# 확장된 개체 개념의 비즈니스 시스템 분석 모델
## (A Business System Analysis Model with Extended Entity Concept)

이 서 정 † 고 병 선 †† 최 미 숙 ††† 박 재 년 ††††
(Seojeong Lee) (Byungsun Ko) (Misook Choi) (Jainyun Park)

**요 약** 기존의 시스템 분석 모델은 설계와의 일관성 유지를 위해 개체들 간의 관계설정과 이벤트의 흐름을 표현하는데 다양한 방법을 제시하고 있다. 그러나, 시스템 내의 개체의 식별보다는 시나리오를 바탕으로 한 시스템의 흐름을 중심으로 그에 관련된 개체를 도출하는데 초점을 맞추고 있다. 시스템에서 개체들을 체계적으로 정의하고 구축하는 작업은 소프트웨어 개발에서 기초적이면서도 매우 중요한 작업이며, 구축된 개체는 시스템의 중요한 재산이 될 수 있다. 특히, 비즈니스 시스템의 경우, 비즈니스 규칙이나 수강업무의 출석부와 같이 두 가지 이상의 개체에서 계산되거나 유도된 정보는 매우 중요한 시스템의 재산이 될 수 있다. 그리고, 이러한 정보들을 관리하는 정보 즉 메타 정보 또한 시스템의 중요한 재산(asset)이 된다.

본 연구는 시스템의 중요한 재산이 될 수 있는 개체 정보를 도출할 수 있는 구분 기준을 도입한 시스템 분석 모델을 제안한다. 이 모델을 통해 시스템은 개체, 인터페이스, 이벤트 또는 행위의 세 가지 부분으로 분석되며, 그 중 시스템의 개체는 독립개체와 그에 관련된 종속개체 및 그에 적용될 수 있는 비즈니스규칙을 포함하는 제약조건개체를 도출하며, 이벤트의 물리적 또는 행정적 관리사항은 관리개체로 표현할 수 있다. 다양한 방식의 개체 식별은 분석과정에서 누락되는 개체를 줄일 수 있다.

***Abstract*** Existing system analysis models suggest various ideas to present entity relations and event flows for consistency between analysis and design paradigms. However, they are preferred to derive and arrange related entities on system flow than to identify entities. To identify entities systematically is a basic and important work of software development, and identified entities can be major assets of business system. In case of business systems, the business rules or the computed or derived information like attendance lists of lecture system can be the most important system assets. The management information or meta data are also.

In this paper, we suggest a business system analysis model to derive and present entities. System is identified entities, interfaces and event or behaviors through this model, then entities are extended to independent entities, dependent entities which are dependent to independent entities, constraint entities which present the rules of independent or independent entities and management entities which shows the physical and administrative notices. Various entity identification can reduce the incompleteness of entity analysis.

## 1. 서 론

시스템 개발 과정 중 분석 단계는 이후의 설계, 구현 및 유지 보수 단계를 지원하기 위한 가장 중요한 작업이다. 또한, 시스템 사용자가 가장 많이 참여할 수 있는 단계이기도 하다. 즉, 분석 단계의 주요 작업은 시스템 내의 정보, 사용자의 요구, 그 처리 과정 및 현재 사용하고 있는 시스템의 상태와 관리사항 등을 분석해 내는 것이다. 그 중, 시스템 내의 정보는 시스템의 흐름이나 외부 환경의 변화에 크게 영향을 받지 않는 정적(static)인 특성과 상대적으로 동적(dynamic)인 특성으로 구성되어 있다[1]. 정적인 정보는 시스템의 사용자, 시스템이 제공하는 제품/서비스, 시설, 비즈니스 규칙 등의 정

· 이 논문은 "2001년 숙명여대의 교비과제"에 의해 연구된 것임.
† 비 회 원 : 동덕여대 교양교직학부 교수
sjlee@dongduk.ac.kr
†† 학생회원 : 숙명여자대학교 컴퓨터과학과
kobs@cs.sookmyung.ac.kr
††† 비 회 원 : 숙명여자대학교 전산학과
cms@cs.sookmyung.ac.kr
†††† 종신회원 : 숙명여자대학교 정보과학부 교수
jnpark@cs.sookmyung.ac.kr
논문접수 : 2000년 7월 19일
심사완료 : 2001년 9월 12일

보로 사용자의 요구를 해결하기 위한 기본 정보가 될 수 있다. 동적인 정보는 정적인 정보를 활용해 사용자의 요구를 해결하는 행위(behavior)로 시스템 구현 후 프로그램 모듈로 시스템 내에 존재한다. 시스템이 안정적으로 운영되기 위해서는 정적인 정보가 체계적으로 구축되어 있어야 한다.

비즈니스 규칙의 경우, 규칙 자체는 변동사항이 있을 수 있으나, 이벤트의 흐름이나 구조의 변화에는 크게 영향을 받지 않는다. 그러나, 이벤트를 처리하는 중요한 정보가 되므로 시스템 분석 모델에 표현하여 설계이후의 단계에 유연하게 적용하고 사용자의 참여를 보다 적극적으로 이끌 수 있다.

시스템 분석 과정에 도출되는 개체의 경우, 출석부나 성적표의 경우를 생각해 보면, 이 정보들은 시스템의 개체로 판단되지만, 그 내용은 시스템의 다른 개체들의 정보를 추출하거나, 계산한 결과를 포함하므로, 시스템의 사용자나 제품/서비스와는 다른 성격의 개체이다. 이들은 유도되거나 계산되지 않은 개체에 비해 보관하는 측면이나 정보를 변경하는 측면 또한 구현하는 측면에서 다른 성격을 가진다. 특히, 비즈니스 시스템에는 이런 정보가 많이 존재하므로, 별도로 관리한다면, 시스템을 더 효율적으로 구축하고 운영할 수 있다.

그리고, 대상 시스템의 소프트웨어 아키텍처(software architecture)가 구체적이거나 기존의 시스템을 재구축하는 경우 시스템을 관리하기 위한 메타 정보는 시스템 분석 단계에서 개체가 도출되면 그 정보도 파악되고, 설계단계에서 활용할 수 있고, 사용자와의 원활한 의사소통에 활용할 수도 있다.

