□특 집□

객체지향 실시간 시스템 개발 방법론

숭실대학교 전자계산학과 이광용 · 류성열**
숭실대학교 소프트웨어공학과 정 기 원**

목 차

Ⅰ. 서 론

대형 소프트웨어 시스템들은 상호 복잡하게 얽혀 있어, 이해하기가 어렵고 비결정적인 요소(nondeterminitic facotor)들이 많아 어려움은 많다. 이러한 시스템의 대표적인 것으로 실시간 시스템이 있는데, 실시간 시스템은 공정제어, 산업 자동화, 의학 혹은 과학 연구, 컴퓨터 그래픽, LAN 혹은 WAN 분야, 군수산업, 원자력 분야 등 우리 생활의 간단한 분야에서부터 복잡한 분야에 이르기까지 폭넓게 응용되고 있다[2,11].

소프트웨어 개발자는 이러한 시스템들을 개발하기 위해 시스템 내부에 존재하는 병행성을 식별하여 시스템의 복잡성을 경감시키고자 하지만 시스템 내부에 존재하는 비결정적인 요소로 인해 병행 프로세스간의 예기치 못한 상호작용을 초래하게 되므로서 이 시스템의 개발의 어려움을 증가시키고 있다. 이러한 시스템을 개발하는 개발자는 복잡성을 경감시키기 위한 병행 프로세스 식별방법 뿐만아니라, 이들 프로세스간의 상호작용에 대한 엄격하고 정밀한 분석 방법이 필요하게 되었다.

실시간 시스템은 그 시스템에서 요구된 기능들을 주어진 시간제약 범위안에서 모두 수행하되 시스템의 외부환경과의 상호작용이 올바르고 믿을만하게 이루어지도록 제어하는 시스템을 말한다. 이러한 시스템을 개발함에 있어 주요 고려사항은 외부환경과의 실시간 통신 문제. 외부에서의 요구가 짧은 시간에 집중적으로 몰려올 때의 우선처리 문제, 시스템의 오류 발생시 대응책, 병행처리 태스크들의 처리문제 등 취급해야할 문제들은 다양하다[3,10].

현재의 실시간 소프트웨어 분석/설계 방법론은 크게 자료흐름중심 방법[1,3,7,8,10,18,19,20]과 객체중심 방법[4,5,6,12,16,17]의 두 가지로 분류할 수 있다.

자료흐름중심의 실시간 소프트웨어 개발방법들은

* 준회원
** 종신회원

병행처리 프로세스와 병행처리 프로세스 간의 통신과 동기화 문제, 프로세스간의 통신 우선순위 문제에 대해서는 적절한 방법과 도구를 제시하고 있다. 그러나, 프로세스 신뢰성 측면의 오류검출, 오류회복, 예외사항 처리 등의 문제, 소프트웨어 재사용과 유지보수 측면에 대해서는 소홀히 취급하고 있다[1,3,7,8,10,18,19,20].

객체중심 실시간 소프트웨어 개발방법들은 객체지향 개념을 적용하여 시스템을 체계적으로 개발하므로서 복잡성을 상당히 해결하고 있으며, 소프트웨어 개발 프로젝트의 대규모화, 프로그래밍 언어의 고수준화, 유지보수해야 할 소프트웨어 양의 증가에 따른 위기를 극복하는데 중요한 역할을 하고 있다. 그러나, 이 방법들 역시 설계방법보다는 분석에 치중하고 있고 분석 중에서도 객체 구조화에 역점을 두고 있어 실시간 소프트웨어와 같은 시스템을 개발을 위해서는 아직은 부족한 면이 많이 남아있다[4,5,6,12,16,17].

그러므로, 객체지향 실시간 시스템 방법론은 분석단계에서는 객체모델을 중심으로 한 동적, 기능 모델을 개발하나 실시간 시스템의 시간특성이 잘 반영되도록 동적모델의 특성이 추가적으로 강조되어야 하고, 설계단계에서는 실시간 소프트웨어 개발문제들을 해결하기 위해, 실시간 객체지향 모델들을 적극 활용하는 방안이 필요하다. 실시간 객체지향 설계모델에는 객체의 병행프로세스와 이들간의 통신과 동기화 문제, 프로세스 스케줄링을 다루는 모델이 될 것이다.

본 고에서는 이러한 자료흐름 중심의 실시간 소프트웨어 개발방법들과 객체지향적 실시간 소프트웨어 개발방법들의 기본개념과 절차, 표기법에 대해 고찰하고, 이들 방법들이 실시간 소프트웨어 개발을 위해 어떤 작업을 수행하고 있는지 살펴보고, 객체지향적 실시간 소프트웨어 개발방법의 연구방향에 대해 살펴보겠다.

II. 자료흐름중심 실시간 시스템 개발방법

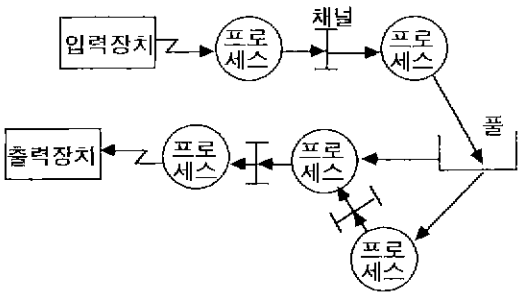
자료흐름중심의 실시간 시스템 개발방법은 MASCOT, Ward & Mellor 방법, Gomaa의 DARTS와 ADARTS, Nielsen & Shumate 방법이 대표적이다. MASCOT는 1970년대 초기에 영국에서 개발되어 MASCOT2, MASCOT3로 버전 업(version up)되었으며, Ward & Mellor 방법은 1980년대 중반에 제안된 방법으로 기존의 구조적 분석/설계 기법에 실시간 시스템의 개발에 적합한 특수한 표기법을 확장한 방법이고, Gomaa의 DARTS와 ADARTS는 기존의 자료흐름 설계 기법에 MASCOT 방법에 근거한 병행처리 그레픽 표현 방법을 적용하여 실시간 특성을 고려하였으며, Ada 기반형 실시간 시스템 개발 방법으로 ADARTS를 제시하였다. 그리고, Nielsen & Shumate 방법은 MASCOT, DARTS, Structured Design, Layerd Virtual Machine 방법들이 함께 통합된 방법이다.

