UML을 기반으로 한 확장된 객체지향 소프트웨어 개발 방법론

○ 최금희 허계범 최영근
광운대학교 컴퓨터과학과

A Methodology of Extended Object-Oriented Software base on UML

Keum-Hee Choi Kwae-Bum Heo Young-Keun Choi
Dept of Computer Science Kawang woon University

요 약

Objectory Process 를 바탕으로 한 Rose 는 실 프로젝트의 개발을 위한 단계에 대한 구체적인 절차제시 및 세부적인 지침의 부족으로 실제 업무 개발 시 혼란을 초래 할 수 있다. 본 논문에서는 Objectory Process 를 확장하여 개발자들을 위한 객체지향 소프트웨어 개발 방법론을 제시한다. 그리고, UML Diagram 과 세부적인 명세서 작성방법 및 산출물을 명시하고, 이와 같은 정보를 저장 및 검색을 관리 할 수 있는 효율적인 객체지향 소프트웨어 개발 방법론을 제시한다.

1. 서론

UML(Unified Modeling Language)은 Booch, Rumbaugh , Jacobson 의 방법들을 통합한 표준 객체지향 방법론이다[1][2][4][5]. 현재 UML 을 지원하는 대표적인 CASE 도구로는 Paradigm Plus, ROSE, SIP/UML 등을 들 수 있다. 그 중에서 UML 을 잘 적용할 수 있는 도구는 Rational Rose 를 들 수 있다. 그러나 Rose 는 CASE 도구의 기능 및 다이어그램 생성과, 기본정보 등을 관리할 수 있으나, 개발자들에게 도움을 줄 수 있는 단계별 지침과 명세서 및 프로세스 정의가 부족하다.

본 논문에서는 UML 을 기반으로 한 확장된 객체지향 생명주기를 제시하고 UML Diagram 에 대한 명세서작성 방법 과 이와 같은 정보를 관리할 수 있는 System 을 구현한다. 본 논문의 구성은 다음과 같다. 2 장에서는 관련 연구와 기존 방법의 문제점에 관하여 살펴보고, 3 장에서는 효율적인 소프트웨어를 개발 할 수 있는 확장된 객체지향 소프트웨어 개발 방법론을 제시하고, 4 장에서는 본 논문에서 제시한 개발 방법론을 적용한 System 구현, 끝으로 5 장에서는 결론 및 향후 연구 방향을 제시한다.

2. 관련 연구

2.1 Rational 사의 Objectory Process

Objectory Process 는 UML 에 기준하여, 프로젝트의 시작부터 종료까지 전체업무를 관리하는 방법을 제시하고 있다[2]. 각 단계는 프로젝트의 영역과 환경 결정을 위한 Inception, 소프트웨어의 구조결정을 위한 Elaboration, 완전한 시스템을 만들기 위한 기능과 특성을 구현하는 Construction, 시스템을 테스트하고 이관하는 Transition 의 4 단계로 구성되어 있다 . Objectory Process 의 주요 특징은 반복 점진적인 관리기법, 아키텍처 중심의 개발 기법, 사용자 중심의 Use-Case Driven 방식을 특징으로 한다[2].

2.2 Rational Rose

Rose 는 Objectory Process 를 기본 생명주기로 소프트웨어 시스템 산출물의 명세 작성, 시각화, 구현, 문서화에 사용할 수 있는 UML 을 바탕한 비주얼 모델링 도구다. 그러나, 실 프로젝트의 개발을 위한 단계에 대한 구체적인 지침과 명세서 부족으로 실제 업무 개발 시 개발자에게 혼란을 초래 할 수 있다.