또한, 그림 1에서 표시된 바와 같이[2], 개발 환경이나 정보 기술의 발달로 시스템 개발기간은 늘어난 반면, 전자상거래 등 비즈니스 환경의 변화로 사용자 요구가 증가하고, 시스템 운영 방식이 변함에 따라 기존 시스템을 사용할 수 있는 기간은 줄어드는 경향이 커지고 있다. 이런 상황을 극복하기 위해서 시스템 개발기간을 단축시키는 방법을 생각할 수 있다. 개발기간의 단축은 기존에 사용하던 시스템을 그대로 유지하면서 하드웨어적 그리고 소프트웨어적인 변화를 수용하는 시스템을 개발함으로써 이루어질 수 있다. 기존의 시스템을 그대로 유지하면서 새로운 시스템으로 전환한다는 것은 시스템 내의 정보는 그대로 유지하면서 이를 이용하는 응용프로그램(application)이 바뀐다는 것을 의미한다.

본 연구에서는 이러한 몇 가지 상황들을 반영할 수 있는 분석 모델을 제안한다. 우선 비즈니스 규칙이나 제약조건에 개체의 개념을 적용하고, 둘 이상의 다른 개체에서 유도되거나 계산된 개체의 경우, 기존의 개체와는 다른 형태로 표현하는 것을 제안한다. 이 방법을 통해, 시스템 관리정보를 분석 단계에 표현하며, 시스템에 존재하는 개체나 이벤트를 사용자와 관리자의 시각으로 도출할 수 있다.

## 2. 관련 연구

분석 단계의 목표는 사용자의 요구사항과 그 요구사항을 해결하게 위해 시스템이 준비해야하는 자원을 파악하는 것이다. 여기서의 자원이란 사용자에게 서비스를 제공하기 위해 갖추어야할 정보를 의미하며, 시스템의 재산이 된다. 즉, 시스템의 재산을 완전히 파악하는 것이 분석단계의 목표라 할 수 있으며, 이들은 시스템에서 개체(*entity*)로 표시된다[3].

UML의 개체를 추출하고 분석하는 단계 즉, 개체 모델링의 개념은[4] 그림 2와 같다.

유즈케이스모델(Usecase model)에서 동적인 부분은 순차 다이아그램(Sequence diagram)에서 표현하고, 정적인 부분은 클래스 다이아그램(Class diagram)에서 표현한다. 클래스 다이아그램은 데이타베이스 스키마에 해당하는 도메인 모델(Domain model)을 기초로 확장해간

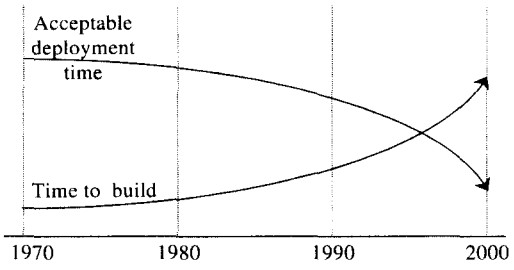


그림 1 개발기간과 시스템 사용기간의 변화

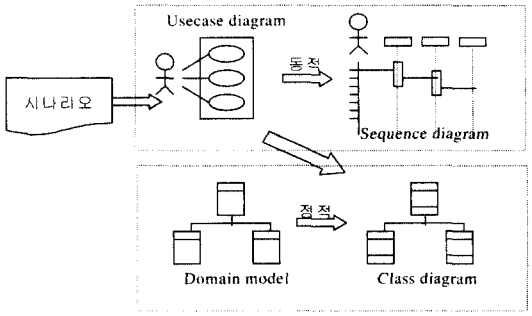


그림 2 UML 객체모델링

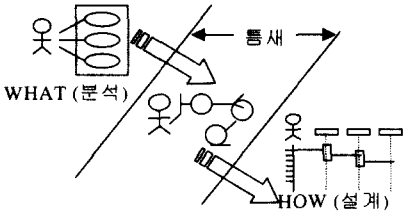


그림 3 'WHAT' 과 'HOW'

다. 동적명세나 정적명세 도중에 도출되는 클래스 또는 개체를 계속 반영하며, 다이아그램을 완성해 나가는 과정을 거친다.

이 사이에 개체의 도출이 완전히 되었는가에 대한 의문과 분석단계의 완전성을 높이기 위해 UML에 정식 채택되지는 않았지만, 올 사용한다. 이는 그림 3과 같이 분석과 설계 사이의 틈새(gap)을 좁히는 효과를 얻을 수 있다.

Robustness diagram는 엔티티(entity)객체, 경계(boundary)객체 그리고 제어(control)객체의 스테레오 타입으로 시스템을 정의한다. 스테레오 타입이란 UML에 공식적으로 정의되지는 않지만 행동이 같은 요소들을 사용자가 정의하여 사용할 수 있도록 하는 UML의 확장장치이다. 이 때, 클래스 다이아그램이 만들어지기까지의 과정에서 분석가마다 다른 결과를 보이는 모호함을 가질 수 있다[5,6]. 분석 단계의 모호함을 해결하지 못한다면 의도하던 시스템과 구축된 시스템은 시간경과와 함께 점점 멀어질 수밖에 없다. 따라서, 분석 단계에서 전체적인 도메인 분석뿐만 아니라 설계의 단계로 넘어가기 전에 객체의 추출 및 분석은 완전한 시스템 구축을 위해 필수 조건인 것이다[4,7,8].

또한, 시스템의 관리 정보는 시스템 분석 단계에서 개체가 도출되면 그 정보도 파악된다. 이를 분석 단계에서 표현한다면, 설계단계에서 활용할 수 있고, 사용자와의 원활한 의사소통에 활용할 수도 있다[9]. 특히, 기존 시스템이 존재하거나, 대상 시스템의 아키텍쳐가 구체적인 경우에는 개체를 누가 관리하는지, 어느 데이타베이스에 저장이 되어 있는지 혹은 될 것인지가 비교적 상세히 분석될 수 있다. 이는 분석 단계에서 반드시 해야하는 부분은 아닐 수 있으나, 분석 단계에서 표현할 수 있다면 이후 단계에 드는 노력을 줄일 수 있다. 기존의 연구에서는 메타정보를 구현 단계에서 다루거나[10] 분석 모델과는 별도로 두어[1,7] 사용자가 가장 많이 참여할 수 있는 분석 단계에서 다루어지지 않으므로, 많은 부분이 분석가의 몫으로 남겨지게 된다.