2.1 MASCOT

MASCOT(Module Approach to Software Construction Operation and Test)는 1970년대 초기 영국에서 개발되어 MASCOT2, MASCOT3로 버전 업(version up)이 된 실시간 시스템 설계 방법[1,3]으로서 하드웨어 구성이나 프로그래밍 언어와는 독립적으로 접근할 수 있는 정형적 접근 방법일 뿐아니라 소프트웨어 생명주기 전단계를 지원할 수 있어 사업관리 기법으로 사용되는 방법이기도 하다. MASCOT3는 MASCOT2가 대형 시스템 개발이나 다중 프로세서 구조를 잘 다루지 못한점과 계층적 설계(hierarchical design)와 프로세스 분해(process decomposion)를 지원하지 못한 점을 해결하였다.

MASCOT의 설계방법의 주요개념으로 모듈화(Modularity)를 들 수 있는데, 설계(design), 구조화(construction), 구현(implementation), 시험(testing) 단계 동안에 적용된다. MASCOT의 설계방법은 자료객체와 이 객체들을 조작하는 행위(activity)들을 생성하고, 설계의 주요도구로 자료흐름, 서브시스템, 자료객체(IDA; Intercommunication Data Area), 행위(activity), 채널(channel), 풀(pool), 서버(server)를 이용한다. 자료객체는 채널과 풀로 표현되는데, 채널은 자료객체들을 포함하는 큐(queue)에 해당하고, 풀은 생성, 소멸, 갱신이 가능한 데이타베이스에 해당한다. 그리고, 서버는 외부 하드웨어 장치를 표현한다. 행위들은 채널과 풀을 조작하며, 시스템의 요구기능을 표현하는 병행 프로세스에 해당한다.

설계는 자료흐름을 가지고서 이들로부터 상호작용하는 병행 행위들을 정의하는 것으로부터 시작하여, 그 다음에는 이 병행행위들 간의 자료흐름을 채널과 풀로 표현한다. 이 결과 생성된 행위-채널-풀 네트워크(그림 1)는 개발될 시스템의 상위수준 설계를 표현한 것이다. 상위수준 설계 동안에 식별된 병행 프로세



(그림 1) 행위-채널-풀 네트워크

스들을 행위-채널-풀 네트워크로 다시 상세설계를 하여 전체 시스템 구조를 완성한다.

MASCOT에 의한 6단계의 설계방법을 요약하면 다음과 같다.

1. 하드웨어, 소프트웨어, 시스템 테스트 등의 기본적인 외부 요구사항과 제약사항을 정의한다.
2. 단계 1에서 확인된 소프트웨어 요구사항에 기초를 둔 상위 부분의 설계 제안(design proposal)을 한다.
3. 자료흐름 과정간의 상호작용을 병행처리의 네트워크 형태로 상세 설계한다.
4. 각각의 순차적 프로세스들을 보다 더 상세하게 설계한다.
5. 전 단계에서 명세화 모듈들을 구현한다.
6. 완전한 소프트웨어 시스템으로 통합하고 테스트한다.

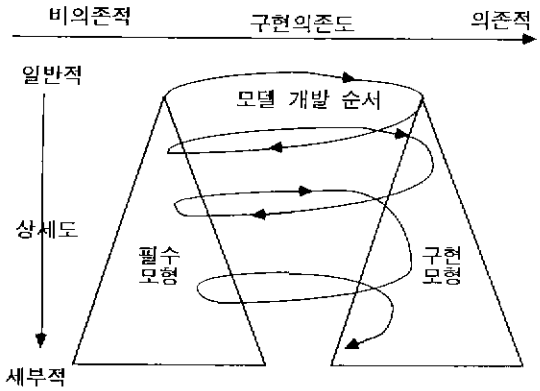
2.2 Ward & Mellor의 방법

1980년대 중반 Ward와 Mellor에 의해 제안된 방법[18,19]으로 기존의 구조적 분석/설계 기법에 실시간 시스템의 분석과 설계에 적합한 특수한 표기법을 확장한 방법이다. 실시간 시스템의 행동적 특성을 더 정확히 표현하려는 의도에서 이 방법은 시작되었다.

Ward와 Mellor의 실시간 시스템 개발 방법의 주요 생각은 (그림 2)와 같다. 필수모형과 구현모형은 일반적인 사항에서부터 점차 세부적인 사항으로 심화되고, 구현 의존도는 구현에 독립적인 것으로부터 의존성이 있는 것으로 발전한다.

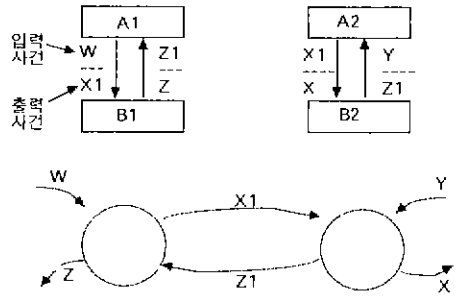
이 방법에서는 (그림 3)과 같이 2가지 측면에서 확장하여 표기하는 방법을 제공하고 있다.

