경영정보학연구
제6권 2호
1996년 12월

클라이언트/서버 정보 시스템 개발을 위한 객체 지향 시스템 설계 기법

임 홍 순[1] • 김 종 우[2] • 박 성 주[3]

Object-Oriented System Design Techniques for Client/Server Information System Development

최근 들어 많은 정보 시스템들이 클라이언트/서버 환경에서 개발되고 있다. 클라이언트/서버 정보 시스템은 기존의 정보 시스템에 비해 복잡도가 높은 시스템이므로, 시스템 개발 시 체계적인 개발 방법론의 적용이 필요하다. 객체 지향 개발 방법론은 객체 지향 개념에 기반하여 시스템의 재사용성, 확장성, 신뢰성, Scalability를 증진시킬 수 있는 방법론이다. 본 연구에서는 객체 지향 방법론에서 클라이언트/서버 정보 시스템 설계 작업을 체계적으로 수행하기 위한 기법과 절차를 제시한다. 이러한 기법은 설계 작업시 사용자 인터페이스에 대한 충분한 고려가 가능하고, 설계 객체의 규명을 체계적으로 지원하고, 응용 시스템 논리를 일관성 있게 배분할 수 있도록 도와준다. 한국과학기술원의 인텔리전트 캠퍼스 프로젝트에서 개발된 학사 시스템의 적용 사례와 함께 본 연구에서 제시하고 있는 객체 지향 설계 기법에 대해 소개한다.

Keywords : 객체 지향 개발 방법론, 클라이언트/서버 정보 시스템, 정보 시스템 설계

1) 한국과학기술원 경영정보연구센터
2) 충남대학교 자연과학대학 통계학과
3) 한국과학기술원 테크노경영대학원

I. 서 론

정보통신기술의 발전으로 컴퓨터를 이용한 다양한 형태의 정보 교환과 서비스 제공이 가능하게 되면서 정보 시스템은 점점 더 이질화(Heterogeneity), 분산화(Distribution), 비동기화(Asynchronous) 되고 있다. 과거에는 정보 시스템이 텍스트 중심의 인터페이스와 단순한 자료 입출력 기능만을 지원하였으나, 최근에는 정보 시스템에 대한 사용자의 기대 수준이 급격히 높아져서 GUI(Graphic User Interface)의 제공은 물론, 다양한 플랫폼에 존재하는 자료의 공유, 통합된 서비스 등과 함께 그래픽스, 멀티미디어 등의 복잡한 자료 형태를 사용한 다양한 기능들이 필요하게 되었다. 이와 같은 정보 시스템의 추세에 따라 과거의 호스트 중심 환경으로부터 클라이언트/서버 환경으로 정보 시스템들이 급속히 이전되고 있다.

클라이언트/서버 환경하에서의 정보 시스템 개발은 분산 환경에 따른 컴퓨팅 요소간의 상호의존성(Inter-dependency), 상호작용성(Inter-operability), 통합성(Integration), 이식성(Portability) 등의 문제를 고려해야 함에 따라 시스템 개발의 복잡성(Complexity)이 증가하게 된다[2,24]. 따라서 클라이언트/서버 정보 시스템의 개발과 관리를 효과적으로 하기 위해서는 체계적인 개발 방법론이 필요하다.

객체 지향 기술은 소프트웨어의 생산성 및 품질 향상을 위한 기반 기술일 뿐 아니라 복잡한 환경에 적응하기 위한 필수적인 기술로 인정받고 있다. 객체 지향 기술에 기반한 객체 지향 개발 방법론은 객체 지향 프로그래밍 언어와 연결이 용이하며, 시스템의 생산성, 재사용성, 확장성, 신뢰성, Scalability등을 증진 시킬 수 있는 방법론이다[10,25]. 현재까지 개발된 대표적인 객체 지향 개발 방법론에는 Object Oriented Design[3], Object Modeling Technique[20], Object Oriented Analysis[6], Object Oriented Analysis and Design[14], Object Oriented Software Engineering[11] 등이 있다. Fusion[7], MOSES [10], SOMA [12], Unified Modeling Language[4,5] 등은 기존의 방법론들의 단점을 보완하여 최근 제시되고 있는 방법론들이다.

클라이언트/서버 환경에 객체 지향 개발 방법론을 적용하는 것은 현재까지 객체 지향 방법론이 갖는 성숙되지 못한 부분과 클라이언트/서버 환경의 복잡성이 복합되어 시스템 개발을 복잡하고 어렵게 만들고 있는데, 특히 시스템 설계 단계에서 많은 문제점이 도출된다. 이러한 문제점들에는 설계 단계에서 객체 규명의 어려움, 화면 설계의 변경에 따른 잦은 객체 변경, 응용 시스템 논리의 분배 원칙 부재, 객체간의 협력(Cooperative Work)관계 표현의 어려움, 설계 절차의 체계적인 지원 부족 등이 포함된다. 결과적으로 기존의 객체 지향 개발 방법론을 클라이언트/서버 시스템 개발에 적용하는 경우, 구현된 시스템은 구현에서 요구사항까지의 추적성(Traceability)이 부족하

고, 일관성과 타당성의 부재로 생산성 및 품질 보증에 한계를 갖는다.

본 논문에서는 객체 지향 개발 방법론을 사용하여 클라이언트/서버 정보 시스템을 개발하는 경우, 설계 단계를 체계적으로 수행하기 위한 기법과 절차를 제시한다. 본 논문에서 제시하고 있는 설계 기법 및 절차는 CAMIS 객체 지향 개발 방법론의 일부이다. CAMIS 객체 지향 개발 방법론은 한국과학기술원에서 수행 중인 인텔리전트 캠퍼스 과제에 실제적으로 적용되면서 개발된 방법론으로서 클라이언트/서버 정보 시스템 개발의 생산성을 높이고 개발된 시스템의 품질을 보증하기 위한 객체 지향 방법론이다[1]. 본 논문의 설계 기법 및 절차는 시스템 개발 수명 주기 전반에 걸친 모든 기법 및 절차를 포괄하지는 않으며 구현 환경에 독립적인 설계 단계를 대상으로 한다.

본 논문의 구성은 아래와 같다. 제2장에서는 객체 지향 설계를 위한 개념틀(Framework)에 대한 소개로서, 기존의 객체 지향 개념틀에 설계를 위하여 추가적으로 고려되어야 할 주요 개념들을 기술한다. 제3장에서는 예제를 중심으로 설계 기법, 절차 및 구조물(Construct)들을 제시하고 제4장에서는 설계와 코딩과의 연결을 4GL(Generation Language) GUI 생성 도구를 사용하는 사례를 중심으로 설명한다. 제 5장은 본 논문의 결론 및 향후 연구 방향을 제시한다.

II. 객체 지향 설계를 위한 개념틀 (Framework)

2.1 개 요

객체 지향 방법론에 대한 연구들은 객체 지향 기술을 타당성과 실행 가능성을 갖춘 기술로 발전시켰다[3, 5, 6, 7, 10, 12, 14, 20]. 특히 객체 지향 시스템의 생산성과 품질은 다양한 평가 차원(Dimension)(예를 들면, 재사용성, Cohesion, Coupling 등)에서 우수성이 인정되어 왔지만 이러한 결과는 단지 객체 지향 시스템의 최종 산물(Deliverable Objects)인 객체의 특성에 기인한다[23]. 객체 지향 기술이 품질과 생산성을 보장받고 실용성과 현실성을 갖추기 위해서는 프로그래밍 언어, 분석 및 설계 기법, 개발 절차, 객체 관리 등에 관한 추가적인 연구가 필요하다[16]. 특히 객체 지향 설계의 경우, 기존의 방법론에서 체계적인 기법 및 절차에 대한 지원이 이루어지지 않고 있기 때문에 시스템 개발의 일관성, 구현으로의 자연스러운 전이(Seamless Transition)를 유지하지 못하고 있다. 이러한 이유로 아직 많은 응용 시스템 개발이 객체 지향 기술로 수행되지 못하고 있다[25].

인텔리전트 캠퍼스 과제는 한국과학기술원의 교육, 연구, 행정에 관련된 대학의 모든 활동을 지원하기 위한 캠퍼스 정보 시스템의 구현을 목적으로 수행되었다[18]. CAMIS 객체 지향 개발 방법론은 기존의 객체 지향 개발 방

법론을 인텔리전트 캠퍼스 과제에 적용하면서 축적된 경험 및 노하우와 실증적인 사례들을 바탕으로 개발된 방법론으로서 클라이언트/서버 환경에서 정보 시스템을 구현하는 경우, 시스템의 개발 생산성을 높이고 개발된 시스템의 품질을 보증하기 위한 객체 지향 시스템 개발 방법론이다[1].