비즈니스 규칙의 경우, 기존의 시스템 분석 모델[7,11, 12,13]에서는 이벤트의 제약조건(constraints)으로 표시하거나, 표시하지 않고 구현 단계에서 프로그램에 포함했다. 그 결과, 제약조건이 바뀌거나 추가되는 경우, 프로그램을 매 번 수정하게 된다. 그러나, 이러한 변동사항은 분석 모델에 표시될 수 없었으며, 어떤 이벤트와 어떤 규칙이 연관되는지를 표현할 수 없었다. Catalysis는 "invariant"를 이용해 비즈니스 모델의 제약사항을 표현한다. 그러나, 이는 별도의 개체 개념이 아닌 프로그램 모듈의 개념으로 표현한 방식이다[14].

예를 들어, 수강시스템의 개체 중 출석부나 성적표 등은 기존의 연구에서는 특별한 언급 없거나, 이벤트의 결과로써 일회적인 성격으로 다루고 있다. 예를 들어, "성적 산출"이라는 이벤트가 있다면, 그 결과물로 성적표가 만들어지고 성적표가 시스템 내에 어떤 식으로 존재하는가에 대한 언급은 없다. 그러나, 출석부나 성적표 등의 정보는 시스템의 재산으로 영구히 보관해야되지는 않지만 일정기간 보관이 필요한 경우이므로, 시스템의 어딘가에 있어야하고, 이를 표현할 수 있어야 한다. 특히, 비즈니스 시스템의 경우, 이러한 성격의 정보가 상대적으로 많이 존재하므로 이에 대한 대안이 더욱 필요하다.

### 2.1 UML의 객체모델링의 예

UML의 예로 투자 거래 시스템의 객체 모델을 도출해본다. 결과물은 유즈케이스 다이아그램, 순차 다이아그램 및 클래스 다이아그램이며, 중요하게 관찰해야하는 부분은 개체 도출이 잘되었는가 하는 점이다. 참고로 이 부분은 [4]의 예의 일부이다.

'거래등록', '보고서 생성' 그리고 '거래갱신'의 기능을 하는 투자거래 시스템을 가정하자. 거래 등록에는 '사자 등록'과 '팔자등록'이 있다.

(1) 그림4와 같이 시스템의 유즈케이스 다이아그램을 그린다.

이 중, 이 절에서는 '거래등록' 중의 '사자 등록'에 관한 예를 든다.

(2) '사자 등록'의 세부 시나리오는 다음과 같다.

기본흐름은 일반적인 경우의 처리흐름이고 대안흐름은 예외상황에서의 처리흐름을 의미한다.

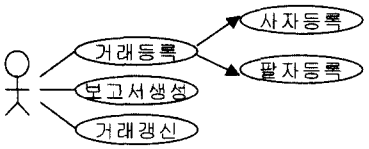


그림 4 UML예-유즈케이스

<기본흐름>
1. 거래인은 채권등록윈도우를 이용하여 거래에 필요한 자료들을 입력한다.
2. 시스템은 일반거래자료와 채권에만 해당되는 자료를 모두 확인 후 거래를 처리한다.

<대안흐름>
1. 만일 확인에 실패하면 사용자에게 틀린 값을 강조하여 표시해주고, 새로운 값을 받는다.

(3) 시나리오에서 정적모델을 위한 개체를 UML의 클래스 식별기준에 따라 도출하면, 다음 세 가지가 된다. 하나의 개체는 클래스 다이아그램에서 하나의 클래스가 된다.

***거래 , 일반거래, 채권***

여기서 '거래'는 거래에 필요한 자료들을 포함하고 '일반거래'는 거래 중 채권거래가 아닌 경우를 의미한다.

(4) Roubustness diagram을 그리면 그림 5와 같다[4]. 도출한 객체 중 '일반거래'와 '채권'은 '거래'에 포함되는 의미로 다이아그램에 표현되지 않았다.

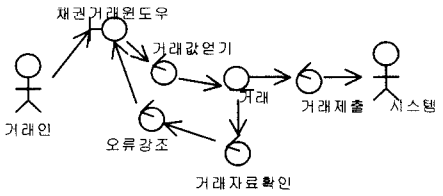


그림 5 UML예-Roubustness diagram

(5) 순차 다이아그램을 그림 6과 같이 그릴 수 있다.

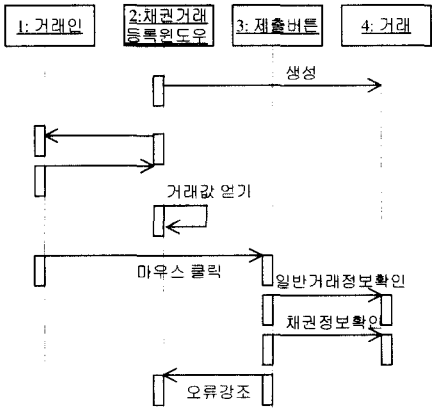


그림 6 UML예 - 순차 다이아그램

(6) 클래스를 도출한다. [4]에서의 결과는 '거래'에 포함된 정보에 대해 하위로 분류하여 클래스 다이아그램을 그렸는데, 이는 데이타베이스의 운영을 고려하여 작은 크기의 테이블로 나누어 관리하고자하는 측면과 본 사례의 시나리오에서는 도출되기 힘든다.

그리고, 클래스 다이아그램을 그리는 것보다는 클래스를 제대로 도출하는 것이 이 사례에서는 중요하므로 사례범위 내에서 도출할 수 있는 클래스들을 보이면 그림 7과 같다. 앞에서 도출된 '일반거래'와 '채권'은 '거래'의 확장이 되고, 시스템에서 거래가 이루어진 히스토리 관리를 위해 '거래목록'이 추가된 것을 알 수 있다.

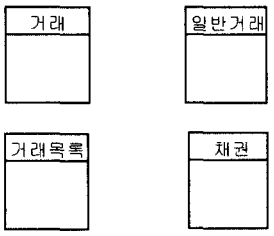


그림 7 UML예-클래스도출

## 3. 제안한 분석모델

### 3.1 기본개념

본 연구에서 제안하는 비즈니스 시스템 분석 모델은 시스템에 존재하는 다양한 정보를 분석 단계에서 표현하기 위하여 비즈니스 다른 개체에서 유도되거나 계산된 정보에 개체개념을 적용했다.