3. 확장된 객체지향 소프트웨어 개발 방법론

본 논문에서는 그림 1 과 같이 4 단계로 이루어지며 각각의 세부적인 단

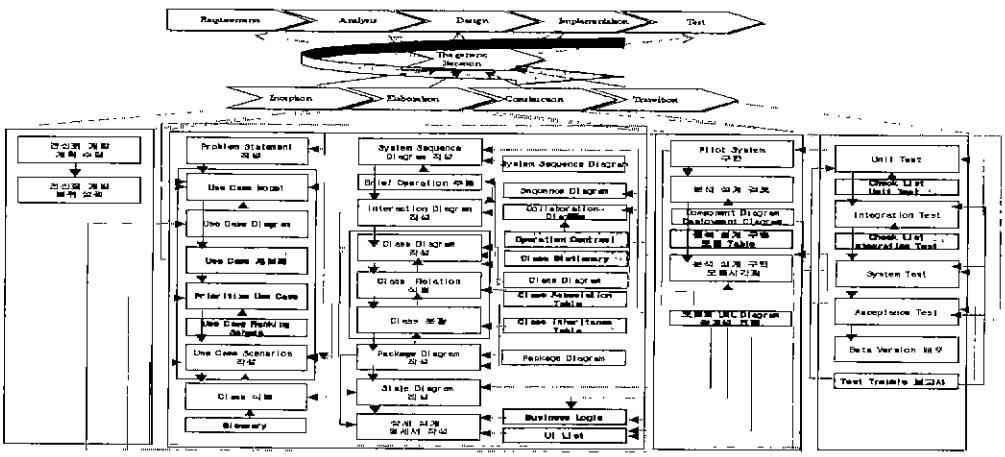

그림 1 객체지향 소프트웨어 개발 생명주기

계는 Objectory 프로세스를 바탕으로 보다 확장 적용한다. 제시절차 및 내용은 표 1과 같다

표 1 본 논문에서 제시 절차 및 내용

유지보수	분석 설계 구현 모든군관을 위한 Table 작성 및 UML Diagram 등 관리 유지(보관) 지식화 Test Trouble List
검증	Unit Test, Integration Test, System Test, Acceptance Test
구현	Service, 분석 설계, 구현 모든 Table 작성
	Use Case Qualities 를 기준으로 프로그램 Pattern 의 원 프레임 형식 구조를 결정하고 4,5 주정도의 Pilot System 을 구현과 분석 설계 검토 및 Component Deployment Diagram 작성
설계	표준 안(Naming 화면 Help)에 따라서 UI 및 Service 를 상세설계사항 작성 한다 결과로 Service 별 분석 설계, 구현 관련정보 관리
	State Diagram 작성을 통한 객체의 행위를 기술
	추출된 Class 를 Package 로 구분하여 Package Diagram 을 작성
	Class Diagram, Class Relation 식별, Class 조직 및 의무를 통해 Class 의 모든 정보를 설정하고 Class Dictionary 작성
	Interaction Diagram 작성을 통해서 Use Case 간의 상호 Operation 파악 및 Sequence, Collaboration Diagram 작성
분석	Class 식별 & 분류 및 System Sequence Diagram 을 통한 Operation 추출을 위한 Contract 추출
	요건정의서(Problem Statement) 작성 Use Case Modeling 을 통한 Use Case 명세서 Scenarios 작성 및 Top Down 방식의 Use Case 세분화 와 Use Case 에 순서화 개념 도입

3.1 세부 분석 및 설계 단계

세부 분석 및 설계단계는 Diagram Modeling 중심으로 설명 한다

3.1.1 Use Case Modeling

이단계에서는 요건 정의서로부터 Use-Case 모델링 통한 전체 Use-Case 기술 내용을 재검토하고 반복한다 산출물로는 Use Case Diagram 과 Use Case 명세서가 있다 표 2 의 명세서는 Use Case 의 개요와 타입 . 사전 사후 조건, Scenarios, Priority 및 Cross Reference Section 으로 구성 되며, 산출물간의 추적이 가능하도록 한다

표 2 Use-Case 명세서

Item	Content							
Use Case 명칭		Type		Quality 점수		순위		
Use Case 개요			Actor List					
조건	Pre Condition				Post Condition			
Flow of Events	Basic Flow of Events				Alternate Flow of Events			
Special Requirements	Special Requirements Content							
	Performance				Security			
Cross Reference								