첫째로, 기존의 자료흐름도에서 제어 프로세스/데이타 프로세스, 비연속적인 자료/연속적인 자료, 그



(그림 2) 필수모형과 구현모형의 관계

(a) 확장된 자료흐름도 표기법



(b) 상태전이도와 제이프로세스

(그림 3) Ward와 Mellor의 확장된 자료흐름도 표기법

리고 일반 자료저장소/버퍼로 구분하였으며, 제어흐름을 신호/활성화/비활성화 흐름으로 구분하였다. 둘째로, 자료흐름도의 제어 프로세스와 제어흐름으로부터 상태전이도(상태전이표)를 모델링하는 방법을 제공하였다.

이 방법에 의한 실시간 시스템 개발절차는 필수모형 구축과 구현모형 구축으로 대별된다. 필수모형은 환경모델과 행위모델로 구성되는데, 환경모델은 시스템과 시스템 외부와의 상호작용을 기술하며, 행위모델은 이 시스템에서 요구되는 행위를 정의한다. 또한 구현모형은 프로세서모델, 태스크모델, 모듈모델로 구

성되어 있는데, 프로세서모델은 프로세서(processor)들과 이들 사이의 인터페이스를 묘사한 모델이고, 태스크모델은 태스크들과 태스크들 사이의 인터페이슬 보여주고, 모듈모델은 필수모형의 결과를 모듈로 할당한 모델이다.

구조적 분석 단계인 필수모형을 작성하는 단계는 다음 4단계로 요약할 수 있다.

1. 문맥도(context diagram)를 그린다.
2. 외부사건목록(external event list)을 작성한다.
3. 자료, 자료흐름, 제어흐름이 포함된 자료흐름도를 연속적으로 세분화하고, 자료흐름의 명세를 작성한다.
4. 상태전이도(state transition diagram)를 그린다.

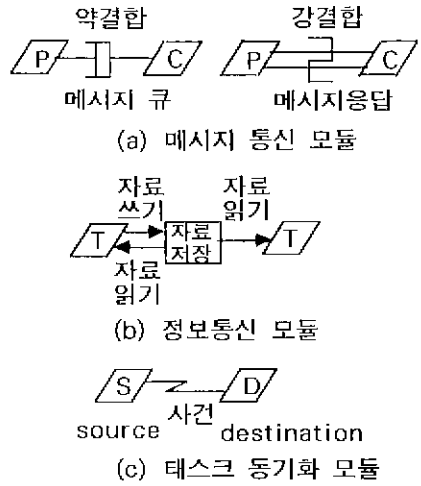
구현 모형을 작성하는 단계는 다음 5단계로 구성되어 있다.

1. 자료흐름도를 이용하여 기능들을 프로세서(processor)와 태스크, 모듈들에 할당한다.
2. 자료흐름도로부터 구조도(structure chart)를 작성한다.
3. 각 모듈의 명세를 작성한다.
4. 각 모듈의 구성을 재사용성, 복잡성, 결합도, 응집도의 관점에서 구조도 설계를 평가한다.
5. 목표 프로그래밍 언어로 작성한다.

2.3 Gomaa의 DARTS와 ADARTS 방법

1984년 Hassen Gomaa는 기존의 자료흐름 설계 기법에 MASCOT 방법에 근거한 병행처리 그래픽 표현 방법을 적용하여 실시간 특성을 고려한 설계 기법인 DARTS 방법[7]을 제안하였다. 그후 1989년에는 Ada 기반형 실시간 시스템 개발방법으로 ADARTS[8]를 개발하였다. 이 방법에서는 미 해군 연구소(NRL; Naval Research Laboratory)에서 사용한 정보은닉 모듈 구성방법과 객체지향설계 방법에서의 객체구성방법을 결합한 방법으로 모듈구조와 기준을 적용하여 소프트웨어 구성요소의 유지보수성과 재사용성을 달성했으며, 실시간 시스템을 병형처리 태스크로 나누기 위해 DARTS 방법의 병행처리 태스크 구조화 기준을 적용했다.

DARTS 방법에서는 먼저 자료흐름도 분석개념에 입각해서 소프트웨어 분석을 한다. 이로써, 자료흐름도가 생성되고 자료사전이 정의되고 시스템과 외부 객체와의 인터페이스가 확립된다. 그 다음으로 이 자료흐름도에는 비동기적인 병행 태스크들을 표시하고 있지 않기 때문에 DARTS 방법의 주안점은 이들



(그림 4) DARTS의 태스크 통신과 동기화

자료흐름도로부터 실시간 시스템 태스크들을 식별하는데 두고 있다. 실시간 태스크들을 식별하는 휴리스틱으로 Gomaa는 입출력 의존성이 있는 기능들, 시간이 긴요한(time critical) 기능들, 계산량이 많은 기능들(computational function), 기능적 응집력이 있는 기능들, 시간적 응집력이 있는 기능들, 주기적인 실행이 필요한 기능들로 정의하고 있다. 이 기준들은 자료흐름도의 몇개의 기능(transform)이 모여서 하나의 태스크로 구성할 때 적용한다. 이 기준들을 적용하여 태스크들이 정의가 되면, DARTS 방법에서는 다음 (그림 4)와 같은 태스크들 사이의 정보교환과 동기화를 위한 태스크 인터페이스 모듈(태스크 통신 모듈과 태스크 동기화 모듈)을 작성한다. 태스크 통신 모듈은 메시지 통신 모듈과 정보은닉 모듈의 두가지 범주로 구성되어 있다. 메시지 통신 모듈은 메시지 큐를 다루는 방법을 제공하고 정보은닉 모듈은 자료 풀 혹은 자료 저장소에 접근하는 방법을 제공한다. 한편, 태스크들 사이에 자료대신에 제어가 전달될 때에는 태스크 동기화 모듈을 이용한다.

