韓國 컴퓨터情報學會 論文誌
第11卷 第6號, 2006. 12

2006-11-6-1-4

객체모델을 이용한 컴포넌트 설계 및 검색 프로토타입

임 근*

Component Design and Retrieval ProtoType Using Object Model

Lim Keun*

요 약

본 논문에서는 컴포넌트 설계와 이를 기반으로 한 검색 프로토타입을 제시하였다. 재사용 단위로 단일 객체만을 대상으로 하게 되면 연관관계가 있는 객체간 상호작용을 이해하기 어렵고 관련성 유지가 어렵다. 따라서 본 논문에서는 재사용 단위로 클러스터링 컴포넌트 제안하였다. 이것은 객체간 관련성과 연관관계를 부여하여 사용자가 컴포넌트를 이해하고 검색할 수 있도록 하였다. 제안한 프로토타입 시스템은 애매한 검색 정의에 대하여 3가지 검색 환경을 제시하여 정확한 검색을 가능하게 하였다.

Abstract

In this thesis, we presents the component design and retrieval prototype system which can retrieve the most appropriate component for reuse system. If it used only an abject to reuse unit, it can not understand mutual reaction and can not sustain relationship between objects. Therefore, we porposed clustering component for reuse unit. It makes to understand relationship and reaction between object, so User can retrieve the proper component for reuse system. And porposed retrieval prototype system can select the correct things, it provide 3 facet retrieval environment against ambiguous retrieval definition.

▸ Keyword : 객체모델링(Object Modeling), 컴포넌트 기반 개발(Component Based Developement), 정보검색(Information Retrieval)

• 제1저자 : 임 근
• 접수일 : 2006.10.26, 심사일 : 2006.12.12, 심사완료일 : 2006. 12.22
* 서울보건대학 컴퓨터정보과교수

Ⅰ. 서론

객체지향 방법에서 주체가 되는 객체 모델링 방법은 비정화된 방법을 사용하기 때문에 표현의 모호성과 의미의 부정확성을 내포하고 있기 때문에 모델에 대한 자의적 해석이 문제가 된다[1]. 따라서 이러한 비정형성으로 인하여 모델간의 일치성 및 완전성의 보장이 어렵다. 또한 객체지향 방법에서의 객체는 고유한 특성을 가지고 있지만, 각 객체가 적용되는 시스템의 범위나 특성에 따라 해당 시스템에서의 역할이 달라질 수 있다. 즉 객체지향 방법에 의해 개발된 소프트웨어 컴포넌트의 재사용성 이전에 재사용 가능 컴포넌트에 대한 신뢰성 문제를 해결해야 한다[2]. 따라서 본 논문에서는 재사용 컴포넌트가 되는 객체나 클래스 모델에 대하여 완전성 여부를 확인하기 위해 형식 명세언어로 변환하고, 정형화 형식명세 언어의 특성에 의해 검증된 객체를 대상으로 재사용 컴포넌트를 설계하고 이것을 프로토타입 시스템에 적용하여 검색 과정을 수행하였다. 본 논문의 구성은 2장에서 기반연구 3장에서 컴포넌트 설계 4장에서 검색 프로토타입 및 5장 비교평가 6장에서 결론을 기술한다.

Ⅱ. 관련연구

2.1 클러스터링 방법

자료집합의 항목을 클러스터하기 위해 항목사이의 연관정도를 정형화할 방법이 필요하고 여기에는 거리측정 또는 유사도나 차이점에 대한 측정이 해당된다[3,4]. 클러스터링에 기초한 검색에서 각 객체간의 유사도 결정은 각 객체를 특징짓는 색인된 용어에 할당된 가중치면에서 보면 객체 표현과 선택된 유사도 계수에 근거한다. 클러스터에 기반한 검색에서 유사성 계수에 대한 결과는 문헌벡터의 길이에 의해 정규화된 측정을 사용하는 것이 중요하다.