본 논문에서 제시하는 설계 기법 및 절차는 기존의 객체 지향 개발 방법론을 바탕으로 클라이언트/서버 환경의 특성을 반영하여 체계적인 객체 규명 방법을 제시하고, 규명된 객체간의 명확한 동적 관계를 표현하기 위한 추가적인 구조물을 제공한다. 그러한 결과로 일관된 객체 모형의 개발을 유도하고, 구현으로의 자연스러운 전이(Seamless Transition)를 제공한다. 따라서 본 논문의 목적은 클라이언트/서버 정보 시스템의 설계를 위해 기존의 객체 지향 개념틀(Framework)에 추가적인 개념 및 기법을 제시하고 이를 바탕으로 구체적인 개발 절차 및 지침을 제공하는 것이다. 다음 절에서는 이러한 개념틀을 이루는 객체 모형과 동적 모형에 대해서 기술한다.

2.2 객체 모형

객체 지향 개발 방법론의 가장 큰 장점 중의 하나는 전 시스템 개발 수명 주기(System Development Lifecycle)에 걸쳐서 단일 구조물을 사용한다는 것이다. 즉, 시스템 개발에 관련된 모든 지식을 객체라는 구조물에 집약하여 표현하는 것이다. 따라서 객체 지향 개발 방법론의 가장 중요한 성공 요인은 효율적으로 완결성과 유연성을 갖는 객체 모형을 개발하는 것이다. 또한 객체 모형의 개발은 객체 규명 작업으로부터 시작되기 때문에 체계적인 객체 규명 작업의 지원이 성공적인 객체 모형 개발의 관건이 된다.

객체의 규명 작업은 객체의 발견(Discovery)과 발명(Invention)을 필요로 한다[3]. 객체의 발견은 현실 세계에 존재하는 객체를 규명하고 정의하는 작업으로서 일반적으로 시스템 개발의 분석 단계에서 수행되는 반면, 객체의 발명은 분석 단계에서 발견된 객체에 대한 추상화 외에 현실 세계에 존재하지 않지만 정보 시스템을 만들기 위해 부수적으로 필요한 객체를 창조하는 것으로 일반적으로 설계 단계에서 수행된다. 객체 규명 작업이 분석과 설계에 따라서 달라져야 하는 이유는 분석과 설계의 대상이 각각 문제 공간(Problem Space)과 해 공간(Solution Space)으로 달라지기 때문이다[15]. 현재까지의 대부분의 객체 지향 개발 방법론들이 사용하는 객체 규명 방법은 분석 단계에서 전체 시스템을 대상으로 문제 기술서(Problem Statement)의 문장으로부터 객체를 추출하여 각 객체에게 책임성(Responsibility)을 할당하는 Class-Responsibility-Collaborator(CRC) 기법이다[3, 4, 7, 11, 20]. 여기서 책임성은 필수적으로 수행해야 할 의무(Duty)를 의미함으로써, 해결해야 할 문제를 규명하고 잠재적인 해결 방안을 제시하는 것에 대한 바탕이 된다[7]. 그러나 기존의 객체 지향 방법론들은 단지 분석 단계의 객체 발

견에 초점을 두고 있을 뿐 설계 단계에서 수행되어야 할 객체 발명에 대한 체계적인 지원이 부족하다. 이에 따라 설계 단계에서 시스템 설계자들의 자의적이고 임의적인 객체 발명으로 시스템의 일관성이 저해되고 시스템 개발 후에 해당 시스템의 관리를 어렵게 한다.

클라이언트/서버 환경의 특성을 반영하여 객체의 의미와 역할에 따라 객체를 분류하는 것은 객체의 규명을 체계적으로 지원해 줄 수 있다. 클라이언트/서버 환경은 다양한 컴퓨팅 요소들이 서로의 독립성을 보장하면서 협조 처리(Cooperative Processing)가 이루어지도록 분산되어 있는 것이다[2]. 이러한 컴퓨팅 요소들은 크게 그 역할에 따라 표현 로직(Presentation Logic), 응용 로직(Application Logic), 데이터베이스 로직(Database Logic)으로 구분된다[2,19,24]. 따라서 클라이언트/서버 환경의 특성을 위해서는 객체를 세가지 종류의 컴퓨팅 요소에 대한 특성을 반영한 객체로 분류하는 것이 효과적이다[19, 21].

객체 분류를 반영한 방법론은 Jacobson의 방법론과 Krasner의 방법론이 있다[11, 13]. Jacobson은 객체를 Entity Objects, Control Objects, Interface Objects로 분류하고 Krasner는 객체를 Model Objects, View Objects, Controller Objects로 분류한다. Jacobson의 방법론은 요구 분석 단계에서 작성된 Use Case 모형을 바탕으로 객체를 규명한다. Use Case 모형은 시스템의 외부에서 시스템에 자극을 주는 액터(Actor)와 시스템의 기능성(Functionality)을 의미하는 Use Case의 상호 작용을 표현한다. Jocobson의 방법론에서 객체 규명 작업은 Use Case에 참여하는 객체들을 규명한 후, 객체들의 특성에 따라 객체를 분류하는 것이다. 즉, Jacobson의 방법론에서 분류된 객체들은 창조되는 것이 아니라 발견되는 객체들이다. 따라서 Jacobson의 방법론에서는 설계 단계에서 수행되어야 할 객체 규명 방법이 제공되지 않는다. 또한 객체들의 오퍼레이션 규명이 Use Case가 갖는 기능들을 적절히 객체들에게 배분하는 방식을 취하고 있다. 따라서 분석가의 임의적인 내용이 많이 포함되어 다른 방법론들과 마찬가지로 객체 지향 설계의 난점을 그대로 내포하고 있다[9]. Krasner의 방법론은 주로 사용자 인터페이스를 위한 개념틀로 개발되었다. Model Objects는 응용 영역에 대한 모형을 의미하고 Controller Objects는 사용자의 입력을 받을 수 있는 센서와 같은 객체들(예를 들면 버튼, 메뉴)을 의미하고 View Objects는 사용자에게 정보를 표시하는 객체들을 의미한다. 따라서 Krasner의 방법론에서는 GUI의 모든 구성요소(Widgets)를 객체로 모형화 하기 때문에 매우 많은 노력을 수반한다. 또한 사용자 인터페이스에 대한 고려가 중심적으로 되어 있을 뿐 클라이언트/서버 환경을 위한 세가지 컴퓨팅 요소의 특성을 반영한 객체들은 아니다.

본 설계 기법에서는 객체들을 영역 객체(Domain Object), 폼 객체(Form Object), 통제 객체(Control Object)로 분류한다. 영역 객체는 현실 세계(문제 영역)에 존재하는 객체들로서 정보 중심의 객체들을 표현한다. 즉, 경영

정보 시스템의 경우는 대상 영역에서 발견되는 객체가 대부분 정보를 표현하는 개체(Entity)가 된다. 따라서 클라이언트/서버 컴퓨팅 요소 중의 하나인 데이타베이스 로직은 영역 객체 모형을 통해서 표현된다. 폼 객체는 사용자가 시스템과 상호작용하는 부분에 대한 내용을 다루는 객체로 클라이언트/서버 컴퓨팅 요소 중의 표현 로직을 표현한다. 폼 객체 모형화는 GUI의 모든 구성 요소를 개별적인 객체로 모형화 하지는 않고 구성 요소들의 집합 객체(Aggregated Object)인 폼을 사용하여 GUI 객체를 추상화하여 작성한다. 통제 객체는 응용 분야의 경영 규칙, 정책 등에 대한 내용들과 폼 객체들의 통제, 사용자로부터 들어오는 외부 이벤트의 해석 등을 표현하기 위한 객체로서 클라이언트/서버 컴퓨팅 요소 중 응용 로직을 표현한다. 영역 객체가 문제 공간에 존재하여 발견되는 객체인 것과는 달리 폼 객체와 통제 객체는 해 공간에 존재하는 객체들로서 설계 단계에서 규명되고 모형화되는 객체들이다.

객체 지향 설계의 어려움은 분석과 설계의 대상이 되는 공간에 대한 치밀한 고찰의 부족에서 파생된다[14]. 클라이언트/서버 정보 시스템의 특성을 반영한 객체의 분류는 분석과 설계의 대상이 되는 객체를 각각 다르게 인식함에 따라 대상 공간에서 객체의 역할에 따른 책임성을 명확하게 규정할 수 있다. 또한 그들 간의 관계를 명확히 함으로써 객체 모형의 완결성과 유연성을 확보할 수 있다. 이들 객체들의 규명 방법 및 절차는 다음 장에서 자세히 기술한다.