시스템은 사용자의 요구를 분석하여, 시스템 내의 정보를 이용해 사용자가 원하는 응답을 제공하는 기능을 한다. 이 때, 요구는 ***이벤트(event)***로, 사용자와 시스템 내의 정보는 ***개체(entity)***로 정의하며, 개체의 종류는 사용자에 관한 정보, 제품/서비스에 관한 정보, 시설 장비에 관한 정보, 비즈니스 규칙 및 관리 정보 등으로 구별할 수 있으며[15], 이들 사이의 협력으로 시스템의 이벤트가 처리된다. 시스템의 이벤트를 처리하기 위한 내부 이벤트를 ***액션(action)***으로 정의한다. 개체를 구별하고 표현할 수 있는 정보는 다음과 같다.

첫째, 사용자에 관한 정보

사용자에 관한 정보는 시스템에서 다루는 제품을 구매하는 사람, 시스템 사용자 또는 외부 시스템에 관한 정보를 말한다. 예를 들면, 수강신청 시스템에서는 학생, 은행의 관리자 또는 클라이언트 시스템에서는 서버 시스템/시스템관리자 등이 이에 해당된다. 그 외, 제품 공

급자, 업무관리자, 정보운영자 등이 사용자에 포함될 수 있다. 이를 본 연구에서는 ***사용자 개체(user entity)***로 구분한다.

둘째, 제품/서비스에 관한 정보

시스템에서 취급하는 제품이나 서비스에 관한 정보로, 제품은 두 가지로 나누어질 수 있다. 하나는 수강신청 시스템의 개설과목, 백화점의 상품, 은행의 금융상품 등 시스템의 기본적인 목적과 밀접한 개체로 다른 개체와 독립적으로 존재할 수 있는 특성을 갖는다. 본 연구에서는 이를 ***독립개체(dependent entity)***로 정의한다. 다른 하나는 둘 이상의 독립개체들 간의 계산이나 추출 및 병합에 의해 구성되어 원시정보(resource)의 변동에 따라 내용이 변할 수 있는 개체가 있다. 수강신청 업무의 출석부를 그 예로 들 수 있다. 이를 기존에는 제어(control)의 개념으로 사용했는데 문제는 제어의 결과물은 시스템의 재산으로 표현할 수 없어서, 실제 시스템에서는 이를 시스템의 재산으로 보고 일정기간 보관을 하며 따로 관리를 하고 있는데 분석 과정에 도출이 되지 않으므로 실제와는 다른 분석을 하게된다는 점이다. 본 연구에서는 이러한 성격의 개체를 ***종속개체(dependent entity)***로 구분한다.

셋째, 시설/장비에 관한 정보

제품이나 서비스를 생산하고 지원하기 위한 정보로 수강신청 시스템에서는 강의실을 예로 들 수 있다. 정보의 목적은 제품/서비스에 관한 정보와는 다른 면이 있지만, 시스템 내에서 이를 다루는 방식은 독립개체의 성격을 가지므로 독립개체로 구분한다.

넷째, 비즈니스 규칙/제약조건 정보

비즈니스 규칙이나 제약조건은 시스템 특히 비즈니스 시스템을 운영하는 중요한 정보로, 시스템에서 큰 비중을 차지한다. 기존의 분석 모델에서는 이를 취급하지 않았는데 그 결과 시스템과 비즈니스 정보와 가시적으로 연결되지 못했다. 이를 개선하기 위해 비즈니스 규칙 정보를 개체의 개념으로 구분하는 것을 제안한다. 이에 대한 구현은 UML에서 최근 시도하는 OCL[8]이나 웹 환경의 비즈니스 모델에서 많이 시도되고 있는 규칙베이스(rule base)를 활용할 수 있다. 분석 모델에서는 어떻게 구현하는가 보다는 어떤 이벤트가 어떤 비즈니스 규칙 정보와 연관 되어있는가를 표현하는 것이 중요하며, 본 연구에서는 이를 ***제약조건개체(constraints entity)***로 구분한다. 수강신청 시스템에서는 필수 이수과목의 기준, 부전공 신청의 조건 등이 이에 해당된다.

다섯째, 관리 정보

관리 정보는 시스템 내의 개체에 부여된 전반적인 관리사항을 명세하는 것으로 관리자, 접근권한이나 개체가 저장된 개념적 또는 물리적 위치 등을 의미하며, 분석 단계에서는 레거시(legacy)가 없는 경우에는 완전히 도출될 수 없는 경우도 많다. 그러나, 대상 시스템의 소프트웨어 아키텍처가 구체적이거나 기존 시스템을 체계적으로 관리하고 있었다면 분석 단계에서 도출될 수 있다. 즉, 분석 단계에서 알 수 있는 최대한의 정보를 명세한다.

개체와 이벤트는 시스템에서 그림 8과 같은 관계를 갖는다. 사용자 개체는 시스템에 이벤트를 보낸다. 시스템은 사용자 인터페이스를 통해 받은 이벤트를 처리하기 위해 그에 해당하는 액션을 처리해야 한다. 이 과정에서 시스템 내에 구축된 개체들을 이용할 수 있다.

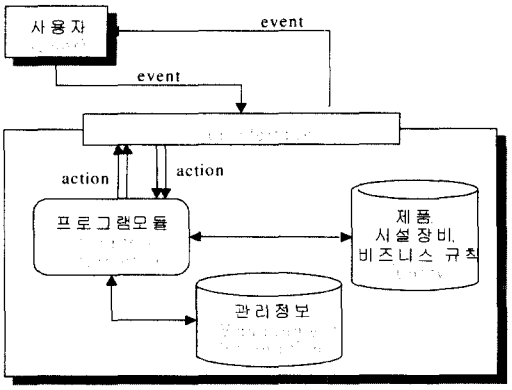


그림 8 시스템 내의 정보간의 관계

### 3.2 개체 도출

우선, 사용자 개체를 구별한다. 사용자 개체는 시스템의 직접적인 사용자, 시스템의 관리자, 외부 시스템 등이 될 수 있다.

다음으로 독립개체와 종속개체를 구별하기 위해 각 사용자 개체의 이벤트를 정의한다. 이벤트는 사용자가 시스템에 접근할 때, 인터페이스를 통해 시스템에 보내는 요구를 의미한다. 각 분석 모델마다 이벤트의 수준을 다르게 사용하는데 순서에 상관없이 무작위로 발생할 수 있는 사용자의 요구 단위[16,17], 시스템 내부의 개체를 변경하는 사용자의 요구 단위[14] 또는 사용자가 하는 모든 행위 단위[1] 등이다.