2.2 인덱싱 자동화 방법 고려

기존 정보검색 방법에서는 대부분 분석 작업과 인덱스 생성과정이 해당 전문가의 수작업에 의해 구성되고 있다[5]. 그러나 소프트웨어 라이브러리를 구성하는 컴포넌트에 대한 인덱싱은 프로그래머의 지식에만 의존하는 것보다 방법에 의한 인덱싱이 바람직하다[6]. 즉 컴포넌트 인덱싱 과정은 컴포넌트 작성자가 제공하게 될 컴포넌트 개발시 기능, 사용법 등에 대한 기술을 담고 있는 문서를 기반으로 자동 분석하는 방법이 바람직하다. 인덱싱을 수작업으로 하는 것보다는 자동화된 인덱싱 방법이 갖는 장점은 비용 뿐만 아니라 일관된 방법의 사용으로 인덱스들이 호환성을 갖게 된다.

2.3 인덱싱 항목의 추출

컴포넌트의 특성을 대표할 수 있는 인덱스 항목의 추출을 위해 컴포넌트의 기능을 기술한 문서와 주석의 구문적 분석을 통해 각 인덱스 항을 결정하고 빈도수를 파악한다[7]. 객체지향 라이브러리를 구성하는 클래스 컴포넌트는 고유의 특성에 의해 자료부분과 자료 관리에 필요한 오퍼레이션 부분으로 구성되므로 클래스를 특성짓는 인덱스를 정의한다[8,9].

Ⅲ. 재사용 컴포넌트의 설계

3.1 컴포넌트간 관계

하나의 시스템을 이루기 위해서는 여러 개의 객체가 서로간의 관계를 갖고 참여한다[10]. 이들간의 관련성은 현재 구성되어 있는 시스템 내에서는 중요한 의미를 갖는다[11]. 이것은 여러 개의 객체가 모인 하나의 그룹화된 컴포넌트로 볼 수 있도록 만들어 준다. 따라서 하나의 컴포넌트를 구성하면서 내부적으로는 밀접한 객체간의 관계가 표현되어 있으며, 외부적으로는 상세한 내용이 추상화되어 있어 단위 컴포넌트로서의 구조를 구성한다[12].

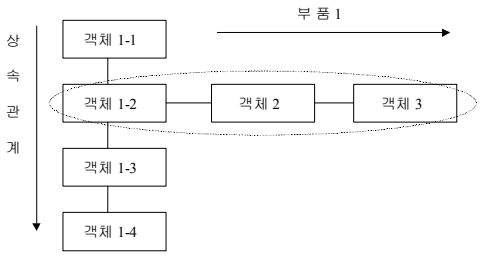

그림 1. 객체의 상속 관계와 컴포넌트
Fig. 1 Inheritance relation and Component of Object

<그림 1>은 객체들간의 관련성을 나타낸다. 하나의 객체는 상속 관계로 수직적으로 표시하고, 하나의 컴포넌트를

이루는 객체들은 수평적으로 표시한다. 하나의 컴포넌트에 참여하고 있는 다른 객체들 역시 또 다른 상속 관계에 속하고 있는 상세화된 객체의 하나이다.

컴포넌트1에 속한 3개의 객체 1-2, 2, 3은 컴포넌트 내에서의 관련성으로 묶여진다. 그 관련성은 association, aggregation, inheritance 등이 될 수 있다.

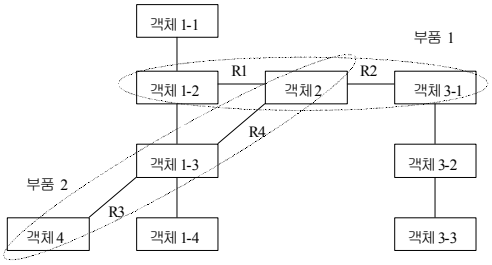

그림 2. 객체와 컴포넌트
Fig. 2 Object and Component

<그림 2>에서와 같이 컴포넌트에 참여하고 있는 객체들은 각각 다른 컴포넌트에 참여할 수 있고, 그 객체는 또 다른 상속 관계에 참여한다. 컴포넌트 1의 관련성 R1, R2와 컴포넌트 2의 관련성 R3, R4는 association, aggregation, inheritance 관련성의 조합으로 이루어지며, 하나의 객체는 그 객체가 사용되는 용도에 따라 다른 모습을 갖게 된다.