한편, ADARTS 방법에서는 모듈과 태스크 구조화를 통해 NRL, OOD, DARTS 방법들의 강점을 결합하였다. ADARTS의 주안점을 실시간 시스템을 병행처리 태스크들과 정보은닉 모듈로 분해하는 원리를 제공하는데 있다. 이 원리는 태스크 구조화 기준과 모듈구조화 기준에 나타나 있다. 태스크 구조화 기준은 사건 의존성 기준, 태스크 응집 기준, 태스크 우선순위 기준의 세가지 범주로 분류되는데, 사건 의존성 기준은 태스크가 언제, 어떻게 활성화되어야

할지 결정하기 위해 디바이스의 입출력 의존성, 주기적인 시간, 주기적인 I/O, 제어기능, 사용자 인터페이스 의존성을 적용한다. 그리고, 태스크 의존성 기준은 계층적 분해로 인해 수많은 함수들이 식별되므로써 발생하는 복잡성을 해결하기 위해 이 함수들의 비동기적인 성질을 식별하는 기준으로 순서적 응집, 시간적 응집, 기능적 응집 기준들을 적용하고 있다. 또한, 태스크 우선순위 기준은 시간적 긴요도(time critical)에 따라 우선순위를 달리할 때 적용한다. 시간적으로 긴요한 태스크는 실행에 있어 높은 우선순위를 두고, 주로 계산만 하는 태스크들은 낮은 우선순위를 부여한다. 한편, 모듈 구조화 기준으로 NRL 방법의 정보은닉 모듈과 OOD 방법[4]의 객체구성 방법을 결합한 방법으로 디바이스 인터페이스 모듈과 자료 추상화 모듈로 자료흐름도의 기능들을 할당하고 남은 나머지 기능들에 대하여 행위은닉(behavior hiding) 혹은 소프트웨어 의사결정 기준에 따라 모듈로 분할한다.

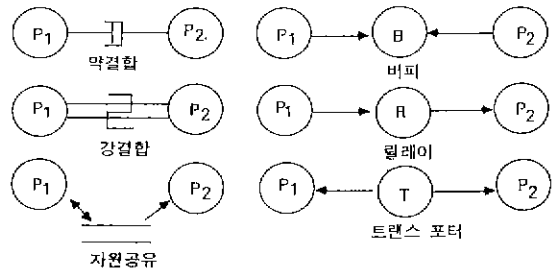
ADARTS에 의한 설계방법은 다음과 같고, 단계 1에서 3과 단계 5는 일반적인 실시간 방법에서 적용될 수 있으나, 단계 4는 Ada에 기반을 둔 실시간 시스템에만 적용된다.

1. 실시간 구조적 분석 명세서를 개발한다.
2. 태스크 구조화 기준을 적용하여 병행처리 태스크들을 식별한다.
3. 모듈 구조화 기준을 이용하여 시스템에 모듈들을 식별한다.
4. Ada 지원 태스크들이 추가 및 Ada 태스크 인터페이스 정의 등의 Ada 중심 설계를 행한다.
5. 태스크와 모듈들에 대한 구성품 인터페이스 명세서를 정의한다. 구성품 인터페이스 명세서에는 각각의 구성품에 대한 외부 관점을 표현한다.

2.4 Nielsen & Shumate의 방법

이 방법[10]은 1988년 Nielsen과 Shumate가 제안한 것으로 MASCOT, DARTS, Structured Design, Layerd Virtual Machine 방법들이 함께 통합된 방법이다. 이 방법은 병행 시스템이나 순차적 시스템에 모두 적용가능하며 Ada를 구현언어로 사용하였다.

이 방법의 주요개념은 처리해야 할 중심부분(middle part) 분해, 프로세스 인터페이스 결정, 중간 프로세스 도입과 프로세스의 호출/피호출 관계 정의, Ada 패키지로 패키지화, 대규모 태스크의 분해, 객체지향 개념이다.



(그림 5) 프로세스 인터페이스와 중간 프로세스

하드웨어와 직접적으로 인터페이스 하는 부분에 프로세스를 할당하므로써 그 시스템에서 처리해야될 중심 기능만 남게되는데, 중심분해는 이 부분을 자료흐름, 제어흐름, 프로세스들로 분해하는 것을 의미한다. 프로세스 인터페이스 결정(그림 5a)은 프로세스들 사이의 약결합과 강결합, 전역 데이타 자원의 이용에 대한 정의를 하는 것이다. 중간 프로세스 도입(그림 5b)과 프로세스 간의 호출/피호출 관계 정의는 먼저, 프로세스들 도입하게 되는데, 이 프로세스들은 버퍼(buffer), 릴레이(relay), 트랜스포터(transporter)다. 중간 프로세스가 결정이 되면 태스크들 사이의 호출/피호출 관계 정의는 기계적이다. 버퍼는 생산자와 소비자 태스크들이 서로 호출하는 관계를 표현하고, 릴레이는 생산자 태스크가 이 태스크를 호출하고, 이 태스크는 소비자 태스크를 호출하는 관계를 표현하며, 트랜스포터는 직접 생산자와 소비자 태스크를 직접 호출하여 정보를 전달하는 관계를 표현한다. 또한 식별된 태스크들을 Ada 패키지로 변환하는 것은 재컴파일성, 재사용성, 가시성 등의 모듈화 휴리스틱에 따른다. DARTS의 휴리스틱에 의해 자료흐름도로부터 태스크들을 식별하게 되므로써, 어떤 태스크에는 과중한 기능이 할당될 수도 있으므로, 이 방법에서는 가상기계(virtual machine)와 객체(object)개념을 적용해서 이들의 기능을 분해한다.