Scenarios 작성은 표 3 과 같이 구조적측면과 내용적측면을 포함 할 수 있도록 정상적인 흐름과 선택적, 예외적인 측면으로 기술 한다

표 3 Scenario 의 사건 흐름 작성 지침

	Scenario 의 Flow of Event 작성 지침
구조적측면	1 Use-Case 의 시작과 종료 방법을 기술한다
	2 선택적 사건흐름 및 Sub flow 의 순서를 기술한다
	3 고정된 것으로 존재하는 Sub-flow 를 기술하고 고정되지 않는 Sub flow 는 언급하기만 한다
	4 크고 복잡한 Sub-flow 를 기술할 때 sub-section 을 사용한다
	5 명시적인 흐름을 기술하기 위하여 sub-section 을 사용한다
내용적측면	1 Actor 와 Use-Case 사이에 교환되는 정보가 무엇인지 기술한다
	2 중요한 요구 사항이 아니면 사용자 인터페이스의 자세한 사항은 기술하지 않는다
	3 기능성 뿐만 아니라 사건의 진행을 기술한다
	4 그 Use Case 에 속하는 사건만 기술하고 다른 Use-Case 에서 발생하는 것은 기술하지 않는다
	5 예를 들어 "등" "정보"와 같은 애매한 용어는 피한다,
	6 흐름을 자세히 기술한다

3.1.2 Use Case 세분화

Use-Case 는 Business Process 와 같으므로 세분화 될 필요가 있다 이를 단순화 하기 위해서 시스템을 우선 black box 화 하여 Use-Case 를 식별한다 식별된 Use-Case 로부터 하위 레벨로 각 기능을 세분화 하며, 세분화된 Use-Case 들 사이에는 관계(ex uses et .) 및 A-1, A-1 1 과 같은 프로세스 번호를 부여하여 관계성을 표시한다 Use-Case 세분화 기본 지침은 다음과 같다 첫째 개발자 이해를 위해서 한 Sub-System 에 Use-Case 가 10 개 이상인 경우 세분화한다 둘째 문제의 영역이 변경되는 경우 세분화 한다 셋째 문제의 영역이 복잡한 경우 문제의 영역을 세분화 한다 이때 가능하면 더 이상 분해할 필요가 없는 원시기능에 도달 할 때까지 반복수행 한다 세분화는 이해가 용이하고, 식별된 Use-Case 로부터 각 Class, 및 속성 추출이 쉽고, Scenario 작성이 명확한 장점이 있다.

3.1.3 Prioritize Use Case

이 단계에서는 세분화된 Use-Case 에 순서를 부여하며. 높은 순서의 Use-Case 는 초기 개발단계에서 다루어져야 한다 전체적인 전략은 먼저 Core 구조에 명백하게 영향을 미치는 것을 다룰 수 있도록 한다 Use-Case 를 순서화 하기위한 Qualities 및 사유작성은 표 4, 표 5 와 같다.

표 4 Use Case 순위 Schema 1

Use Case	a		b	합계	순위	Justification(사유)
a						
b						

SUM= Σ (항목 가중치 + 항목 점수) / 항목 수

표 5 Use Case Qualities

	점수	Use Case Qualities
내용	a	a 일관된 서비스 요청이나 도메인 레이어에 많은 클래스를 추가하는 것과 같은 구조적 설계에 영향을 미치는 것 영향을 미치는 Class 후보 수 ? □ 10개 이상 □ 9-5개 □ 4-2개 □ 1개 □ 없다
	b	b 위험 시간의 중요성 복잡한 기능을 포함하는 것? □ A □ B □ C □ D □ E
	j	j Use-Case 와 Use-Case 사이의 관련된 Use-Case 수는? □ 10개 이상 □ 9-5개 □ 4-2개 □ 1개 □ 없다
	k	k Use-Case 연관된 Actor 의 역할 비중은? □ 매우 중요 □ 중요 □ 보통 □ 조금 □ 전혀 없다