ObjectID	Inheritefrom	Inheriteto	Componet

그림 3. 객체의 확장 특성
Fig. 3 Extension feature of Object

Attributes나 Behavior는 객체의 내부 속성이면서, 그 내용이 변하지 않는 정적인 내용이다. 따라서 그 내용 자체가 라이브러리에 직접 등록되어 있어야 할 필요는 없다. 다만, 객체의 모습을 사용자로부터 이해시키는 이해지원 부분에서 요구된다. 상속 관계를 가진 객체들은 자신의 기본 클래스와 파생 클래스를 찾기 위해 이 속성을 이용할 수 있다. 객체들의 상속 관계는 <그림 4>와 같다. 상속 관계를 통해 객체는 상세화되어 있으며, 하나의 대표이름으로 객체가 선택되어도 그 하위 객체들까지 검색을 할 수 있다.

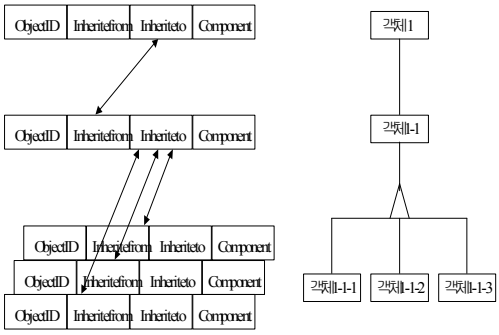

그림 4. 객체간 상속 관계
Fig. 4 Inheritance of Between objects

객체들은 컴포넌트의 목록을 가지고 있으므로, 자신이 속한 컴포넌트를 잘 알고 있다. 이 객체는 Inheriteto속성을 이용하여 상세화된 하위의 객체들까지 검색의 대상이 될 수 있다. 하나의 객체에 대하여 객체의 내부 속성에 대한 정의는 유일하지만, 외부 속성에 대한 정의는 하나 이상일 수 있다.

3.2 컴포넌트의 구조

컴포넌트를 구성하고 있는 객체들은 관련성을 가지고 있다. 컴포넌트는 참여하고 있는 객체들뿐만 아니라, 그들간의 관련성을 표현해 주어야 한다.

RelationID	Description	Object1	Card	Object2	Card	..	..

그림 5. 객체간 관련성
Fig. 5 Relation of Between objects

<그림 5>는 객체간의 관련성을 표시하기 위한 구조이다. 객체들 간의 관련성은 일반적으로 두 개(binary association), 혹은 세 개(ternary association)의 객체들 사이에서 이루어진다. 관련성은 그 이름을 가지고 있으며, 일대다, 다대다 관계를 표현하기 위해 관련 객체간의 대응된 Cardinality 속성을 가지고 있고, 관련이 없는 객체의 Card는 0이다. Object1, Object2는 앞에서 설명한 ObjectID를 의미한다. 컴포넌트의 정의<그림 6>는 해당 컴포넌트를 이루고 있는 관련성만을 나타냄으로서 전체 시스템의 모든 내용을 포함할 수 있다. 이런 컴포넌트의 구조는 컴포넌트에 참여하고 있는 관련성으로 부터 객체나 컴포넌트를 찾을 수 있다.

ComponentID	Description	RelationID1	...	RelationIDn

그림 6. 컴포넌트 구조
Fig. 6 Component structure

객체로부터 컴포넌트를 찾는 구조뿐만 아니라 하나의 컴포넌트에 참여하고 있는 객체들과 다른 객체들간의 관련성을 잘 나타낼 수 있다.