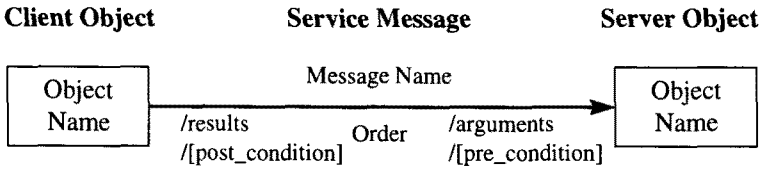
2.3 동적 모형

클라이언트/서버 환경의 특성 중에 하나는 분산된 객체들이 자신이 수행하는 역할에 따라 시스템 내에서 일부분의 컴퓨팅 처리를 수행한다는 것이다. 객체들이 분산됨에 따라 객체들 간의 명확한 정적인 관계 및 동적인 관계의 표현이 필수적으로 요구된다. 객체 지향 방법론에서 동적 모형으로는 전통적으로 상태 전이 모형(State-Transition Model)이 사용되어 왔다. 그러나 상태 전이 모형은 단일 객체에 대한 행태를 보여 줄 뿐, 타 객체와의 동적인 관계를 표현하지는 못한다. 객체간의 관계를 보여 주는 모형으로는 이벤트 추적도(Event Trace Diagram)[20], 상호작용도(Interaction Diagram)[3, 4, 7, 8, 11] 등이 많이 사용되고 있다. 그러나 이러한 모형들은 객체들 간의 이벤트 흐름(Mechanism)[3]만을 고려할 뿐 이벤트의 결과나 객체들의 이벤트에 대한 반응을 고려하지는 않는다. 또한 모형을 인스턴스(Instance) 수준에서 작성하기 때문에 시나리오에 따라 매우 많은 모형을 작성하게 될 뿐더러 객체 모형과의 일관성이 유지되지 못한다.

본 설계 기법에서는 동적 모형으로 상태 전이도와 서비스 요구/반응도(Service Request/Response Diagram)를 제시한다. 상태 전이도는 단위 객체들의 내부 행태를 표현하고 서비스 요구/반응도는 객체들간의 메시지 전달 관계를 클래스 수준에서 표현한다. 객체들간의

주고받는 메시지는 수신 객체가 정해지지 않은 Broadcasting 메시지와 특정한 객체에게 특정한 서비스를 받을 목적으로 전달되는 메시지가 있다. 전자의 경우 객체간의 동적인 관계 모형화가 의미가 없는 반면, 후자의 경우는 특정한 서비스에 대해서 객체들에 대한 요구와 반응을 포함하고 있다. 클라이언트 객체로부터 서비스 요구를 받은 서버 객체는 그에 상응하는 서비스를 클라이언트 객체에게 제공해야 한다. 즉, 서버 객체는 메시지에 대한 책임성을 갖는다. 객체의 책임성이 객체가 수행해야 할 의무를 의미하는 것과 같이 메시지의 책임성은 메시지가 수행해야 할 의무를 표현한다. 이러한 메시지의 책임성을 표현하기 위해 서비스의 요구와 그 서비스에 대한 반응을 하나의 메시지에 함께 표현하도록 한다. 이러한 표현은 객체간에 이루어지는 메시지 전달(Passing) 관계를 체계적이고 명확하게 한다. 따라서 기존의 객체들간의 동적인 관계를 표현하는 모형들이 하나의 요구 사항에 대한 만족성(Satisfiablity)을 검증하기 위해서는 계속적인 다른 메시지의 추적(Navigation)이 필요함에 비해 서비스 요구 반응도는 점진적으로 요구 사항의 만족성을 검증할 수 있다.

서비스 요구/반응도는 객체간의 통신이 메시지를 통해 서비스를 요구하는 클라이언트 객체와 그 메시지를 해석하여 이에 상응하는 서비스를 제공하는 서버 객체간에 이루어지는 것을 클래스 수준에서 표현하는 것이다. 서비스 메시지는 서비스를 요구하기 위한 선행조건(Pre-condition), 서비스 요구 시 서버 객체에 주어지는 변수들(Arguments), 서비스 처리 후 만족해야 하는 후행 조건(Post-condition), 서비스 처리 후 돌려주는 변수들(Results)을 갖는다. 서비스 요구 반응도의 구조물(Constructs)은 [그림 1]과 같다. 그림에서 순서(Order)는 메시지간의 선후 관계를 표현하기 위한 것으로 Dewey Decimal 형태로 표현한다.



[그림 1] 서비스 요구/반응도

서비스 요구/반응도가 다른 방법론들의 동적 모형 표현 도구와 크게 다른점은 다이아그램에서 구성물들이 다른 방법론의 동적 표현 도구와 같이 네트워크 구조를 같지만 그 안에 루프(Loop)가 존재하지 않는 트리(Tree) 구조를 갖는다는 것이다. 이는 해당 이벤트의 트리거(Trigger)에 따른 메시지의 추적을 용이하게 함으로써 해당 이벤트에 대한 객체들간의

협력 관계 및 객체들의 이벤트에 따른 반응 표현을 명확하게 한다. 또한 객체간의 관계가 클래스 수준에서 표현되기 때문에 객체 모형과의 일관성을 유지하고 객체 모형이 갖는 추상화의 장점을 그대로 동적 모형에서도 수용할 수 있다.

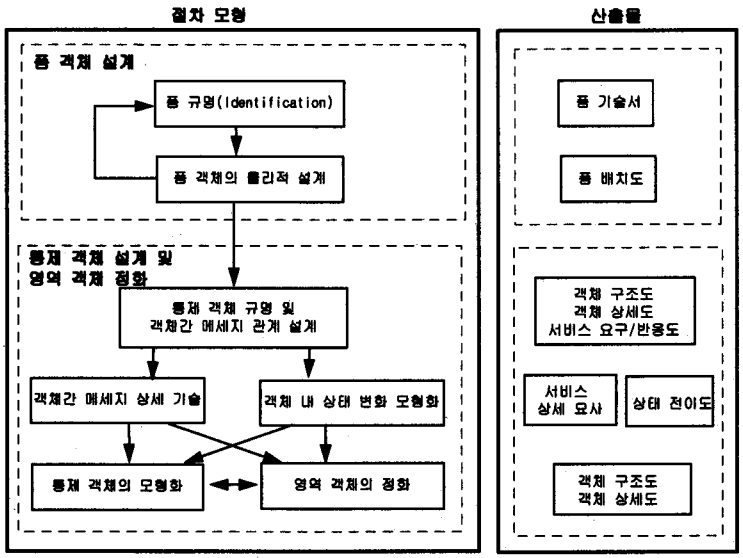
Ⅲ. 객체 지향 설계

3.1 개 요

객체 지향 개발 방법론에서 설계 단계의 작업들은 시스템 설계와 객체 설계로 나누어 볼 수 있다[18]. 시스템 설계 작업은 시스템 구현 도구의 결정, 시스템 아키텍춰의 결정 등의 시스템 개발에 관한 주요한 의사결정을 수행하는 단계이다. 반면에 객체 설계는 분석 단계에서 파악된 객체를 기초로 하여 문제 해결을 위한 객체 모형을 작성하는 것이다. 객체 설계는 문제의 해결 방안에 대한 표현이기 때문에 설계자의 아트(Art)적인 내용이 많이 포함되게 된다. 따라서 설계 단계의 주된 문제점은 객체 설계에 내포되어 있기 때문에 본 논문에서는 시스템 설계 작업 보다는 객체 설계 작업에 초점을 맞추어 기술한다.

본 논문이 제시하는 설계 단계는 분석 단계에서 규명된 영역 객체들을 기초로 하여 폼 객체와 통제 객체를 발명하고, 그들 간의 동적인 관계에 대한 모형을 통해 각 객체들의 속성 및 오퍼레이션을 설계한다. 또한 분석 단계에서 정의된 영역 객체들은 오퍼레이션 및 속성의 명확한 규명 및 개량이 이루어진다. 본 설계 기법에서의 객체 설계 절차와 단계별 산출물은 [그림 2]와 같다. 객체 설계 절차는 크게 (1)

객체 설계 세부 절차 및 단계별 산출물



[그림 2] 객체 설계 단계 및 산출물

폼 객체의 설계와 (2) 통제 객체 설계 및 영역 객체의 정화로 구분된다. 각 단계에서 다음 단계로의 전이는 이전 단계의 작업 결과를 바탕으로 해당 단계에서 수행해야 하는 작업에만 집중할 수 있도록 구조화 되어 있다. 설계 절차의 소개를 위해서 예제로 제시되는 그림은 인텔리전트 캠퍼스 과제를 통해 수행된 학사 시스템의 수강 신청에 관한 부분이며 설계 단계 이전에 수행되어야 하는 영역 객체 모형화 작업은 수행된 것으로 가정한다.