3.1.4 Interaction Diagram 작성

이 단계에서는 Entity . Boundary , Control 객체들 사이의 행위를 할당하며, Use-Case 에 연관된 객체들 사이에서 발생하는 세부적인 상호작용을 보여준다 그리고 Class 사이에 오퍼레이션들을 분산한다 Interaction Diagram 은 Use-Case 명세서와 System Sequence Diagram(SSD)을 기준으로, 상호작용 객체 시스템을 설계한다 본 논문에서 Responsibility 의 Type 은 GRASP(General Responsibility Assignment Software Patterns)을 기본 원칙으로 적용 한다 추출된 Operation 의 Contract 항목은 표 6 과 같이 작성하며, 기능이 복잡한 경우에는 분해한다

표 6 Operation Contract

	Contract		
Name	Name of Operation Parameters	Type	Type Name(Concept class)
Responsibilities	Operation 이 수행하는 정보		
Cross Reference	System Sequence Diagram Reference Number UC01 1 UC01 3		
	Use-Case Diagram(명세서포함) Reference Number		
Description	Design Note		
Basic Algorithms	계산식 처리방법		
Exception	Exception Case		
Output	Non-UI output Report 등의 System 외부로 보내지는 자료		
Pre-Conditions	Operation 수행 이전 System 기준		
Post-Conditions	Operation 수행 이후 System 기준		

3.1.5 Class Modeling (Class Diagram· CLD)

이 단계의 목적은 실제 필요한 Class 에 대한 모든 정보를 결합 및 설정하는데 있으며, 가장 적게 연관된 클래스로부터 가장 많이 연관된 클래스로 구현한다 산출물로는 CLD, Class Relation Table, Class Dictionary(CDC)가 있다 ,초기에는 개략적 Diagram 작성에서 Class Relation 식별과 분할업무의 반복을 통해서 좀더 상세한 CLD 로 점진적으로 작성한다. Class 간의 Relation 작성 절차는 첫번째 Class 간의 의존성(Generalization /Specialization)은 파악하고, 두 번째 Association 과 Aggregation 을 구분하여 상호작용의 관계를 추출한다 "부분-전체" 계층을 표현하기 위해서 Gamma 의 "Composite" 패턴을 적용한다[3] 세 번째는 요건정의서 및 SSD 로부터 지시된 행동, 의사소통, 소유관계를 포함하도록 작성 한다 CLD 작성 정보를 통해서 CDC 를 작성하며, CDC 정보는 표 7 과 같이 Class 의 기본정보와 발생주기, 보관주기, Attribute, Operation 등을 작성하며, Package 정보는 다음 단계를 통해서 추가된다 작성된 CLD 는 Core Class 누락, 불필요한 Class 존재, 중요 관계누락, 불필요한 관계존재, 관계가 잘못 선정된 경우, 속성이 잘못 선정된 경우 등을 제외하고 지나치게 일반적이거나 편협적인 관계가 설정 되었는지를 검토 한다.

표 7 Class Dictionary(CDC)

Class Content					
Package 명칭		Name			
Type		Rule Name			
Super Class Ref		Sub Class Ref			
Class Description					
발생주기		보관주기			
Cross Reference					
Attribute Property	Name		Type		
	Visibility			Initial value	
	Description				
Operation Property	Name		Visibility		
	Parameter , N				
	Return Value Type				

3.1.6 상세 설계 명세서 작성

이 단계는 Use-Case 기준으로 UI(User Interface) List 및 UI Service 를 결정 하는 업무다 먼저 Use-Case, CLD 로부터 Observer 의 형식인 Model-View-Controller 패턴을 적용하여 최종 UI 식별한다 구분된 내용을 통해서 UI 의 표준 안을 결정 하고, UI 에 따른 서비스의 종류와 내용을 결정 한다 Service 는 실 구현을 위한 내용으로 표 8 과 같다 표준 안은