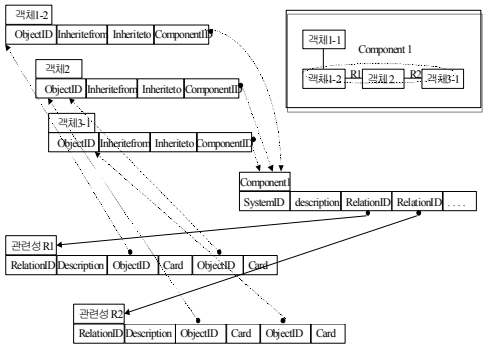

그림 7. 컴포넌트를 구성하는 객체간 관계
Fig. 7 Relation Between objects of Component

3.3 컴포넌트의 클러스터링

사용자가 라이브러리를 사용하고자 할 때 먼저 원하는 컴포넌트를 갖는 라이브러리로 접근하여 라이브러리 시스템이 이해하는 단어들로 원하는 컴포넌트에 대한 기술을 한다. 정보 검색에 있어 분류의 기본 개념은 두 문서사이의 유사성을 추출하는 방법으로 <그림 8>과 같다.

$$SIM(DOC_i, DOC_j) = 2| P_i \cap P_j | / | P_i | + | P_j |$$
$$SIM(DOC_i, DOC_j) = | P_i \cap P_j | / | P_i \cup P_j |$$
$$SIM(DOC_i, DOC_j) = | P_i \cap P_j | / | P_i | \times | P_j |$$

P_i : DOC_i 의 인덱스 집합
p_j : DOC_j 의 인덱스 집합

그림 8. 분류를 위한 유사도 측정 방법
Fig. 8 Similality measure for Classify

$$SIM(COMPi, COMPj) = \frac{\sum_{k=1}^{n} (TERMik \times TERMik)}{\sum_{k=1}^{n} TERMik + \sum_{k=1}^{n} TERMjk - \sum_{k=1}^{n} (TERMik \times TERMjk)}$$

그림 9. 컴포넌트간 기능적 유사도 측정
Fig. 9 Functional Similality measure between components

<그림 8>에 나타난 여러 방법은 인덱스를 집합으로 보고 집합의 기본 연산인 집합의 원소를 구하는 | Pi | 나 | Pi ∩ Pj |, | Pi ∪ Pj | 들을 사용한다. 그러나 본 논문에서는 인덱스를 벡터 모델로 표현하고 벡터 모델로 표현된 값들은 인덱스가 아닌 인덱스의 기능적 유사도를 통하여 각 개체를 클러스터링 한다. 따라서 그림<그림 9>과 같이 컴포넌트간 기능적 유사도에 의한 측정 한다. 정확한 검색을 위하여 각 컴포넌트의 분류 구조를 3개 facet, 즉 Relation facet, Operation facet, Domain facet으로 분류하였다.

- Relation facet : 컴포넌트가 다른 컴포넌트에 어떻게 관련이 있는지를 보인다. 예를 들면 하나의 컴포넌트는 다른 컴포넌트에 의해 호출되거나 호출할 수 있다.
- Operation facet : 이 facet은 컴포넌트 구현시 프로세스 종류, 즉 각종 기능에 관련된 내용을 포함하고 있다.
- Domain facet : 개발자는 일반적인 객체에 의해 소프트웨어 대상을 분류함으로서 그 대상을 이용할 수 있으며, 대규모 어플리케이션에서 소프트웨어 컴포넌트는 특정한 영역 오퍼레이션을 지원한다.

다음은 은행의 대출업무를Use Case별로 구성하여 각 단계마다 진행되는 영역별 과정을 3개 facet으로 구분한 예이다.