3.2 폼 객체 설계

폼 객체 설계는 시스템과 액터의 인터페이스에 대한 모형화 작업을 의미하고 폼의 규명 작업과 폼 객체의 물리적 설계 작업으로 구분된다.

3.2.1 폼 규명

본 설계 기법에서 객체 설계의 시작은 폼 객체의 규명에서 시작된다. 폼 객체를 우선적으로 규명하는 것은, 폼 객체가 다른 설계 객체에 비해 업무 분석(Task Analysis)의 결과를 통해서 비교적 쉽게 규명될 수 있기 때문이다. 폼 객체는 특정 액터(Actor)가 특정 업무를 수행하기 위한 작업 영역의 역할을 하는 객체들의 집합을 의미한다. 폼 객체의 개요적 기술을 위해서, 폼이 처리해야 하는 외부 이벤트, 주요 기능, 표시 항목(display items), 관련 폼 등을 기술하는 폼 기술서(Form Description)를 작성한다. [그림 3]은 수강 신청 폼의 폼 기술서

폼 기술서			
대상 시스템	학사 시스템	문제영역	수강
작성자/작성일	임홍순 95/05/08	화면명(객체명)	수강신청화면(w_att_select)
설명	학생이 수강신청을 하기위한 화면		
입력이벤트	수강과목신청, 수강과목변경, 수강과목삭제, 수강과목 임시저장, 수강신청완료		
기능	- 수강과목신청, 수강과목 변경 - 개설교과목, 수강신청과목, 시간표 표시		
표시항목 및 출력사항	개설교과목{구분,과목번호,분반,과목명,강실학,담당교수,요일,시간} 수강신청과목{과목번호, 분반, 교과목명, 학점, 담당교수} 시간표		
관련 폼	수강신청 변경화면, 수강신청 취소화면, 학생별 수강신청 화면, 학생별 수강신청 취소화면		

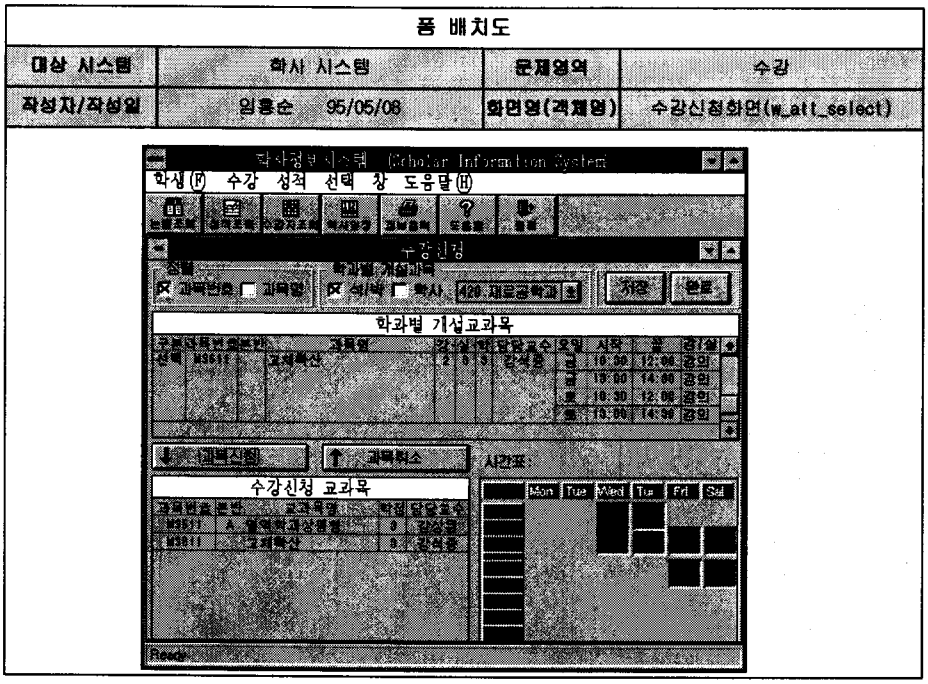
[그림 3] 폼 기술서

이다. 수강 신청 폼이 처리해야 하는 주요 외부 이벤트는 수강 과목 신청, 신청 과목 변경, 수강 과목 삭제, 수강 신청 완료 등이고 수강 신청 폼이 보여 주는 자료는 개설 교과목 목록, 수강신청과목 목록, 시간표 등이다.

3.2.1 폼 객체의 물리적 설계

폼 규명 다음 단계에서는 폼의 물리적 형태를 개략적으로 설계한다. 폼의 물리적 설계는 폼 규명을 통해 작성된 폼 기술서를 바탕으로 수행되게 된다. 즉, 폼 객체의 물리적 설계는 폼 기술서에 기술된 해당 폼이 사용자에게 보여 주어야 하는 정보와 사용자로부터 들어오는 이벤트를 처리하기 위한 폼 구성 요소의 집합(Aggregation)을 설계하는 것이다. 이 단계의 산출물로 나오는 것은 [그림 4]와 같은 폼 배치도(Form Layout)이다. 폼 배치도 작성시 유의할 점은, 폼 배치도 작성 작업의 목적이 폼에 필요한 자료 요소가 무엇인지, 폼 전체적인 사용자 인터페이스의 형태를 어떻게 가져 갈 것인지(예를 들어, MDI(Multiple Document Interface)로 할 것인지, SDI(Single Document Interface)로 할 것인지)에 대한 결정이므로 폼의 그래픽한 요소들, 즉 폰트 크기, 컨트롤의 위치 및 색상 등에 관한 의사 결정에 많은 시간을 소모하지 않도록 하는 것이다.

[그림 4]는 수강 신청 화면의 폼 배치도이다. 수강 신청 화면은 자료 항목을 보여 주는



[그림 4] 폼 배치도

개설 교과목, 수강 과목에 대한 목록과 수강 과목의 신청, 변경, 삭제 버튼들로 구성되어 있다. 그림의 우 하단에는 현재 수강 신청한 교과목의 시간표가 있다.

3.3 통제 객체의 설계 및 영역 객체의 정화 단계

이 단계에서는 (1) 통제 객체들을 규명하고, (2) 객체들간의 메시지 전달 관계와 객체의 상태 전이도 작성을 통해서 통제 객체를 상세하게 설계하고, (3) 이와 관련된 영역 객체들을 정화한다. 이 단계를 통해서 객체 지향 시스템 개발을 위해 필요한 모든 객체에 대한 설계 작업이 완결되지만 일회적으로 모든 작업이 완결되지는 못하고 반복적(Iterative) 작업을 통해 계속적인 모형의 개량 및 정화가 필요하다.

3.3.1 통제 객체의 규명

통제 객체는 주로 응용 로직(Application Logic)을 가지고 있는 객체이다. 통제 객체의 규명은 폼 객체와 마찬가지로 해 공간을 대상으로 발견되는 객체가 아니라 발명되는 객체이기 때문에 통제 객체의 발명이 객관적이고 일관성있기 위해서는 해 공간의 응용로직이 어떻게 구성될 수 있는 지를 먼저 파악해야 한다. 클라이언트/서버 정보 시스템을 위한 통제 객체는 사용자 인터페이스를 통제하는 객체와 업무 규칙을 가지고 있는 통제 객체로 구분될 수 있다. 클라이언트/서버 시스템에서 통제 객체를 규명하기 위해서 아래와 같은 지침(Guideline)을 활용한다.

(1) 하나의 업무 프로세스를 하나의 통제 객체로 정의한다.

하나의 업무 프로세스 자체를 통제 객체로 규명할 수 있다. 하나의 업무 프로세스에는 여러 영역 객체가 모여서 업무를 수행하고 그 속에서 나타나는 업무 규정, 정책 등은 대부분의 경우 어느 단일 객체의 속성값이나 오퍼레이션으로는 표현할 수 없다. 즉, 객체들은 특정한 일을 수행하기 위해서 집합(Collection)을 이루고, 그 집합 속에 참여하는 객체들은 각각 자신들의 역할을 수행하지만 단일한 참여 객체만으로는 집합 자체의 Semantic을 표현하지는 못한다. 예를 들면, 수강 신청 프로세스에는 수강, 학생, 교과목, 학과 등의 객체가 모여서 수강 신청 업무를 수행하고 수강 규정은 학생의 과정, 구분 등과 수강 교과목의 구분, 소속 학과 등의 값들이 모여서 규정을 이룬다. 따라서 수강 신청 프로세스 객체는 각 영역 객체들의 속성값들을 조회하고 그 결과를 바탕으로 수강 규정을 발현시킬 수 있는 객체로 규명될 수 있다. 즉 수강 신청 프로세스 객체는 수강 규정에 따라 관련된 영역 객체들의 생성, 소멸, 갱신 등을 통제한다.