일관된 사용패턴 유지, 사용상의 오류 최소화, 개발자의 생산성 향상, 일관된 구현패턴을 유기할 수 있는 장점이 있다

표 8 Service, 분석, 설계, 구현 모듈 Table

Service Content						
Name		Type		UI Name		
Service Description	Purpose		Owner			
	Create Date		Update Date			
	Parameter 1..N					
	Return Value Type		Reference Type			
	사용방법					
	Response Type		Response Time			
	Description					
Cross Reference	Use-Case Diagram					
	Class 1			Class N		
	System Sequence Diagram					
	Interaction Diagram					
	Class Diagram					
Service Location	tuxedo sales pl src pa pbl					
Component Name	Pa dll					
Component Location	tuxedo sales pl bin pa dll					
Package Name						
C/S 구분	Server					

3.2 구현 단계

Pilot System 구현 단계는 Use Case 내용 중 우선순위가 높은 내용을 먼저 작성한다 이전 단계를 통해서 Program 레벨에서의 표준 안, 기본구조, 상속구조, 사용 Event 및 추상화를 결정한다 그리고, 업무를 기초로 분석 설계, 구현을 검토하고 Component, Deployment Diagram 및 분석 설계 모듈 Table 을 작성 한다 작성된 Table 은 표 9 의 내용을 포함한다.

3.3 검증 단계

검증단계는 첫째 단위테스트로 코딩에 대한 최소한의 테스트이다 둘째 결합테스트는 Sub-System 내에서 각 모듈과 모듈사이의 Interface 에 대한 테스트이다 셋째 시스템테스트는 요건정의의 내용을 기능상 질성적으로 충족시켰는지를 테스트한다. 넷째 인수테스트는 모든 흐름이 기능 및 성능상의 측면에서 제대로 수행되는 지를 테스트 한다 작성된 결과표는 분석 설계의 결과 및 프로그래머의 평가에 기본 자료가 된다.

3.4 분석, 설계 정보 저장 및 검색 방법

본 논문에서 산출된 정보는 모듈별로 DB 화하여 저장 및 검색 기능하도록 하였다 표 9 에서는 소프트웨어 개발 생명주기에서 산출된 정보들간의 결합된 관계를 나타내고 있으며 유지보수 및 재사용을 용의하게 한다

표 9 분석, 설계 구현 검증 정보 조회항목

① 분석	Use Case Name	자원처리	UI Name	Service Name	Data0[?]01_hs
		A 010-DALA01D	W_loan010	Service Type	Bat
	Type		Actor List	자원담당자 회계 System, 요청처리 System, 결과 System	
	요건정의서	A 010-PL01			
② 설계	Cross Reference	Use Case Diagram Name		A 010-DAL A01	
		System Sequence Diagram Name		D-010-DAL A01	
		Sequence Diagram Name		D 030-S DAL A01	
		Collaboration Diagram Name		D-030-C DAL A01	
		Class Diagram Name		D-040 DAL A01	
		Class List			
		자원담당	자원정서	스케줄	신청정서
		<<Entity>>	<<Entity>>	<<Entity>>	<<Entity>>
③ 상세시계 및 구현	Service Description	Owner	Create Date	Modifier	Update Date
		Mafa	1995.06.15	Informix	1999 07 22
		Purpose	자원스케줄 자동 생성		
		Parameter	스케줄 startdate, enddate 조건 Object, 전급		
		Return	스케줄 Object		
		사용방법	GetSchedule(스케줄 startdate enddate 조건 Object 전급)		
		Description	미제 = 원금 * 일수 * 이자율 /36500 일수 = DaysAfter(Start Date * 조건 Object 기준조건 Start, Start Date * 조건 Object 기준조건 End) . .		
		Service 구분	Server	Location	/tuxedo/sales/pl/src
	Package	Name	자원관리		
	Component	Name	PA DLL	Location	/tuxedo/sales/pl/bin
④ 검증	Test Check List				
	화면,보고서 이미지 및 운영방법(15%)	시스템기능의 전실(60%)	안정% (20%)	성능 (5%)	결과
	B	A	A	D	B