첫 번째의 경우, 순서에 상관없이 발생하는 단위는 이벤트의 추상화 수준이 높아 시스템의 전체 범위는 한눈에 알기 쉬우나 이벤트 내부가 복잡하고 관련 개체를 도출하기가 어렵다. 두 번째의 경우, 조회나 검색은 이

표 1 확장된 개체 개념을 적용해 분석한 수강업무

| 업무 | 사용자 | 이벤트 | 관련개체 | | | |
|---|---|---|---|---|---|---|
| | | | 독립개체 | 종속개체 | 제약조건개체 | 관리개체 |
| 수강업무 | 학생 | 수강신청<br>수강신청변경 | 과목<br>교수 | | 선수과목규정<br>이수학점규정 | 학생관리정보<br>과목관리정보<br>교수관리정보 |
| | 교수 | 강의시간표출력 | 과목<br>강의실 | 이번학기개설과목<br>강의시간표 | 강의시간규정 | 과목관리정보<br>강의실관리정보 |
| | 관리자 | 수강부출력 | 과목<br>교수<br>학생<br>강의실 | 이번학기개설과목<br>수강부 | 분반규정 | 과목관리정보<br>교수관리정보<br>학생관리정보<br>강의실관리정보 |

벤트로 분류될 수 없는데, 비즈니스 시스템의 경우 복합적인 조회나 검색이 많이 발생하고 그 결과가 시스템의 재산으로 활용될 수 있는 경우에는 부적합하다. 예를 들어, 수업업무의 수강부는 한시적이지만 시스템의 재산으로 활용되고, 사용자가 많이 요구할 수 있는 이벤트다. 이 경우, 다른 방법에서는 분석 모델에서 표현하는 기준이 모호하다. 세 번째는 사용자의 모든 행위를 하나의 이벤트로 보는 경우에는 도출되는 이벤트가 너무 많아 전체시스템의 윤곽을 파악하기가 어려워질 수 있고, 실제로 모든 이벤트를 한 번에 도출하기가 어렵다.

이런 모호함을 줄이기 위해서 본 연구에서는 유·무형의 서비스를 받기 위해 개체를 생성 및 변경하는 단위 또는 종속개체를 다루는 단위로 이벤트를 정의한다. 예를 들어, 수업업무의 경우 "수강신청"이나"강좌개설" 등의 개체를 변경하는 단위의 이벤트와 "수강부출력"등의 종속개체를 다루는 이벤트 등이 도출된다.

각 이벤트가 참조하는 개체를 정의한다. 개체는 3.1절에서 정의한 대로 제품/서비스와 시설/장비에 관한 정보를 독립개체와 종속개체의 개념으로 구분하여 도출한다. 독립개체는 시스템의 관리대상 개체 중 다른 개체들의 변동 사항에 영향을 받지 않는 개체로 수업업무의 경우, "과목", "전공" 및 "강의실"등이 될 수 있다. 종속개체의 예는 수업업무에서 관리자는 "수강부"를 관리하는데 이 때 "수강부"는 "강의실","과목","교수" 등에 연관된 종속개체가 될 수 있다. "이번학기개설과목"은 "과목"과 "교수"에 연관된 종속개체이다.

다음은 제약개체를 도출한다. 제약개체는 시나리오 상에 존재하는 제약조건을 개체화 한 것으로 최근의 여러 연구에서 설계이후 단계에서 별도로 관리하고 있다. 분석단계에서 시나리오 상에만 존재하던 제약조건이 설계단계에서 개체화된다면, 그 사이의 갭(gap)이 커지므로 이벤트에 연관된 개체로 구별해 놓는다.

관리개체를 도출한다. 시스템의 관리사항, 즉, 관리자명, 접근 권한, 데이타베이스명 또는 테이블명, 서버정보 등이 될 수 있다.

수강업무를 분석하면 표 1과 같이 분석할 수 있다. 표 1에 나타난 수강업무의 범위는 수강신청, 교수별 시간표 출력 및 수강부 출력 등의 업무로 제한한 것이다.

다음 3.3절에서는 표 1에서 분석된 이벤트나 개체들을 연관지어 표현하는 도구들과 그들에 대해 자세히 설명한다.

### 3.3 분석 산출물 작성단계

본 연구의 분석모델은 대학의 연구업무 분석 등 4개 시스템에 적용해 보았으며, 그 중 가장 최근의 적용사례는 웹데이타마이닝을 위한 ETT(Extraction / Transportation / Transformation), Web-Robot, Miner 등의 서버를 분석이다[16]. 개발 팀의 관리자, 개발자2-3명 그리고 분석가 2명이 팀을 이루어 워크스루(work-through)를 통해 진행했으며, 기존 시스템에 대한 아이디어만 있는 상태이면서 시스템에 대한 이해가 높은 사용자들의 특성을 잘 살려 분석 산출물을 구현까지 연결할 수 있었다. 다음 다이아그램은 웹데이타마이닝 시스템 중 ETT시스템의 일부를 사용한 것이다.

(분석1) 업무의 범위를 배경도로 작성한다.

배경도는 시스템의 범위를 표현하며, 이를 바탕으로 시스템 내에서 어떤 프로세스가 일어나게 될 것인가를 알 수 있는 중요한 도구이다[17].

시스템을 표현하는 타원과 관련자를 표현하는 사각형 사이의 이벤트를 표현한다. 관련자는 시스템에 직접 접근 가능하여 시스템에 자극(stimulus)을 줄 수 있는 사용자이며, 고객(client)와 업무관리자(job staff)로 구별될 수 있다. 그림 9는 배경도의 개념을 보여준다.

(분석2) 사용자와 그에 대한 이벤트를 이벤트다이아그램으로 정의한다.

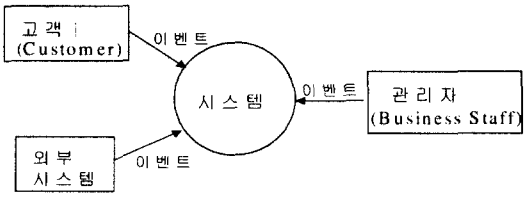


그림 9 배경도

이벤트다이아그램은 사용자가 시스템에 요구하는 이벤트를 WOD(Wanierr- Orr Diagram, 워니어-오어 다이아그램)의 형태로 표현한다[18]. WOD는 행위의 처리를 서술식으로 일목요연하게 볼 수 있는 장점이 있고, 그 형태는 그림 10과 같이 시작(start), 처리(process), 끝(end) 부분으로 구성되어 있다. 배경도에 나타난 사용자들과 시스템 간의 관계를 상세히 서술식으로 표현한다. 배경도가 시스템의 범위를 설정하는 도구라면, 이벤트다이아그램은 사용자의 활동을 보여주는 도구이다. 배경도에 나타난 사용자 개수만큼 이벤트다이아그램이 만들어진다.