(2) 시스템의 주요 기능 중심으로 사용자 인터페이스를 통제하는 객체를 통제 객체로 정의한다.

업무 프로세스를 통제하기 위한 객체가 영역 객체들의 집합에 대한 Semantic을 표현하는

것과 마찬가지로 폼 객체들의 집합에 대한 Semantic을 표현하는 객체가 필요하다. 폼 객체가 특정한 집합을 이루는 근거는 시스템의 주요 기능에 따른다. 즉, 해당 업무를 처리하기 위한 시스템의 기능에 따라 폼 객체들이 집합을 이룬다. 예를 들면, 수강 신청 처리를 위해서는 관련 폼들의 연결, 외부 사건의 해석, 자료의 표현 규정에 대한 결정 등을 수행하는 통제 객체를 규명할 수 있다. 사용자로부터 수강 신청 외부 사건이 들어오면 사용자의 신분을 확인하여 사용자가 수강 신청이 가능한 학생인지 여부를 판단하고, 해당 학생에게 맞는 수강 신청 화면을 활성화시키며, 해당 학생의 수강 신청을 위한 적절한 자료 값을 수강 신청 화면에 전달해 주는 객체가 필요하다. 이러한 사용자 인터페이스를 처리하는 객체는 주로 시스템의 기능 단위로 정의가 가능하다.

(3) 데이터베이스 영역에서의 뷰(view)를 하나의 통제 객체로 정의한다.

또 하나의 통제 객체로는 데이터베이스 영역에서 언급되는 뷰(View)가 있다. 응용 시스템에서 뷰를 생성하는 것은 그 뷰를 생성하기 위한 목적이 있고 뷰 객체는 이 목적을 위한 속성과 오퍼레이션을 갖는다. 예를 들면, 수강 신청을 위한 개설 교과목을 뷰로 만들었을 때 이 객체의 오퍼레이션으로는 뷰의 생성, 소멸, 소팅, 필터링 등의 일반적인 것들과 수강 신청자에 따른 적절한 개설 교과목 자료 표시, 해당 개설 교과목에 대한 수강 신청 요구 등이 있다. 영역 객체와는 달리 뷰 객체는 영역 객체의 인스턴스 집합(Set)을 하나의 속성값으로 가질 수 있다. 예를 들면, 개설 교과목 뷰 객체의 인스턴스는 여기에 포함되는 교과목 영역 객체 인스턴스들의 자료 집합을 속성값으로 갖는다. 뷰 객체의 규명에는 폼 배치도가 유용하게 사용된다. 폼에 표시되는 객체의 뷰들을 통제 객체로 정의한다. 하지만 하나의 뷰 객체를 여러 개의 폼 객체가 함께 사용할 수 있다.

(4) 시스템 관리를 위해 필요한 경우 통제 객체를 정의한다.

마지막으로 시스템 관리를 위한 통제 객체를 규명할 수 있다. 시스템 관리는 예외 사항 처리(Exception Handling), 서버와의 연결(Server Connection)등을 위한 객체들이다.

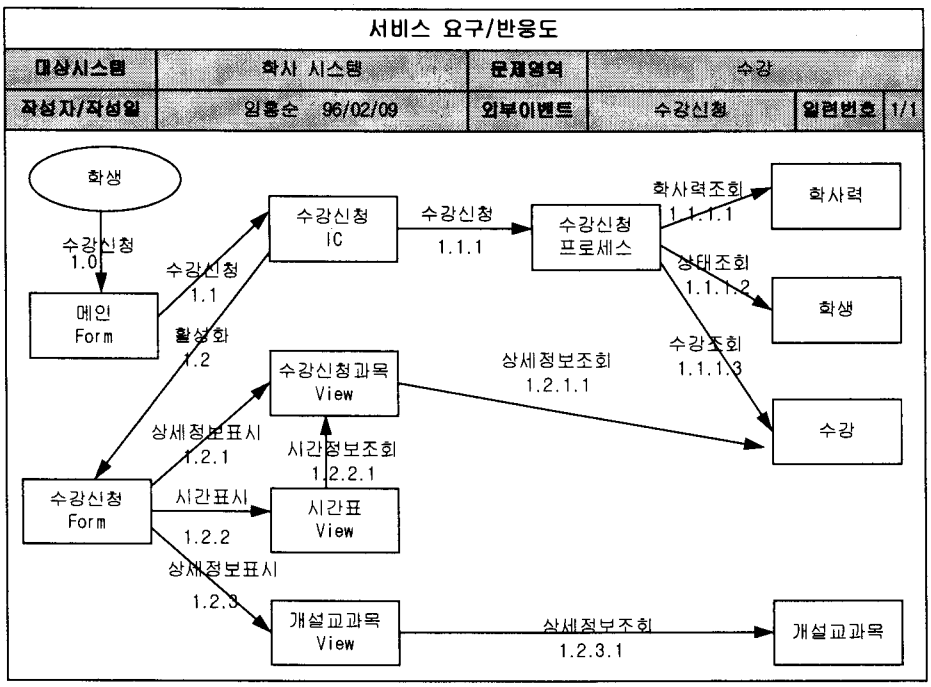
위에서 제시한 지침에 따라 주요한 통제 객체들을 정의할 수 있다. 예를 들면, 수강 신청을 위한 통제 객체들은 다음과 같이 규명된다. 업무 프로세스를 위한 객체로서 수강 신청 프로세스, 인터페이스 통제를 위한 수강 신청 IC (Interface Control), 그리고 뷰 객체들로는 수강 신청 화면에 기초하여 수강신청과목 View, 시간표 View, 개설 교과목 View 객체가 규명된다.

영역 객체와 더불어 폼 객체, 통제 객체가 규명되면 설계 단계를 위한 객체 규명 작업이 완결된다. 그러나 객체 규명 작업이 객체의 속성 및 오퍼레이션에 대한 규명 작업까지 보장하지는 않는다. 규명된 객체들의 속성 및 오퍼레이션은 다음 절에서 설명될 서비스 요구/반응 모형에 의해 체계적으로 규명되고 정화된다.

3.3.2 서비스 요구/반응 모형 작성

정의된 객체를 바탕으로 폼 객체의 주요 외부 이벤트에 따라 서비스 요구/반응도를 작성함으로써 속성과 오퍼레이션을 포함한 통제 객체의 정의가 보다 명확해진다. 앞 장에서 설명했듯이 서비스 요구/반응도는 객체간의 메시지 전달 관계를 표현하는 구조물이다. 따라서 모형을 작성하기 전에 객체들간의 협력관계에 대한 시나리오를 준비하는 것은 모형의 작성에 많은 도움을 준다. 그런데 시나리오는 객체들의 인스턴스 수준에서 작성되는 반면 서비스 요구/반응도는 클래스 수준에서 작성되기 때문에 시나리오에 대한 가시적 표현(Visual Representation)이 서비스 요구/반응도는 아니다.

[그림 5]는 수강신청 이벤트에 대한 서비스 요구/반응도이다. [그림 5]는 학생 액터로 부터 수강 신청이라는 이벤트를 시스템이 받았을 때 보여지는 객체간의 메시지 전달관계를 보여주고 있다. 사용자로부터 수강 신청 이벤트를 받은 메인 폼은 수강 신청과 관련된 사용자 인터페이스를 통제하기 위한 수강 신청 IC(Interface Control) 객체에게 이 이벤트를 전달한다. 수강 신청IC 객체는 다시 이벤트를 수강 신청 프로세스 객체에게 전달하고 수강 신청 프로세스 객체는 수강 신청 이벤트에 대해서 수강 신청 기간 여부, 학생 상태, 수강 과목에 대한 규정을 검토하여 그 결과를 다시 수강 신청 IC 객체에게 전달한다. 수강 신청IC 객체는 수강 신청 프로세스 객체로부터 돌려 받은 서비



[그림 5] 서비스 요구/반응도

스의 결과를 해석하여 수강 신청 조건을 만족시켰을 때 수강신청 폼을 활성화시키고 필요한 자료 값을 넘겨준다.

수강신청 폼은 폼 활성화 이벤트를 받아 세 개의 View 객체에게 자료 표시에 대한 이벤트를 전달하고 각각의 View 객체는 영역 객체 또는 다른 View객체로 부터 자료 값을 조회하고 그 결과를 서비스를 요구한 객체에게 돌려준다. 메시지의 선후 관계는 Dewey Decimal형태로 표현된다. 예를 들면, [그림 5]에서 수강신청 메시지 1.1.1은 학사력 조회 메시지 1.1.1.1, 상태 조회 메시지 1.1.1.2, 수강 조회 메시지 1.1.1.3을 발현(Invoke)시키며 메시지 1.1.1.1은 메시지 1.1.1.2에 선행된다.