표 1. 분류 스키마
Table 1 Classified schema

	Classifier	Description
Relation	CALLER	다른 component 호출
	INTERACTION	VIEW, WINDOW 등 객체
	LEAF	종단 노드
	ROOT	계층구조의 시작노드
	CALLEE	다른 component에 호출
Operation	SECURITY	페스워드 체크
	CALCULATION	수치값 계산과정
	CLIENT/SERVER	통신 프로토콜
	EXCEPTION	예외처리
	DISPLAY	스크린상 표시
	SQL	데이터베이스 동작
	THREAD	객체 링크
	SEARCH	탐색
	UTILITY	일반 기능성
Domain	COMM_REC	comercial recomend
	CONTACT	업무 협조
	CUSTOMER	구매 회사
	FINANCIAL	재정기구
	GENERAL	모든영역에 적용
	PARTNER	파트너 쉽

표 2. Object의 분류
Table 2 Classified Object

Object	Relation	Opertion	Domain
ACCOUNT-LINK#1	INTERACTION	DISPLAY, SEARCH	CUSTOMER FINANCIAL
APROV-FRONT-CHK	CALLER	SECURITY	CUSTOMER
COMMENT-DETAIL	CALLER	DISPLAY,UTILITY, THREAD	CUSTOMER
CUST-INFO-UPDATE	INTERACTION	DISPLAY, RETRIEVE	CUSTOMER
CUST-PROD-1	INTERACTION	DISPLAY	CUSTOMER
...			
CUST-DELETE	CALLEE	DISPLAY	CUSTOMER
FIN1-INFO	LEAF	DISPLAY	CUSTOMER
FINU1-RETRIEVE	LEAF	DISPLAY	FINANCIAL
PRODUCT-DETAIL	CALLEE	DISPLAY	FINANCIAL
.....			

VI. 검색을 위한 프로토타입 시스템

검색 과정은 <그림 10>과 같이 3개 list-box로 부터 facet value를 선택하면서 시작한다. 각 list-box는 앞선 언급한 3개 분류 범위중 하나와 대응한다. 범위를 만족하는 모든 컴포넌트의 이름을 <그림 11>과 같이 나타낸다. 분류 box를 통해 찾아진 컴포넌트를 사용자가 자세하게 정의할 수 있도록 한다. 즉 도메인 facet의 값을 선택하고 전,후의 네비게이션을 선택하므로서 개발자는 리스트로부터 이전 또는 이후의 컴포넌트를 볼 수 있다.

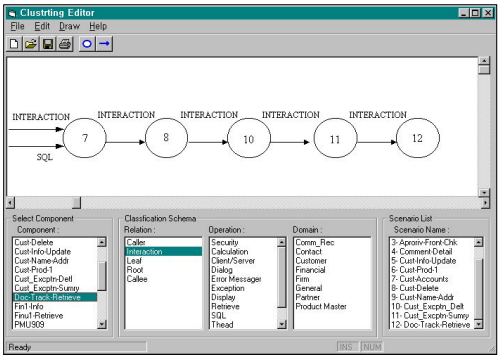

그림 10. list box의 구성
Fig.10 List box construction

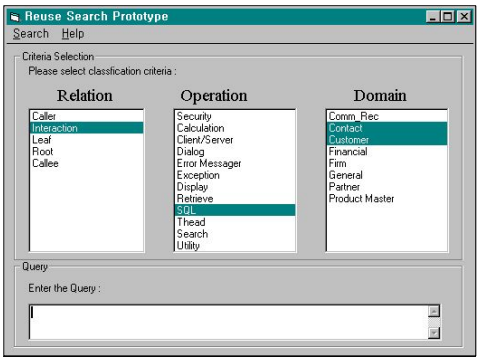

그림 11. 객체 모델과 컴포넌트 구조
Fig.11 Object Model and Component structure

<그림 11>은 개발자의 입장에서 생성된 모델의 구조와 각 모델이 하나의 클러스터링된 컴포넌트로서의 관계를 보여주기 위한 화면이다.