' { ' 는 집합기호 중 오른쪽 괄호( } )를 제외한 형태의 괄호만을 취한 것이다. 이벤트 구조도의 제일 왼쪽, 즉 첫 단계 { 앞에는 사용자 개체명을 쓴다. 사용자는 배경도의 사용자와 동일하다. 시작 부분에는 사용자가 시스템에 접근하면서 발생하는 이벤트를, 끝 { 에는 시스템을 종료하면서 발생하는 이벤트를 기술한다. 이벤트들 간에는 순서가 있는 경우도 있고 그렇지 않은 경우도 있다. 이벤트를 정의하는 단위는 3.2절의 기준을 따른다.

이벤트에 상세 명세가 필요한 경우, 이벤트에 번호를 붙여 상세이벤트명세에 정의한다.

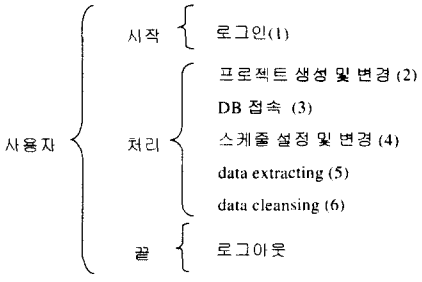


그림 10 이벤트 다이아그램

(분석3) 상세이벤트명세를 작성한다.

상세이벤트 명세는 이벤트 다이아그램에 나타난 각각의 이벤트를 처리하기 위한 세부 이벤트들을 명세하는 것이다. 액션은 이벤트를 처리하기 위한 선행조건이나, 이벤트를 수행한 후의 조치, 또는 단순히 이벤트의 일부분이 될 수 있다. 명세 방법은 이벤트 다이아그램과 마찬가지로 WOD의 기법을 이용한다. 선행조건은 "시작{" 부분에, 후조치는 "끝{" 부분에 작성한다. "처리{"부분의 액션 수행순서는 case문의 경우 ⊕{ 로, 논리적 AND나 OR의 경우, AND{ 또는 OR{로 각각 표현한다.

그리고, 사용자의 요구에 수반해서 시스템 내에서 처리해야 하는 액션이나 이벤트를 "시스템:" 부분에 그림 11과 같이 명세한다.

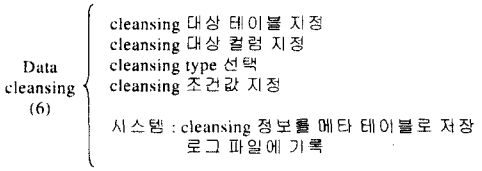


그림 11 상세이벤트 명세

(분석 4) 배경도에 정의한 사용자에 대한 정보다이아그램를 작성한다.

정보다이아그램은 이벤트를 해결하기 위한 자료를 표현한 그림으로 그림 12와 같다. 정보다이아그램을 통해, 사용자는 시스템에 어떤 요구를 할 수 있으며, 그 요구를 만족시킬 수 있도록 어떤 자료들이 구비되어 있어야 하는가 또는 준비되어야하는가를 알 수 있다.

또한, 정보다이아그램의 자료요소는 사용자와 시스템의 인터페이스 역할을 한다. 즉, 사용자가 이벤트를 요구하기 위해 시스템에 제공해야하는 정보가 되며, 구현 후, 사용자 인터페이스상에 나타날 자료요소로 연결된다.

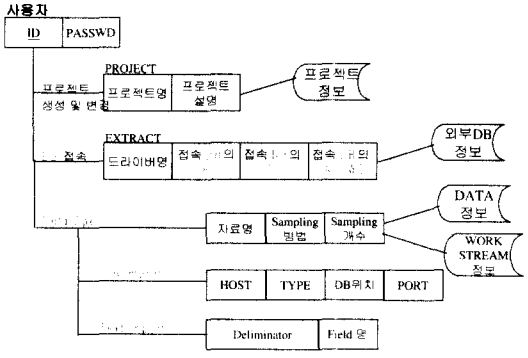


그림 12 정보다이아그램

이벤트를 처리하기 위해 필요한 개체를 3.2절에 설명한 종류 별로 도출하고 다이아그램에 표현한다. 그룹화된 이벤트의 경우, 세부 이벤트들을 대표 이벤트의 일부로 표현한다.

그림 13은 비즈니스 규칙/제한사항의 예로 수강신청 업무의 사용자 '학생'이 부전공의 신청하는 이벤트의 경우이다. 이 때, 부전공 신청 기준에 따라 수강신청이 가능한 경우의 정보다이아그램이다.

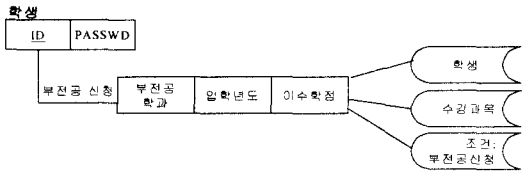


그림 13 비즈니스규칙을 표현한 정보다이아그램

(분석 5) 액션다이아그램을 작성한다.

액션 다이아그램은 이벤트를 해결하기 위한 세부적인 처리를 메소드 형태로 정의한다. 이 단계에서 도출되는 명세는 완벽할 수는 없지만, 정보다이아그램 상의 자료를 활용하고, 이벤트다이아그램의 이벤트를 구체화 할 수 있는 도구로써, 분석 마지막 단계와 설계 단계에서 더욱 구체화 될 수 있다. 각 정보다이아그램에 대해 그림 14와 같은 하나의 액션다이아그램이 산출된다.

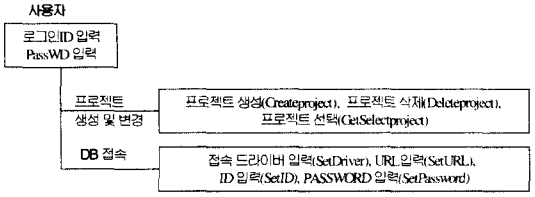


그림 14 액션다이아그램

(분석 6) 관리다이아그램을 작성한다.