서비스 요구/반응도를 통해서 액터가 시스템에 요구한 하나의 메시지에 대해서 각 객체의 역할을 살펴볼 수 있고 그 메시지의 결과가 다시 액터에게 전달되는 메카니즘을 볼 수 있다. 서비스 요구/반응도는 각 객체들이 제공해야 하는 서비스(메소드)와 함께 서비스의 입출력 변수들을 정의하는 역할을 한다. 이러한 서비스에 대한 자세한 내용들은 서비스 상세 묘사에 표현된다. 또한 서비스 요구/반응도는 객체를 메시지 처리에 대한 블랙박스(Black Box) 관점으로 다루어 객체의 외부에서 들어오고 나가는 이벤트를 명확하게 규정할 수 있기 때문에 객체 상태 전이 모형을 작성하는 기초로 사용된다.

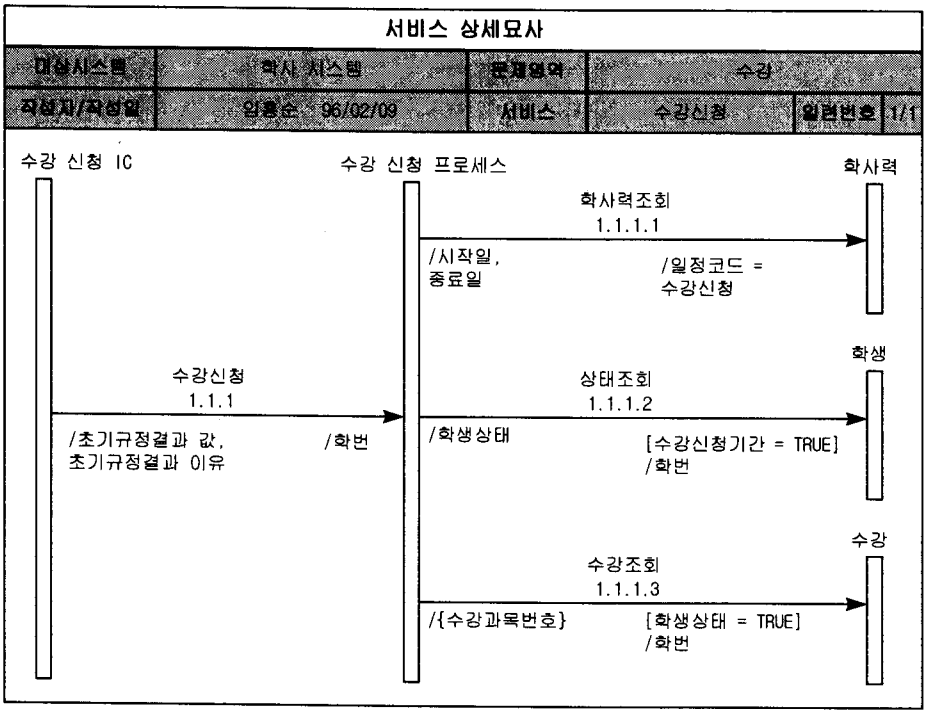
3.3.3 서비스 상세 묘사 및 객체 상태 전이 모형 작성

서비스 요구/반응도에 표현된 서비스는 서비스 이름과 순서(Order) 만이 표시되고 서비스 상세 묘사에서 각각의 서비스 메시지에 대한 변수 값, 조건, 결과값들이 표현된다. 예를 들면, [그림 6]은 [그림 5]의 수강 신청 프로세스 객체에서 영역 객체들인 학사력, 학생, 수강으로 연결되는 서비스 메시지의 상세 묘사이다. 수강 신청 프로세스 객체에서 영역 객체로 연결되는 서비스 메시지는 학생이 수강 신청을 할 수 있는 조건인 수강신청기간에 대한 규정, 재학생에 대한 규정, 기 수강 여부에 대한 검사를 수행하기 위해 요구되는 메시지이다.

객체들의 상태 변화를 표현하기 위해서는 상태 전이도를 사용한다. 상태 전이도의 표현 방식은 다른 방법론에서 사용하는 일반적인 상태 전이도와 동일하므로 별도로 소개하지 않도록 한다.

3.3.4 통제 객체 및 영역 객체의 정화

설계 단계에서 수행한 작업들은 궁극적으로는 완결성과 일관성, 유연성을 갖춘 객체 모형을 작성하기 위함이다. 따라서 서비스 요구/반응 모형 및 객체 상태 전이 모형 등의 동적 모형을 통해 각 객체들이 개량되고 정화된다. 즉, 객체 모형에서 개괄적으로 규명된 객체의 속성 및 오퍼레이션들에 대한 정의가 동적 모형으로부터 검증받고 수용된다. 객체의 속성에 대해서는 속성의 존재 여부, 자료 형태(Data Type) 등에 대한 검증이 수행되고 오퍼레이션에 대해서는 오퍼레이션의 존재 여부, 입출력 인수(Input Output Parameter)의 유무, 입



[그림 6] 서비스 상세 묘사

객체 속성 상세 정보					
문제 영역	수강		객체	개설교과목	
작성자/작성일	임용순 95/07/26		일련번호	1/1	
#	속성이름(한글)	속성이름(영문)	Type	Size	비고
1	개설년도	lecture_year	int	4	
2	개설학기	lecture_term	int	1	
3	강의시간	lecture_time	date		요일과 시간
4	분반번호	lecture_class	char	2	
5	학점	creadits	int	3	
6	강의설명	about_lecture	varchar	255	
7	강의계획	syllabus	blob		
8					
9					
10					
11					
12					

[그림 7] 객체 속성 상세 정보

출력 인수의 자료 형태, 반응 이벤트, 오퍼레이션이 발현되기 위한 이전 조건(Pre-condition), 오퍼레이션이 발현된 후의 이후 조건(Post-condition) 등에 대한 검증 및 보완 작업이 수행된다.

각 객체의 속성과 메소드 등에 대한 자세한 정보들은 객체 속성 상세 정보와 객체 오퍼레이션 상세 정보에서 표현된다. 실제로 분석 단계에서 도출된 영역 객체들도 설계 단계의 작업들을 통해서 정화되고 보완되게 된다. [그림 7]과 영역 객체인 개설 교과목 객체의 객체 속성 상세 정보의 예이고 [그림 8]은 수강 신청 프로세스 객체의 오퍼레이션 상세 정보의 예이다.

객체 오퍼레이션 상세 정보			
문제영역	수강	객체명	수강 신청 프로세스(course_registration_pro)
작성자/작성일	임홍순 95/07/26	일련번호	1/5
이름(영문이름)	수강 신청 (course_registration_pro.register_course)		
설명	수강신청에 대한 메세지가 들어 왔을 때 수강 신청을 위한 초기 규정으로 수강신청기간, 학생상태, 수강 과목의 유무에 의하여 이미 수강신청이 완료되었는 가를 체크하여 결과를 리턴한다.		
반응 이벤트	수강신청		
이전 조건 (Pre-condition)			
입력값	학번		
변환내용 (Transformation)	수강신청의 초기 규정을 체크한다 1. if 수강 신청 기간 시작일 <= Today() <= 수강 신청 기간 종료일, then true. 2. if 학생상태 = 재학생, then true. 3. if 수강과목 =/ null, then true		
출력값	초기규정 결과 값, 초기규정 결과 이유		
이후 조건 (Post-condition)			

[그림 8] 객체 오퍼레이션 상세 정보

[그림 8]과 같은 객체 오퍼레이션 상세 정보의 작성은 [그림 6]과 같은 서비스 상세 묘사의 내용을 집약하여 표현하는 것으로써, 일부 내용에 있어서는 자동적인 정보의 전이가 이루어진다. 예를 들면, [그림 8]의 입력 값과 출력 값, 학번과 초기규정결과 값, 초기규정결과 이유는 [그림 6]의 서비스 메시지 수강신청(1.1.1)의 입력 변수 값(Arguments)과 출력 변수 값(Results)의 내용이 객체 오퍼레이션 상세 정보로 자동적으로 전이되어 기술된 것이다.

객체 오퍼레이션의 상세 정보에서 이전 조건과 이후 조건은 입출력 값과 같이 자동적으로 생성되지는 않는다. 서비스 상세 묘사에 기술된 이전 조건이 상황에 독립적으로 항상 필요

하다면 객체 오퍼레이션 상세 정보의 이전 조건에 기술되지만 특정한 상황에서만 이전 조건이 필요한 경우에는 이전 조건이 표현된 메시지를 보내는 클라이언트 객체의 오퍼레이션 상세 정보의 변환 내용의 알고리즘에 포함되어 기술된다. 예를 들면, [그림 6]의 서비스 상세 묘사에서 상태조회(1.1.1.2)와 수강조회(1.1.1.3) 메시지에 표현된 [수강신청기간=TRUE]와 [학생상태=TRUE]라는 이전 조건은 상태조회(1.1.1.2)와 수강조회(1.1.1.3)에 항상 필요한 조건이 아니라 수강 신청(1.1.1) 메시지에 의해서 발현될 경우만 필요한 조건인 것이다. 따라서 [수강신청기간=TRUE]와 [학생상태=TRUE] 조건은 각각 학생 객체와 수강 객체의 오퍼레이션 상세 정보로 표현되는 것이 아니라 [그림 8]과 같이 수강 신청 프로세스 객체의 수강 신청 오퍼레이션 상세 정보의 변환 내용에 알고리즘으로 삽입된다.