Nielsen과 Shumate 방법은 구조적 분석 설계 방법론에서 파생된 방법론으로써 실시간 시스템 개발에 긴요한 병행 프로세스 식별과 프로세스간의 통신 및 동기화를 수용하였다. 이 방법의 개발절차는 다음과 같다.

1. 하드웨어 인터페이스 결정
2. 인터페이스 부분에 프로세스 할당
3. 중심부분분해

4. 병행처리 결정
5. 프로세스 인터페이스 결정
6. 중간프로세스 도입
7. 태스크를 Ada 패키지로 변환
8. 설계결과를 Ada PLD로 변환
9. 대규모 태스크의 분해

III. 객체지향적 실시간 시스템 개발방법

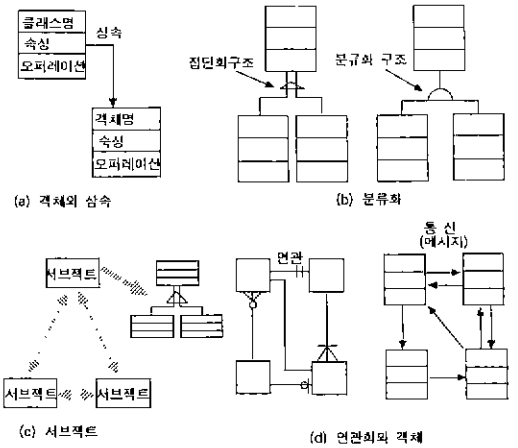
객체중심의 실시간 개발방법은 Coad & Yourdon의 방법, Shlaer & Mellor의 방법, Rumbaugh *et al*의 방법, Booch의 방법이 가장 대표적이다. 이들 방법들이 반드시 실시간 시스템만을 개발대상으로 만들어진 것은 아니지만, 굳이 객체지향적 실시간 시스템 개발방법에 포함시킨 것은 분석/설계의 예로 실시간 시스템을 주로 인용하고 있기 때문이다. Coad & Yourdon의 방법은 순수하게 객체중심의 모델을 구축하려고 노력하였다. 이 방법에서는 객체모델의 주요요소인 객체들과 객체구조, 그리고 속성과 서비스들을 표현하고 있으며, 이에 추가하여 객체간의 통신을 메시지 접속으로 표현하고 있다. Shlaer & Mellor의 방법은 하나의 시스템을 6가지 계층(시스템, 도메인, 서브시스템, 객체, 상태, 기능 계층)으로 분류하여 모델화하고 있으나, 여기에서 개발하는 주요모델은 정보모델, 상태모델, 기능모델이다. Rumbaugh *et al*의 방법은 문제정의로부터 세가지 다른 관점으로 객체, 동적, 기능 모델을 개발하고, 설계에서는 이 모델들을 보다 정제하여 정제된 객체, 동적, 기능 모델을 만드는 것이 특징이다. 이때, 중심이 되는 모델은 객체모델이다. Booch의 방법은 다른 방법들과는 달리 분석과 설계를 따로 구분하지 않았으며, 분석동안에 이용한 객체모델을 설계에 그대로 적용하고 있고, 실시간 시스템의 특성을 다각적으로 표현할 수 있는 여러가지 표기법을 제공하고 있다. 이 방법은 시스템을 논리적인 관점과 물리적인 관점에서 바라보고 있다.

3.1 Coad & Yourdon의 방법

Coad와 Yourdon의 객체지향분석 방법[6]에서는 "객체지향" 개념으로 다음과 같은 정의를 하고 있다.

객체지향=객체+분류화+상속+객체통신

즉, Coad와 Yourdon의 객체지향분석은 객체모델(object model)의 주요요소인 객체(object)들과 분류화



(그림 6) Coad와 Yourdon의 객체지향분석 표기법

(Classificiation)구조, 상속(inheritance) 구조를 표현하는 객체구조(object structure), 그리고 객체들 사이의 서비스 이용을 표현하는 객체통신(Object Communication)으로 구성되어 있다.

Coad와 Yourdon의 객체지향분석 활동은 객체 식별, 구조 식별, 서브젝트 정의, 속성과 연관 관계 정의, 서비스와 메시지 접속 관계 정의의 5단계로 구성되어 있다. 이들 각각의 단계에서는 (그림 6)과 같은 객체지향 표기법을 이용한다.

객체식별 단계 동안에 객체를 식별하는데, 각각의 객체들은 객체명, 속성, 오퍼레이션의 세 부분으로 구성되어 있다. 이들 각각은 다른 객체와의 구별을 위해 객체명을, 객체 자신의 상태와 특성을 표현하기 위한 속성을, 객체 자신이 갖고 있는 기능을 명세화하기 위해 오퍼레이션을 표현한다. 이들 객체들의 하나의 형틀로써 클래스가 정의된다.