관리다이아그램은 앞에서 작성한 다이아그램들의 개념적인 정보에 대한 물리적인 정보, 사용자 권한, 자료의 물리적 위치, 접근 경로, 데이타베이스 명이나 테이블명에 대한 사항 등을 표현한다. 이 때, 각 관리다이아그램은 그림 15의 형태로 정보다이아그램과 일대일 대응 관계를 이루며, 다음의 사항을 명세한다. 관리다이아그램의 사항은 분석 단계에서 모두 명세하지 않아도 된다. 이는 주로 구현 단계에 참조될 사항이므로, 설계 단계까지 완성된다.

-관리자 명 (manager name)

해당 테이블이나 데이타베이스의 관리자를 명시한다. 업무 담당자나 전산실의 데이타베이스 관리자가 될 수 있다.

-접근 권한 (authority level)

숫자 혹은 문자열로 표시되며, 직급이나 담당업무에 따라 시스템 내에서 정해진 레벨을 의미한다. 이는 사용자가 시스템에 로그인하는 시점에 권한에 따라 특정 업무에 대해 사용을 제한하는데 참조된다.

-데이타베이스 명(database name) 또는 테이블 명(table name)

해당 개체의 실제 데이타베이스나 테이블을 의미한다. 시스템에 구현된 물리적인 데이타베이스나 테이블명을 정의하여 한글 질의 처리를 이용한 정보 검색시 참조한다.

-서버위치(server location)

데이타베이스의 실제 위치를 기록한다. 이는 데이타베이스 명이나 테이블명과 함께 실제 데이타베이스에 접근하는 경우에 참조된다.

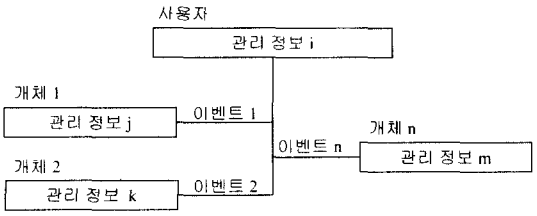


그림 15 관리다이아그램

### 3.3 사례 적용

2.1절의 UML의 객체모델링의 예에서 사용한 '투자거래시스템'을 본 연구에 적용해 본다.


그림 16 정보모델링 예-배경도

배경도는 그림 16과 같이 그릴 수 있다. 시스템의 업무관리자는 시스템의 행정적 관리 등의 역할을 한다. 기존에는 시스템 내에 감춰져 있던 기능을 밖으로 끌어내면서 시스템 내부는 기능적인 요소에 충실할 수 있게 된다.

이벤트 다이아그램은 2,1절의 시나리오를 기준으로 그림 17과 같이 표현된다. 각 이벤트의 번호는 상세이벤트 명세의 번호이다. 예를 들어, '주문등록(1)'의 경우에는 상세이벤트 명세 (1)을 참조한다. '주문 등록'의 상세이벤트 명세는 그림 18과 같다.

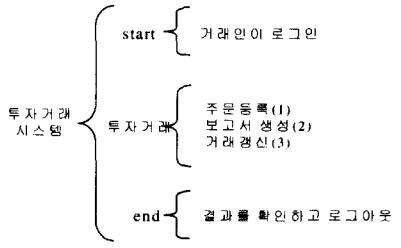


그림 17 정보모델링 예-이벤트다이아그램

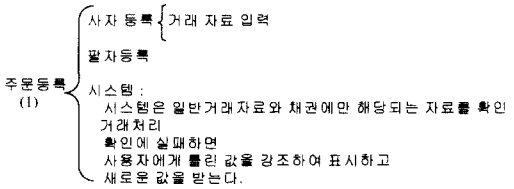


그림 18 정보모델링 예-상세 이벤트명세

정보다이아그램을 '주문등록'의 '사자 주문'에 대해 중점적으로 표현하면 그림 19와 같다. 객체를 도출하는 과정에서 독립개체로 '거래'와 '채권', 종속개체로 '거래목록', 그리고 제약조건 개체로 '사자주문 규약'등이 도출된 것을 알 수 있다.

행위다이아그램은 그림 20과 같은 형태로 표현된다.

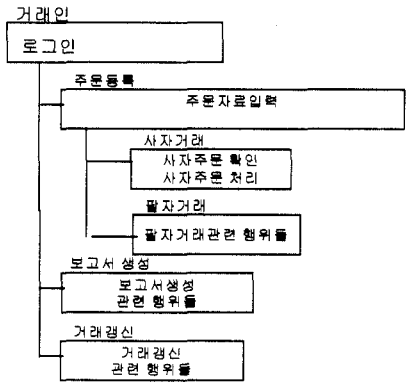


그림 19 정보모델링 예-행위다이아그램

액션 다이아그램과 관리 다이아그램은 시나리오의 구체적인 명세가 되어 있지 않으므로 이 절에서는 다루지 않겠다. 여기서 중요한 것은 본 연구에서는 개체도출에 있어서 확장된 개체개념을 적용함으로써 실제 필요한 개체를 분석단계에서 도출할 수 있으며 결과적으로 안정적이고 완전한 분석에 기여할 수 있다는 점이다.

## 4. 특징 및 기대효과

### 4.1 비즈니스 규칙/제약조건을 개체로 정의

기존 방법에서는 개체와 이벤트는 도출하지만, 비즈니스 규칙이나 제약조건은 개체로 취급하지 않고, 구현 단계에서 프로그램 모듈로 구현해왔다. 이 경우, 변동사항을 반영하기 위해서는 관련된 프로그램 모듈을 모두 수정해야 한다. 이러한 작업은 시스템의 개발 기간을 길게 하며, 유지보수 비용이 증가되는 결과를 초래한다.

본 연구에서는 이런 점을 줄이기 위해 이를 개체로 도출하여 분석 단계에 표현할 수 있도록 했으며, 구현은 규칙베이스(rulebase)를 구축하는 기술로 해결이 가능하다. 어떤 규칙이 어떤 이벤트와 연관되었는가를 분석 단계에서 표현하고, 도출함으로써 상대적으로 프로그램의 구현 및 유지보수의 부담이 덜어질 수 있다.

### 4.2 확장된 개체 개념의 도입

본 연구에서는 기존의 분석 모델의 결과가 모호하고 완전성이 떨어지는 점을 보완하기 위해 체계적으로 개체를 도출하기 위한 노력으로 개체를 4가지로 구분했다. 우선 사용자개체(user entity), 이벤트를 처리하기 위해 사용하는 독립개체(independent entity), 시스템 내의 재산으로 가치가 있지만 둘 이상의 개체로부터 유도되거나 계산되어 원시 정보에 따라 내용이 변경되는 종속개체(dependent entity)가 있다. 그리고 비즈니스 규칙이나 제약조건을 표시하는 제약조건개체(constraints entity)와 관리정보를 표현할 수 있는 관리개체(management entity)가 있다.