4. 구현

본 장에서는 제시한 소프트웨어 개발 생명주기에 따른 개발 프로세스 적용과 분석 설계 정보를 저장 할 수 있는 실제 System 을 구현하고, 논문에서 제시한 방법과 Rational Objectory Process 와 비교 분석한다

4.1 제시환경

본 논문의 적용 System 환경은 OS 는 HP-UX B 10, Informix 7 24 로 구축 한다 개발 언어는 Power Builder 6 5 와 WatCom C++을 사용한다

4.2 적용

다음의 결과들은 본 논문에서 제안한 방법들을 실제 시스템에 적용한 단계별 정보들이다

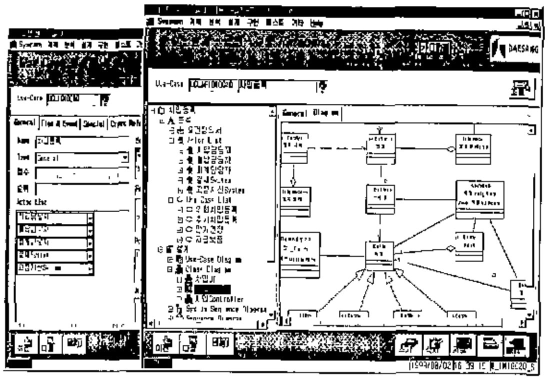

그림 5_실행화면

4.3 비교분석

표 10 은 본 논문에서 제시한 객체지향 소프트웨어 방법론과 Rose 의 Objectory 와 비교한다 기존의 Objectory 에서는 구체적인 개발 단계 및 개발자들에게 도움을 줄 수 있는 체계적인 지침과 프로세스가 부족하다

표 12 개발 프로세스 간의 비교

항목	ROP	PRC	항목	ROP	PRC
Problem Statement	◇	◆	Package Diagram	◆	◆
Use Case Diagram	◆	◆	State Diagram	◆	◆
Actor List	◆	◆	Service Logic 명세서	◇	◆
Use Case 명세서	◆	◆	UI List	◇	◆
Use Case Priority Table	◇	◆	UI 표준 안	◇	◆
System Sequence Diagram	◆	◆	Component Diagram	◆	◆
Operation Contract	◆	◆	Deployment Diagram	◆	◆
Class Diagram	◆	◆	Code 자동 생성	◆	◇
Glossary Table	◆	◆	역공학 지원	◆	◇
Class Dictionary	◆	◆	모듈별 단계식 표현 Table	◆	◆
Sequence Diagram	◆	◆	Test Check List	◇	◆
Collaboration Diagram	◆	◆	Test Trouble 보고서	◇	◆

◆ 지원 ◆ 일부 지원 ◇ 지원 하지 않음 ROP (Rose Objectory Process) PRC (본 논문에서 제시한 Process)

5. 결론

본 논문에서 제시한 방법론은 UML 을 기반으로 개발자들에게 유용한 기법을 제공하고, 재사용을 용이하게 하였으며, 개발 완료 후 유지보수가 용이한 장점이 있다 그러나 자동적 프로그램을 생성할 수 있는 완전한 CASE TOOL 기능을 지원하지 못하고 있다 앞으로의 향후 연구 과제로는 Code 자동생성, 역공학 지원 및 Diagram 자동 생성, Auto Trace 관리 가능한 CASE TOOL 개발을 위한 연구가 필요 하다

6. 참고문헌

[1] UML Group, Unified Modeling Language 1 1, Rational, 1997

[2] Jacobson,Booch,Rumbaugh, THE UNIFIED SOFTWARE DEVELOPMENT PROCESS, Addison-Wesley, 1999

[3] E Gamma, R Helm. R Johnson and J Vlissides, "Design Patterns Elements of Reusable Object-Oriented Software," Addison-Wesley, 1995

[4] Craig Larman, 'Applying Uml and Patterns", Prentice Hall, 1997

[5] 김수동, "실무자를 위한 소프트웨어 공학", 에드텍, 1999

[6] 최영근, 허계범 "객체지향 소프트웨어 공학,"도서출판 한국 실리콘, 1998