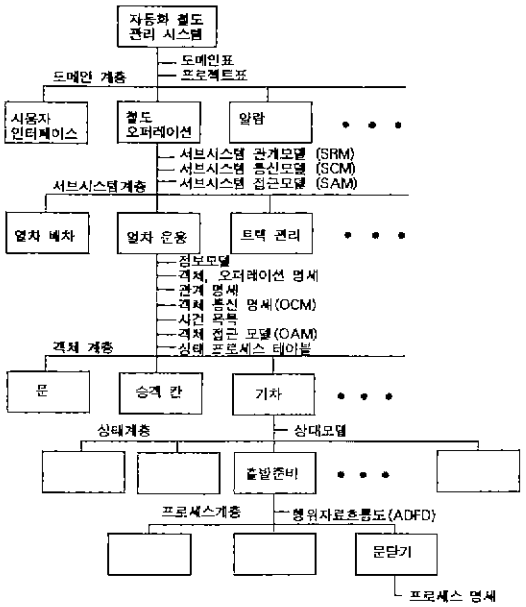
구조 식별 단계에서는 문제의 복잡성을 해결하기 위해 일반화-특수화 개념을 담은 분류화 구조(classification structure)와 전체-부품 개념을 담은 집단화 구조(assembly structure)를 이용한다(그림 6b). 분류화 구조는 객체들의 공통된 속성이나 오퍼레이션들을 상위 계층의 객체로 표현하고, 하위 계층의 객체는 상위 계층의 객체의 속성이나 오퍼레이션을 그대로 승계함과 동시에 자신의 고유 속성이나 오퍼레이션을 지니도록 표현하는 방법이다. 한편, 집단화 구조는 여러개의 객체들로 구성된 전체와 전체의 구성요소인 부품들로 구조화하는 방법이다.

시스템을 성공적으로 개발하기 위해서는 사용자와 분석가 사이의 정보 교환이 원활이 소통되어야 하고,

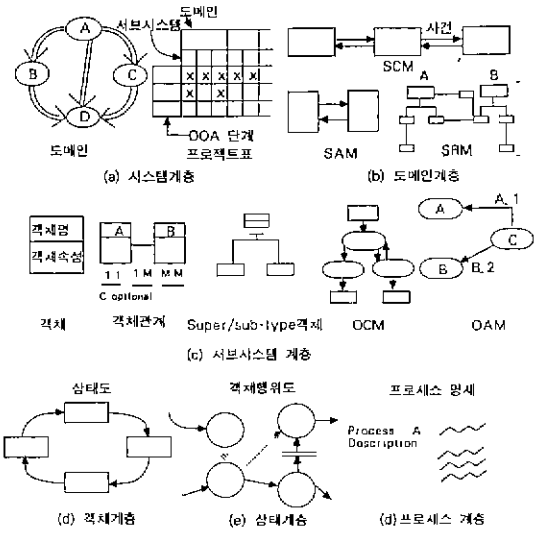
분석가가 취급하기 용이한 크기가 되어야 한다. 서브젝트(subject)(그림 6c)는 분석가가 한번에 취급하기 적당한 분량의 객체들로 분할하는 단위이다. 객체들이 서브젝트별로 분할이 되면 서브젝트들 사이의 메시지(그림 6c)를 표현하여 하나의 완전한 시스템을 형성한다.

속성과 연관관계 정의는 객체별로 식별된 속성들이 객체 구조 속에서 어느 곳에 위치해야 될지를 결정하는 일과 객체들의 연관성(connection)을 관계 다중성(multiplicity)과 함께 표현하는 일을 수행한다. 식별된 속성은 일반화-특수화 개념, 전체-부품 개념에 따라 적절히 분류한다. 정보은닉 속성은 객체내의 정보은닉을 표현하며 객체내의 오퍼레이션에 의해서만 조작이 가능하다. 객체들의 연관성은(그림 6d) 객체의 성질(actor, agent, server)에 따라 다르지만 두 객체사이의 메시지 연결(그림 6d, 실선 화살표)이 있음을 표현한다. 객체의 연관성을 표현할 때는 가능한 최소한의 연결이 이루어지도록 노력해야 한다. 분류화 혹은 집단화 구조에 의해 구조화된 객체들의 연관관계(association)를 표현할 때, 상위 계층에서 객체들의 연관성이 형성되도록 해야한다. 즉, 객체들 사이의 메시지 연결은 이 구조의 상위계층에서 이루어지도록 메시지 추상화(abstraction)가 바람직하다.

서비스 정의는 문제영역(problem domain)에서 시스템이 수행해야 할 역할을 명세화하는 작업이다. 이 역할들의 처리순서에 따라 객체의 동적인 행위를 묘사한 객체생명주기(object life history)를 정의하고, 상태-사건-응답(state, event, response)형태로 시스템 상태와 이 상태에서의 외부 사건, 그리고 이 사건들에 응답하는 서비스를 표시한다. 메시지 접속관계 정의는 객체들 사이의 연관화 구조와 객체들의 메시지 통신(message communcation)구조를 함께 표현하는 활동이다. 이 모델이 완성되면 분석가는 객체 연관성과 메시지 통신 관계를 조사하여 추가적으로 필요한 메시지 접속관계가 존재하고 있는지 검증할 수 있다.



(그림 7) Shlaer와 Mellor의 객체지향



(그림 8) Shlaer와 Mellor의 객체지향 표기법

3.2 Shlaer & Mellor의 방법

Shlaer & Mellor의 객체지향 개발방법[16,17]의 기본생각을 (그림 7)로 요약할 수 있다. Shlaer와 Mellor의 객체지향은 시스템, 도메인, 서브시스템, 객체, 상태, 프로세스의 6가지 계층으로 하나의 대형 시스템을 최상위 시스템 수준으로부터 최하위 수준의 프로세스 수준까지의 하향식파라다임(Top-Down Paradigm)으로 표현하고 있다.

Shlaer와 Mellor의 객체지향 활동은 준비작업 수행, 정보모델 개발, 상태모델 개발, 프로세스 모델 개발의 4단계로 구성되어 있다. 이들 각각의 단계에서는 (그림 8)과 같은 그래픽 표기법, 모델들, 문서를 이용하고 있다.