클라이언트/서버 정보 시스템 설계 마지막 단계에서 중요한 작업은 정의된 객체들의 적절한 분산이다. 대부분의 경우, 영역 객체들은 데이터베이스 서버에 할당된다. 폼 객체와 인터페이스를 위한 통제 객체는 클라이언트에 할당된다. 업무 규칙에 관한 통제 객체는 가능한 응용 서버(Application Server)에 할당되도록 한다. 이는 업무 규칙이 가변성을 가지고 있으므로, 업무 규칙이 변화했을 때, 클라이언트 모듈을 재분배하지 않고, 서버 한 곳의 모듈을 변경하도록 하기 위함이다. 따라서 객체의 분류는 분석과 설계에 있어서 객체의 규명을 체계적으로 지원할 뿐만 아니라 체계적인 객체의 분산을 위해서도 활용된다.

지금까지 논의한 객체 지향 설계 기법은 물리적인 구현 환경에 독립적인 내용들로 구성되어 있다. 그러나 설계 작업에는 물리적인 환경 요소가 고려되어야 한다. 즉 시스템 개발을 위한 구현 도구, 하드웨어, 네트워크, 프로토콜 등의 결정은 시스템 구현에 많은 영향을 미치게 된다. 예를 들면, 구현 도구 중의 하나로 관계형 데이타베이스를 사용하는 것과 객체 지향 데이타베이스를 사용하는 것은 객체 구현에 많은 영향을 미치게 된다. 다음 장에서는 객체 설계의 산출물인 객체 모형이 어떻게 구현으로 연결되는지에 대해서 인텔리전트 캠퍼스 과제의 사례를 중심으로 소개한다.

Ⅳ. 객체 지향 설계에서 구현으로 연결

구현은 설계를 바탕으로 고객에게 시스템을 전달할 수 있는 형태(Deliverables)로 만드는 것이다. 객체 지향 설계를 코딩으로 연결하는 가장 좋은 방법은 객체 지향 프로그래밍 언어를 사용함으로써 시스템 개발의 일관성을 유지하는 것이다. 클라이언트/서버 환경에서는 일반적으로 클라이언트 개발 도구와 서버 개발 도구가 다르게 된다. 설계 산출물을 코딩으로 연결시키기 위해서는 개발 도구들이 제공하는 구조물에 설계 단계에서 정의된 객체들을 Mapping하는 방안이 결정되어야 한다.

인텔리전트 캠퍼스 프로젝트에서 학사 시스

템의 경우, 클라이언트 개발 도구로는 PowerBuilder[25]를 사용하였고 서버 개발 도구로는 Sybase의 OpenServer[24]와 Stored Procedure를 활용하였다. 사용자 인터페이스와 관련된 통제 객체들은 PowerBuilder의 User Defined Object 형태로 구현되었으며, 업무 규정을 담고 있는 객체들은 서버에 OpenServer나 Stored Procedure 형태로 구현되었다. 영역 객체들은 테이블로 구현되었고 영역 객체의 오퍼레이션은 Stored Procedure로 구현되었으며 폼 객체는 PowrBuilder의 윈도우를 포함한 GUI 구성요소(Widget) 클래스로 구현되었다.

예를 들면, [그림 9]는 [그림 5]의 수강 신청IC 객체에 대한 구현 예이다. uo_course_apply_ic는 수강신청 IC객체의 이름이고 "Instance Variables"로 선언된 'student_no', 'bool_term', 'course_complete' 등은 객체의 속성이다. 객체의 메소드는 "Subroutine" 형태로 표현된다. 그림에서 표현한 "Subroutine"은 수강 신청 이벤트가 수강 신청 IC객체로 들어올 때 수행하는 메소드인 'uof_initial_course_apply'를 나타낸다. 수강 신청 이벤트는 [그림5]의 서비스 요구/반응도에서 메인 폼 객체에서 수강 신청 IC객체로 전달되는 수강신청(1.1) 이벤트를 의미한다.

```
User Object : uo_course_apply_ic

Instance Variables
long student_no//수강학생의 학번
boolean bool_term, course_complete//수강신청기간, 수강완료
End of Instance variables

public subroutine uof_initial_course_apply
/*****************************************************************************************
*  Description : 수강 신청 프로세스 객체에게 수강신청 서비스를 전달하고 결과값을 해석하여 규정이          *
*                만족되었을 경우 수강신청 창을 Open한다.                                          *
*                if res_val = 1, then success                                             *
*                else fail                                                                *
*****************************************************************************************/
int res_val
string res_reason


//프로세스 객체에게 이벤트 전달
CONNECT ;
    Declare check_initial_rule PROCEDURE FOR course_registration_pro.register_course
        @student_no= : student_no ;
IF SQLCA.SQLCODE=0 TEHN
    EXECUTE check_initial_rule ;
    IF SQLCA.SQLCODE=0 TEHN
        FETCH check_initial_rule INTO : res_val, : res_reason
    ELSE
        messagebox("학사시스템", SQLCA.SQLERRTEXT)
```

```
    END IF
ELSE
    messagebox("학사시스템", SQLCA.SQLERRTEXT)
END IF
CLOSE check_initial_rule ;
DISCONNECT ;

//서비스 결과 해석
IF res_val <> 1 THEN
    Opensheet(w_att_select, w_sholar_main, 7, layered!)
ELSE
    messagebox("학사 시스템", res_reason)
END IF
end subroutine
.....
```

[그림 9] PowerBuilder로 구현한 통제 객체

[그림 10]은 수강 신청 프로세스 객체의 메소드를 Stored Procedure로 구현한 예이다. Stored Procedure의 이름은 "."(도트)를 중심으로 객체의 이름이 앞에 오고 해당 객체의 메소드 이름이 뒤에 오는 형식으로 구성되어 있다. [그림 10]은 [그림 8]의 객체 오퍼레이션 상세 정보의 내용을 구현한 것으로서 'course_registration_pro'(수강 신청 프로세스) 객체가 수강 신청 이벤트를 받았을 경우에 수행하는 메소드를 'register_course'라는 이름으로 나타내고 있다. 이때의 수강 신청 이벤트는 [그림 5]의 서비스 요구/반응도에서 수강 신청 IC객체에서 수강 신청 프로세스 객체로 전달되는 '수강신청(1.1.1)' 이벤트를 의미한다. [그림 8]의 입력 값, '학번'과 출력 값, '초기규정결과 값'과 '초기규정결과 이유'가 '@student_no'와 '@res_val'과 '@res_reason'으로 정의되었고 변환 내용에 대한 내용이 그림에서 'as'이후로 코딩되었다.

```
create course_registration_pro.register_course
@student_no int,
@res_val int out,
@res_reason varchar(255) out
as
dec are
@s  tart datetime,
@s_end datetime,
@status tinyint,
@complete tinyint
exec s_schedule.retrieve_schedule 7, @s_start out, @s_end out//7:수강신청 기간 코드값.
if @s_start >= getdate( ) and @s_end <= getdate( )//success about duration rule
```

```
begin
select @status=exec s_student.retrieve_status @student_no
if @status=1//success about being student
        begin
        select @complete=exec s_course.retrieve_student @student_no
        if @complete is not null//success about completeness of applying course
                begin
                select @res_val=1
                select @res_reason="모든 규정이 만족 되었습니다."
                end
        else
                begin
                select @res_val=-1
                select @res_reason=수강신청을 이미 완료했습니다.
                end
        end
else
        begin
                select @res_val=-2
                select @res_reason="재학생만 수강신청 할 수 있습니다."
                end
        end
else
        begin
        select @res_val=-3
        select @res_reason="수강신청 기간이 아닙니다."
        end
select @res_val, @res_reason
```

[그림 10] Stored Procedure로 구현한 통제 객체

V. 결론 및 향후 연구 방향

클라이언트/서버 정보 시스템의 개발은 환경의 복잡성으로 인해 체계적인 시스템 개발 방법론을 요구한다. 객체 지향 개발 방법론은 생산성 및 품질을 위한 기반 기술로 인정받을 뿐 아니라 복잡한 환경에 유용하게 적용될 수 있는 방법론으로 알려져 왔지만 실제적인 적용에는 많은 어려움이 지적되어 왔다. 특히, 설계 단계에 있어서 객체 규명의 어려움과 개발 절차의 체계적인 지원 부족이 지적되어 왔다.

