 화면에 처리된 자료를 출력하는 Process를 순차적으로 기술한다. 사용자가 업무를 효율적으로 수행할 수 있도록 작성

VI. 시험 사양(Test Specification)



(Fig 21) 테스트 단계

- VI-0. 시험 계획서: 소프트웨어 시스템 시험은 사용자 요구사항, 설계, 구현의 전과정에 대한 점검이며, 체계적인 소프트웨어 시스템을 테스트하기 위하여 작성한다. 단위 테스트는 프로그램이 완료될 때마다 프로그램을 테스트하기 위하여 사전에 작성하고, 통합 테스트는 전체 시스템 개발이 완료되고 단위 테스트를 마친 경우에 테스트하기 위하여 작성
- VI-1. 시험 결과서: 시험계획서를 바탕으로 소프트웨어 품질을 테스트하여 그 결과를 기록하고 불합격 사항에 대해서는 수정 예정일을 확정하고 수정사항에 대해서는 사후관리를 반드시 해야 한다.

3. 결 론

ERP시스템은 기업내의 정보 기반구조의 핵심인 동시에 EC, CALS환경에 기반이 되며, 개발방법론은 이러한 기반구조 시스템 구축을 위한 Guide-line 역할을 한다. 따라서 성공적인 ERP구축을 위한 개발 방법론은 합리적이고 고 모델링 표현력으로 각 과정별 필요시점에 적절한 입력 및 산출물 생성을 지원해야 한다. 본 논문에서 제안된 UML기반 개발방법론은 서로 다른 여러 개의 기능적, 비 기능적, 조직적인 관점을 표현하는 5개 뷰인 Use-case view, Logical view, Component view, Concurrency view, Concurrency view, Deployment view들이 모이지고 통합되는 과정을 통해 ERP시스템의 거시적이고 미시적인 모든 요구사항이 보다 정확히 표현될 수 있음을 보여준다. 또한 UML기반의 요구사항 분석 및 구현, 테스트의 과정에 요구되는 개발 및 활동절차외에 각 절차를 관리 보증하는 Project/Quality 통제관리 활동을 추가하였다.

참고문헌

[1] Clemens Szyperski, Component Software : Beyond Object-Oriented Programming, Addison-Wesley, 1998.

[2] R Helm, I. Holland, and D. Gangopadhyay "Contracts: Specifying Behavioural Compositions in Object-Oriented Systems," Proceedings of OOPSLA/ECOOP, 1990.

[3] Roser Jennings, Database Workshop - Microsoft Transaction Server 2.0, SAMS Publishing, 1998.

[4] Mary Shaw, "Architectural issues in Software Reuse," Proc IEEE Symposium on Software Reusability, April, 1995.

[5] Kazuo Matsumura, Akihiro Yamashiro, Toshiyuki

Tanaka, and Ikumune Takahashi, "Modeling of Software Reusable Component Approach and its Case Study," IEEE, 1990.

[6] Ted Biggerstaff and Charls Richter, "Reusability Framework, Assessment, and Directions," IEEE Software, pp. 41-49, 1987.

[7] Orfali, Harkey, The Essential Client/Server Survival Guide, Wiley, 1996.

[8] Orfali, Harkey and Edwards, The Essential Distributed Object Survival Guide, Wiley, 1996.

[9] Genensis, SELECT Software Tools Manufal, SELECT PLC. 1998.

[10] 박화규외 객체지향형 표준정보시스템 개발자 지침서, 한국전자통신연구원, 1998.

[11] 이동길, "ERP 전략과 실천" 대청출판사, 1999.



박 화 규

1990년 캘리포니아주립대 산업공학
학과 (석사)
1990년 캘리포니아주립대 산업공
학과 (석사)
1993년 오클라호마주립대 산업공
학과 (박사수료)

1993년-1998년 한국과학기술연구원 시스템공학연구소
1998년-현재 한국전자통신연구원 연구원
관심분야 : Machine Learning, CAD/CAM, ERP
E-mail : parkhk@etri.re.kr



백 종 명

1983년 고려대학교 산업공학과
(학사)
1999년 고려대학교 전산학과 (석사)
1982년-1991년 삼성전자 과장
1991년-1998년 한국과학기술연구
원 시스템공학연구소

1998년-현재 한국전자통신연구원 정보통합연구팀장
관심분야 : ERP, CALS/EC, CIM
E-mail : jmbaik@etri.re.kr