대형 소프트웨어 시스템을 개발하기 위해 시스템을

몇개의 도메인(domain)으로 분할하여 시스템을 개발하는 것이 개발 어려움을 해결하는데 크게 도움이 될 것이다. Shlaer와 Mellor의 객체지향 활동에서는 시스템을 개발하기에 앞서 준비작업을 한다. 이 동안에는 시스템을 여러 도메인으로 분할하고, 도메인을 여러개의 서브시스템으로 나누어서 도메인 도표(domain chart)와 프로젝트표(project chart)를 만든다. 도메인 도표의 도메인은 타원으로, 이중 화살표(그림 8a, 시스템계층)는 하위 도메인에서 제공되는 기능을 상위 도메인이 이용하는 관계를 표시한다. 프로젝트 표는 열(column)로는 도메인과 서브시스템을, 행(row)으로 개발단계를 표시함으로써 일종의 공정 진행표 역할을 한다.

도메인과 서브 시스템이 준비가 되면 이 도메인과 서브 시스템들을 여러 팀들이 나누어서 개별적으로 개발한다.

정보 모델링에서는 서브시스템을 구성하고 있는 개념적인 개체인 객체들을 식별하고, 객체 관계, 관계 다중성, super/sub 관계를 이용하고 객체들을 분류하고 연결하여 객체 구조를 만든다. 동일 도메인 내에 모든 서브 시스템에 대하여 객체구조가 만들어지면 서브시스템 관계 모델(SRM; Subsystem Relation Model)을 작성한다. 이 모델에서 점선 사각형은 서브시스템은 나타낸다.

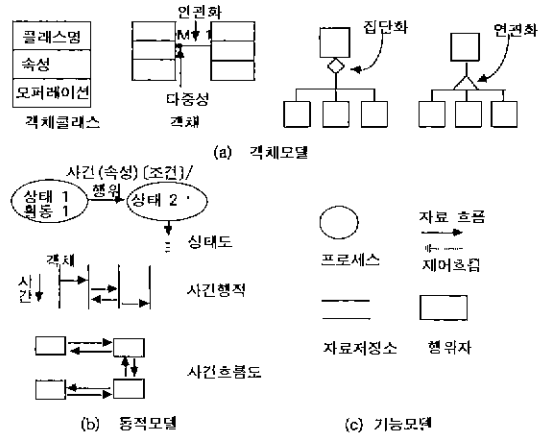
객체 구조가 만들어지면 객체 각각에 대하여 객체의 상태와 사건, 사건에 의한 상태전이, 각각의 상태에서 수행해야 할 행위(action)들을 묘사한 상태모델을(그림 8d, 객체계층) 개발한다. 이와함께, 상태모델이 완성된 후에는 객체들 사이의 협력 구조를 표현하기 위해 객체 통신 모델(OCM; Object Communication Model)을 개발하게 되는데, 이 모델에서 사각형은 외부 객체를 타원은 내부객체, 화살표는 사건 흐름을 표현한다. 또한, 같은 도메인 내에 모든 서브 시스템별로 객체 통신 모델이 완성이 되면 서브 시스템 통신 모델(SCM; Subsystem Communication Model)을 작성한다. 서브 시스템 통신 모델의 사각형은 서브 시스템을, 화살표는 사건들을 표현한다.

상태 모델의 특정 상태에서의 행위들(action)에 대해 자료 흐름도(DFD)가 작성되는데, 이를 행위 자료 흐름도(ADFD; Action Data Flow Diagram)라고 한다. 행위 자료 흐름도는 주료 사용하는 일반 자료흐름도에 제어흐름(그림 8e, 점선 화살표)을 확장하여 표현하고 있다. 이 도표도 상태도, 객체 통신 모델, 서브 시스템 통신 모델 순으로 개발되었듯이 행위 자료 흐름도, 객체 접근 모델(OAM;Object Access Model), 서브시스템 접근모델(SAM;Subsytem Access Model)순으로 개발한다. 객체 접근 모델에서 객체는 타원으로, 다른 객체의 오퍼레이션을 이용하는 관계를 나타내는 객체 접근은 화살표로 표시한다. 서브 시스템 접근 모델도 마찬가지로 서브 시스템들간의 오퍼레이션 이용 관계를 각각 사각형과 화살표로 표시한다.

3.3 Rumbaugh *et al*의 방법

Rumbaugh *et al*의 객체지향[12]에서는 먼저 실세계에 대한 모델링을 하게 되는데 이 과정의 특징으로 추상화(abstraction), 캡슐화(encapsulation), 모듈화(modulation), 계층화(hierarchy)를 들 수 있다.

추상화와 캡슐화 개념은 클래스명, 속성, 오퍼레이션이 함께 표현된 객체 클래스(Object Class)들을 추출할 때 적용이 되고, 모듈화와 계층화는 추출된 클래스들의 시스템 내에서 구조를 표현할 때 이용된다. 특히, 계층화 개념은 분석시에는 연관화, 집단화, 일반화, 설계시에는 시스템의 수평 계층(layer)으로 분할할 때 이용하며, 모듈화 개념은 시스템의 수직 분할(division)로 특징지워진다. 연관화 개념은 생일 파티의 꽃과 촛불의 관계처럼 단순한 연관 관계를 표현하는 것이고, 집단화관계는 연관화 관계가 아주 밀접할 때의 관계를 전체-부품 관계를 표현하는 것이다. 그리고, 일반화 개념은 클리스들 사이의 상속 계층을 표현하는 것이다. 수직 분할은 시스템을 몇 개의 독립된 서브시스템들로 나누는 것을 의미한다. 한편, 계층은 상위계층과 하위 계층사이의 공급자(supplier)



(그림 9) Rumbaugh *et al*의 객체지향 표기법

와 고객(customer) 사이의 관계처럼 하위 계층은 상위 계층에게 서비스를 제공하고 상위 계층은 하위 계층의 서비스를 이용하는 사용자에 해당한다.