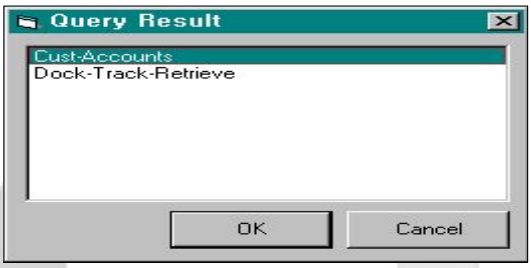

그림 12. 질의 결과 화면
Fig.12 Query Result

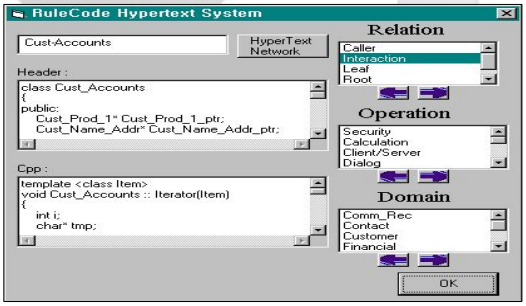

그림 13. 검색된 컴포넌트에 대한 소스코드 및 구성
Fig.13 Source Code and feature

<그림 12>은 list box를 통해 입력한 정보에 대한 질의 결과이다. 본 논문의 결과는 재사용을 위해 구성된 클러스터 컴포넌트의 결합되는 과정을 보여주므로서 재사용시의 이해는 물론 정확한 컴포넌트를 선택, 사용할 수 있다.

<그림 13>은 검색 결과 컴포넌트의 소스코드 공개부분으

로 해당 컴포넌트의 구조를 이해할 수 있도록 한다. 즉 결과로 나타난 컴포넌트의 클러스터 과정에 의해 객체간의 전, 후 관련성을 보여준다. 이를 기반으로 클러스터된 컴포넌트의 확장 정보와 구조 이해를 가능하도록 지원할 수 있다.

V. 평가

표 3. 기존 재사용 단위 및 성능비교
Table 3 Existing Reuse unit and Performance compare

	코드 단위	OMT	디자인 패턴	본 논문 방법
재사용성	-모든 코드에 대해 재분석이 필요하다 -원하는 코드를 얻기가 어렵다	-원하는 객체모델을 찾기가 어렵다. -찾아진 객체모델들을 결합하기가 어렵다.	-제공되는 패턴이 제한적이다. -상세한 구현이 부족하다.	-클러스터링을 통하여 재사용 영역을 확장할 수 있다. -재사용의 단위를 다양화할 수 있다.
유지보수성	-모든 코드에 대한재분석 및 재설계 필요 -유지보수성 낮음	-유지보수는 용이하나 모델들의 융통성이 부족하다.	-제공되는 패턴이외에는 유지보수성이 낮다.	-검증을 통한 컴포넌트를 사용하므로 유지보수가 용이하다.
영역 적용성	-해당 영역에 적합한 코드를 찾기 어렵다.	-작성된 객체모델을 적용할 수 있는 영역은 제한적이다.	-기본 패턴에 대한 적용만 가능 -많은 영역을 지원하지 못하다.	-재사용 단위의 확장이 가능하므로 다양한 영역에 적용할 수 있다.
이해 용이성	-코드 수준의 이해가 용이하지 않다.	-모델 자체는 이해하기 용이하나 상세한 정보는 제공하지 않는다.	-패턴에 관한 문서를 통해 제공	-객체간 관련성을 제공하므로 컴포넌트의 이해가 용이하다.

본 논문에서 제시한 재사용 방법은 재사용율을 높이기 위해 제안된 방법으로 재사용 단계를 볼 때 기존의 코드 단위의 재사용에 비하여 높은 재사용율과 비용 절감 효과를 가져올 수 있다.

재사용의 단위에서 볼 때 객체 단위의 재사용이나 디자인 패턴을 이용한 재사용에 비해 개발 생산성이나 유지보수성 및 영역 적용성과 사용 용이성등이 우수한 것으로 나타났다. 앞에서 설명한 재사용의 단위와 본 논문에서 제안한 방법을 다음과 같이 비교하였다

<표 3>과 같이 코드 단위의 재사용은 재사용하려는 시스템의 특성에 부합하는 코드를 찾기가 어렵고, 찾아진 코드에 변경이 필요할 경우 부가적인 과정을 수반하는 문제점을 가지고 있다. 디자인 패턴의 경우는 제한된 재사용성과 기본적으로 제공하고 있는 환경이외에서는 재사용의 적용이 한정적인 문제점을 내포하고 있다. 특히 제안된 컴포넌트의 구조는 컴포넌트간에 정의된 관계를 통하여 연관 컴포넌트나 클러스터링 컴포넌트 목록을 쉽게 볼 수 있어 사용자로부터 컴포넌트와 관련컴포넌트의 이해를 높일수 있다는 장점이 있다.