### 4.3 관리다이아그램의 도입

관리정보의 경우 기존의 연구에서는 분석 모델 내에 포함된 것은 아니고, 도큐먼트로 존재했다. 그러나, 시스템의 메타정보 역시 시스템의 자산이 되고 실제로 데이타베이스로 관리하므로 본 연구에서는 항목을 정하여 개체의 개념을 적용한 것이다. 시스템에 대한 행정적이거나 정보보안 등에 관한 사항을 분석 단계에 표현함으로써 사용자나 분석가의 이해를 돕는다. 이는 분석 단계에서 완전히 도출될 수는 없지만, 대상 시스템의 소프트웨어 아키텍처가 구체적이거나 기존 시스템을 체계적으

로 관리하고 있었다면 분석 단계에서 효과적으로 활용할 수 있다.

### 4.4 설계 및 구현으로의 연결

본 연구에서 제안한 모델은 정보다이아그램의 인터페이스 정보와 확장된 개념의 개체 그리고 이를 활용하는 이벤트 또는 행위의 세 가지 관점으로 이루어진다. 즉 인터페이스(Interface)-개체(Entity)-제어(Control)의 관점으로 모델을 구성할 수 있다. 이는 객체지향의 모델(Model)-뷰(View)-제어기(Controller)의 구조나, 전통적인 클라이언트/서버의 그래픽사용자인터페이스(GUI)-자료저장소(Data Repository)-논리(Logic)의 3-tier형태에 대응될 수 있어서 설계이후의 단계에서 이러한 구조로 연결하는데 자연스럽다.

## 5. 결 론

본 연구에서는 비즈니스 시스템을 체계적으로 분석하기 위한 노력으로, 시스템 내에서 발생하는 개체를 독립개체, 종속개체, 제약조건개체 및 관리정보의 4가지로 구분했고, 이를 체계적으로 도출하여 표현하기 위한 비즈니스 시스템 분석 모델을 제안했다.

개체의 안정적인 구축은 시스템의 재산이 안정적으로 구축된다는 의미로 내·외부적 환경 변화에 쉽게 적응할 수 있는 효과를 가져올 수 있다. 기존에는 본 연구의 독립개체에 해당하는 개체들만 적용하고 종속개체의 개념은 적용하지 않았으나, 이를 구분하여 도출함으로써 보다 안정적으로 자원을 관리할 수 있다. 그 중 비즈니스 시스템에서 비중이 큰 정보인 비즈니스 규칙과 관리정보를 분석단계에 접목시킨 것은, 프로그램 모듈에 치우치던 시스템 구현 및 유지보수 비용을 덜어줄 수 있는 효과가 있다.

## 참 고 문 헌


[1] Ivar Jacobson, The Object Advantage : Business Process Reengineering with Object Technology, Addison-Wesley, 1994.

[2] David A.Taylor, Business Engineering with Object Technology, pp. 13-28, John Wiley & Sons, 1995.

[3] Chris Marshall, Enterprise Modeling with UML, Addison-Wesley, 2000

[4] Doug Rosenberg, Kendall Scott. UML Object Modeling, Addison-Wesley, 2000.

[5] Sinan Si Alhir, "The UML- Two years agter adoption of the standard", http://home.earthlink.net/~salhir/theumltwoyearsafteradoptionofthestandard.html

[6] A. Evans, "Rigorous Development in UML," ETAPS'99, FASE Workshop, LNCCS, 1999.

[7] Martin Fowler, UML Distilled Applying the Standard Object Modeling Language, Addison-Wesley Longman Inc., 1997.

[8] Jose Warmer, Anneke Kleppe, The Object Constraint Language, precise modeling with UML, Addison-Wesley, 1999.

[9] 이서정, 사용자 중심의 일관된 시각을 지원하는 객체지향 시스템 분석 및 설계, 숙명여자대학교 박사학위논문, 1998년 2월.

[10] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen, Object-Oriented Modeling and Design, Prentice-Hall, 1991.

[11] S. Shlaer and S. J. Mellor, The Shlaer-Mellor Method, Project Technology Inc., 1996.

[12] Donald G. Firesmith, Object-Oriented Requirements Analysis and Design, John Wiley & Sons, 1993.

[13] Rational Software Co., UML ver. 1.0 Notation Guide, Rational Software Co., 1997.

[14] D'Souza and Wills, Object, Components, and frameworks with UML: The Catalysis Approach, Addison-Wesley, 1999, p.496, pp.548-549.

[15] 박재년, "정보구조모델링에 의한 시스템 분석", 숙명여자대학교 논문집, 제33집, 1992년 12월.

[16] ArsMagna, 숙명여대 SE Lab, 정보모델링을 적용한 E-biz Miner분석보고서, 2001년1월

[17] Ivar Jacobson, Grady Booch, James Rumbaugh, The Unified Software Development Process, Addison-Wesley, 1999.

[18] 박재년, 기초전산학, pp. 337-366, 교학사, 1997.



이 서 정

1998년 ~ 현재 동덕여대 교양교직학부 강의전임강사. 1998년 숙명여자대학교 전산학과 박사. 1991년 숙명여자대학교 전산학과 석사. 1989년 숙명여자대학교 전산학과 학사. 관심분야는 소프트웨어 개발 방법론, 메타모델, 메트릭스



고 병 선

1998년 ~ 현재 숙명여자대학교 컴퓨터과학과 박사과정. 1998년 숙명여자대학교 전산학과 석사. 1995년 숙명여자대학교 전산학과 학사. 관심분야는 S/W 공학, 객체지향 프로그래밍, 메트릭스, 품질 평가, 재사용

최 미 숙

1997년 ~ 현재 숙명여자대학교 전산학과 박사 과정. 1995년 ~ 1999년 나주대학 소프트웨어개발과 전임강사. 1994년 숙명여자대학교 대학원 전산학과(석사). 1990년 전북대학교 수학과 졸업(학사). 관심분야는 객체지향 모델링, 컴포넌트 개발 방법론, 컴포넌트 평가

박 재 년

정보과학회논문지 : 소프트웨어 및 응용
제 28 권 제 10 호 참조