본 논문에서는 객체 지향 개발 방법론을 활용하여 클라이언트/서버 정보 시스템 개발 시 설계 단계를 체계적으로 지원하기 위한 객체 지향 설계 기법을 제시하였다. 이를 위해서 기존의 객체 지향 개념틀에 클라이언트/서버 환경의 특성을 반영해 설계 모형을 개발 할 수 있도록 추가적인 개념 및 구조물을 제시하고 체계적인 설계를 수행할 수 있는 절차를 제시하였다.

본 설계 기법에서는 클라이언트/서버 정보

시스템의 특성을 반영하여 객체를 영역(Domain) 객체, 통제(Control) 객체, 폼(Form) 객체로 구분하였다. 영역 객체는 분석 단계에서 문제 공간(Problem Space)을 대상으로 규명된 객체들로서 문제 영역의 지식들을 가지고 있고 통제 객체와 폼 객체는 설계 단계에서 해 공간(Solution Space)을 대상으로 규명되는 객체들로서 응용 논리에 대한 내용을 가지고 있다. 객체간의 메시지 전달에 의한 동적인 관계를 표현하기 위해 서비스 요구/반응 모형(Service Request/Response Model)이 제시되었다. 서비스 요구/반응 모형은 메시지의 책임성(Responsibility)을 보장하기 위해 서비스 요구와 서비스의 결과가 한 메시지안에 표현되도록 만들어졌다. 또한 객체 모형과의 일관성을 유지하기 위해 클래스 수준에서 서비스 요구/반응 모형이 작성되도록 만들어졌다. 설계 절차는 체계적이고 일관된 객체 규명에서 시작하여 점진적이고 반복적인 절차를 통해 설계 작업이 완성될 수 있도록 제시되었다. 특히 객체의 규명을 위한 객체의 정의 지침을 제시하여 설계자의 임의성을 배제하고 일관된 객체 규명 작업이 수행되도록 하였다.

본 논문에서 제시된 객체 설계 기법을 인텔리전트 캠퍼스 과제에 적용하면서 얻은 경험은, 이러한 체계적인 설계 기법 및 절차가 개발 단계의 자연스러운 전이(Seamless Transition), 모형의 완결성을 통한 재사용성 촉진 등에 따른 생산성 증진 뿐만 아니라 모형의 일관성, 유지 보수 등의 품질 보증 측면에서도 필수적이다는 것이다. 향후 연구 과제로는 본 설계 기법을 분산 환경의 대규모 시스템 개발에 활용하기 위한 설계 기법의 Scalability를 증진시키는 방안과 분산 환경의 Semantic을 충분히 묘사하기 위한 비동기성(Asynchronous), 동시성(Concurrence) 등을 함께 표현할 수 있는 구조물에 대한 연구가 추가적으로 필요하다.

참 고 문 헌

[1] 박성주, 임홍순, 김종우, CAMIS 개발 방법론, Technical Report, 한국과학기술원 경영정보연구센터, 1995.

[2] Berson, A, Client/Server Architecture, McGraw-Hill, 1992.

[3] Booch, G., Object-Oriented Analysis and Design with Application, Benjamin/Cummings, 1994.

[4] Booch, G. and J. Rumbaugh, Unified Method, Version 0.8, Rational Inc., 1995.

[5] Booch, G., J. Rumbaugh, and I. Jacobson, The Unified Modeling Language for Object-Oriented Development, Version 0.9, Rational Inc., 1996.

[6] Coad, P. and E. Yourdan, Object-oriented Analysis, Prentice Hall, NJ, 1990.

[7] Coleman, D., et al, Object-Oriented Development : The Fusion Method, Prentice-Hall, 1994.

[8] Filman, R.E, W.S. Faught, and J. Solomon, The Object-Oriented Development of a Transaction-Processing Application, Journal of Object-Oriented Programming, November-December, 1992, pp.51-60.

[9] Firesmith, D., Use Cases : The Pros and Cons, Report on Object Analysis and Design, Vol. 2, No. 2, 1995, pp. 2-6.

[10] Henderson-Seller, B. and J.M, Edwards, Booktwo of Object Oriented Knowledge : The Working Object, Prentice-Hall, 1994.

[11] Jacobson, I., M. Christerson, P. Jonsson, G. Overgaard, Object-Oriented Software Engineering. A Use Case Driven Approach, Addison-Wesley, 1992.

[12] Graham, I., Object-Oriented Methods, Addison-Wesley, 1995

[13] Krasner, G.E. and S.T. Pope, A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80, Journal of Object-Oriented Programming, Vol. 1, No. 3, 1988, pp. 26-48.

[14] Martin, J. and J.J. Odell, Object-oriented Analysis and Design, Prentice-Hall, NJ, 1992.

[15] Monarchi, D. E. and G.I. Puhr, A Research Typology for Object-Oriented Analysis and Design, Communication of the ACM, Vol. 35, No. 9, September 1992.

[16] Pancake, C. M, The Promise and the Cost of Object Technology : A Five-Year Forecast, Communication of the ACM, Vol. 38, No. 10, October, 1995, pp. 33-49.

[17] Park, J.Y. and S.J. Park, University Re-engineering using a BPR Supporting Tool, Proceeding of the Third European Academic Conference on Business Process Redesign, UK, 1996.

[18] Park, S.J. and J.W. Kim, Intelligent Campus : Integration of Digital Library and Campus Information Systems, Proceedings of International Symposium on Digital Libraries 1995, August 22-25, 1995, Japan.

[19] Rosenberg, D., "Using the Object Modeling Technique with Objectory for Client/Server Development," Object Magazine, November-Decem-

ber, 1993, pp. 54-57.

[20] Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, Object-Oriented Modeling & Design, Prentice-Hall, 1991.

[21] Rumbaugh, J., OMT : The Development Process, Journal of Object-Oriented Programming, Vol. 8, No. 2, May, 1995, pp. 8-16, 76.

[22] Shedletsky, J.J. ; Rofrano, J.J., "Application Reference Designs for Distributed Systems," IBM Systems Journal, Vol. 32, No. 4 , 1993. pp. 625-646.

[23] Trampoulidis, K.X. and K.N. Agavanalcis, Object Interaction Diagram : a New Technique in Object-Oriented Analysis and Design, Journal of Object-Oriented Programming, June, 1995, pp.25-32, 39.

[24] Vaskevitch, D., Client/Server Strategies. A Survival Guide for Corporate Reengineers, IDG Books, 1993.

[25] Yourdon, E., K. Whitehead, J. Thomann, K. Oppel, and P. Nevermann, Mainstream Objects : An Analysis and Design Approach for Business, Yourdon Press, 1995.

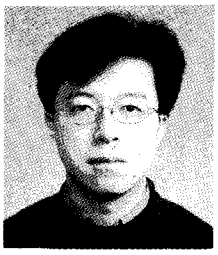
[26] _, SYBASE OpenServer-Library/C, Sybase Inc., 1993.

[27] _, PowerBuilder-Building Application, Powersoft Inc., 1994.

◇ 저자소개 ◇

공동저자 박성주는 서울대 산업공학 공학사, 한국과학기술원 산업공학 공학석사를 취득하고 미국 미시간 주립 대학교에서 시스템 공학을 전공하여 공학박사를 취득하였다. 현재 한국과학기술원 테크노경영대학원 교수와 한국과학기술원 경영정보연구센터 소장으로 재직하고 있다. 주요 관심분야는 경영정보시스템, 정보시스템 통합, CSCW, 객체지향기술, 경영혁신기술, 시뮬레이션 등이다.

공동저자 김종우는 서울대 수학과 이학사, 한국과학기술원 경영과학 공학석사를 취득하고 한국과학기술원 산업경영학 공학박사를 취득하였다. 한국과학기술원 경영정보연구센터 연수연구원으로 근무하였다. 현재 충남대학교 통계학과에서 전임강사로 재직하고 있다. 주요 관심분야는 경영정보시스템, 의사결정 지원시스템, 객체지향 개발방법론, CSCW, 업무흐름 모형화, 전자도서관 등이다.



공동저자 임홍순은 한국과학기술원에서 경영과학 학사 및 석사를 취득하였다. 현재 한국과학기술원 경영정보연구센터에 재직하고 있고 한국과학기술원 테크노경영대학원 박사과정에 재학하고 있다. 주요 관심분야는 경영정보시스템, 객체지향기술, 시스템 개발방법론, 시스템 통합, CASE, 경영혁신기술, CSCW 등이다.