VI. 결론

재사용의 단위를 독립된 객체만으로 구성하게 되면 시스템 구성상 객체간의 상호작용을 이해하기 어렵고 각 객체들간 관련성 유지가 용이하지 않다. 따라서 검증된 객체들은 구축될 시스템의 특성에 따라 객체간의 관계를 가지게 되며 이러한 관련성을 기반으로한 객체를 하나의 클러스터 컴포넌트로 구성하여 소프트웨어 구축시 재사용 단위로 하였다. 재사용을 위한 검색 과정에서는 가장 적합한 재사용 대상의 컴포넌트를 찾기 위해서 단계적 검색과정을 제공하는 프로토타입을 구현하였다. 검색 범위가 애매하게 정의되었을 경우에 대하여 3가지 facet을 기반으로 가장 적합한 컴포넌트를 검색할 수 있도록 하였다. 본 논문의 경우는 클러스터 단계에 따라 적용 영역을 확대할 수 있는 장점을 갖는다. 특히 제안된 컴포넌트의 구조는 컴포넌트간에 정의된 관계를 통하여 연관 컴포넌트나 클러스터링 컴포넌트 목록을 쉽게 볼 수 있어 사용자로부터 컴포넌트와 관련 컴포넌트의 이해를 높일수 있다는 장점이 있다.

참고문헌

[1] Charlie Alfred and Stephen J. Mellor, Observation on the Role of Patterns in Object-Oriented Software Development, Object Magazine, 61-65, May 1995.

[2] D. J. Chen, "A Model Driven Approach to

Accessing Managerial Information: The Development of a Repository-Based Executive Information System," J.Management Information Systems, vol. 11, no. 4, pp.33-66, Spring 1995

[3] Stephen J. Mellor and Ralph Johnson, Why Explore Object Methods, Patterns, and Architectures, IEEE Software, 14(1), 27-30, January 1997.

[4] Michael J. Port, software Engineering with C++ and CASE Tools, Addison-Wesley, 1996.

[5] Arthur J. riel, Object-Oriented Design Heuristics, Addison-Wesley Publishing Company, 1996.

[6] Sally Shlaer and Stephen J. Mellor, Recursive Design of an Application-Independent Architecture, IEEE Software, 14(1), 61-72, January 1997.

[7] Stefan Sigfried, Understanding Object-Oriented Software Engineering, IEEE Press, 1996.

[8] 이경환, 소프트웨어 재사용을 위한 객체모델링 기법, 교학사, 1993.

[9] 임 근외, "객체의 동적, 정적 모델로 부터 객체 구현을 중심으로한 객체지향 소프트웨어 개발에 관한 연구", 한국정보과학회지, vol. 20-1, 533-536, 93.

[10] Lee seon Keun. " 효율적인 정보검출을 위한 NIDS 시스템 설계에 관한 연구", 한국컴퓨터정보학회 논문지, 제8권 제3호, pp.156-162, 2003.

[11] 채원석, "SMIL 문서 편집기 개발을 위한 객체모델링", 한국컴퓨터정보학회 논문지, 제10권 제3호, pp.161-171, 2005

[12] 임 근외. " 객체모델에 대한 형식명세로의 변환방법", 한국컴퓨터정보학회 논문지, 제8권 제4호, pp.21-27, 2003.

저 자 소 개

임 근

1992. 3- 현재
서울보건대학 컴퓨터정보과 교수
<관심분야> S/W 공학, 객체지향방법론, 정보검색 등