Rumbaugh *et al*의 방법은 분석, 설계, 객체설계, 구현의 4단계로 구성이 되어 있다. 분석 단계에서는 (그림 9)와 같은 그래픽 표기법을 이용하고 있다.

객체모델은 객체도라고 하는 도표를 이용하여 시스템의 구조를 파악하고 문서화 되는데, 실세계 문제영역으로부터 객체와 클래스를 추출해 그들간의 관계를 연관화, 집단화, 일반화 관계를 중심으로 규명하며, 여기에 클래스의 속성과 오퍼레이션을 함께 표현하므로서 시스템의 정적인 구조를 파악한다. Rumbaugh *et al*의 객체 모델을 도출하는 절차는 다음과 같이 8단계로 요약할 수 있다.

1. 객체와 클래스 식별
2. 클래스에 대한 자료사전 준비
3. 클래스간의 관계파악
4. 객체 속성 및 링크 관계 파악
5. 상속 관계 구성
6. 모델 검증
7. 모델 반복 정제
8. 클래스들의 모듈화

동적 모델은 상태도라고 하는 도표를 이용하여 시스템의 동적인 행위를 파악하는데, 상태도에서 이용되는 주요 개념은 상태(state)와 사건(event)이다. 사건은 하나의 객체로부터 다른 객체에 자극을 주는 것을 의미하며, 이 자극에 의해 객체의 상태가 변한다. 상태는 특정 시점에서의 객체 속성값을 의미한다. 동적 모델은 먼저 사건들에 대한 시나리오 작성으로부터 시작하여, 객체간의 사건 흐름(event flow)과 사건 행적(event trace)작성, 이들을 이용하여 객체들에 대한 상태도를 수립하고, 마지막으로 객체간의 일관성 검증을 통해 모델이 완성된다.

기능 모델은 자료 흐름도로 표현되는데, 이 도표는 다수의 프로세스, 자료흐름, 제어흐름, 자료저장소, 행위자들로 구성되어 있다. 기능모델을 개발하는 절차는 다음과 같이 5단계로 요약할 수 있다.

1. 외부와 시스템 사이의 입출력 값 파악
2. 시스템의 자료 흐름도 생성
3. 프로세스에 대한 기능 명세서 작성
4. 제약 조건 파악
5. 최적화 기준 명세화

설계 방법에서는 대상 문제 파악에 중점을 둔 분석 단계와는 달리 시스템 구현을 위한 문제 해결 방안을 모색하여야 한다. 시스템 설계 단계 동안에 만들어져야 하는 사항들로 시스템을 서브 시스템으로 분할, 객체들의 병행 수행성과 상호 배타적인 요소(mutual exclusion)들의 판별, 병행 수행되는 서브 시스템들을 특수 목적용 하드웨어 장치, 태스크들에 할당, 축적된 자료의 효율적인 관리, 전역 자원의 접근 방식 결정, 소프트웨어 시스템 제어 방식 결정 등등의 시스템 전반에 걸친 구현 고려사항을 설정한다.

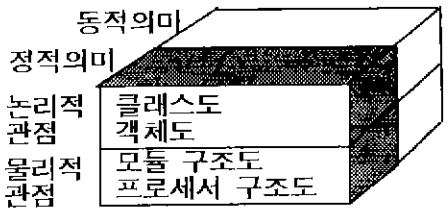
객체 설계 단계에서는 분석 단계에서의 분석 모델과 시스템 설계 단계에서의 세부 구현 사항을 가지고서 구현시에 사용할 클래스들의 추가, 인터페이스의 정의, 알고리즘 정의 등의 작업을 수행한다. 분석 모델을 보강하여 설계 산출로 발전시키는 것이 객체 설계의 목적이므로 설계자는 다음의 작업을 순차적으로 해나간다.

1. 객체, 기능, 동적 모델의 결합
2. 알고리즘 설계
3. 설계의 최적화
4. 제어 방식의 구현
5. 상속성의 조정
6. 연관의 표현
7. 객체의 표현
8. 모듈 생성
9. 문서화

마지막으로, 구현 단계에서는 객체 설계 단계의 세부 객체의 모델, 세부 동적 모델, 세부 기능 모델, 그리고 기타 문서들을 이용하여 시스템으로 구현한다. 구현은 객체 지향 언어를 사용하면 가장 용이한 것이 사실이지만, 비객체지향 언어를 사용하여 구현할 수도 있다. 객체지향 언어로 구현할 때는 클래스 정의, 객체 생성, 연산 호출 생성, 상속성 사용, 클래스 혹은 객체간의 연관 표현에 대해 고려해서 개발하고, 비객체 언어일 경우에는 클래스 자료구조로 변환하는 방법, 다형성 문제의 해결, 객체를 기억 장소로 표현, 상속 관계의 구현, 병행처리 해결 등의 문제에 대한 해결 방안을 모색하여 개발하여야 한다.

3.4 Booch의 방법

Booch 방법[5]의 객체지향에 대한 기본 생각은 (그림 10)으로 요약할 수 있다. 이처럼 여러 관점으로 모델을 구성하는 이유는 하나의 모델만으로는 복잡한 소프트웨어 시스템의 세부사항을 찾아내기가 어렵기 때문이다.



(그림 10) Booch의 객체지향

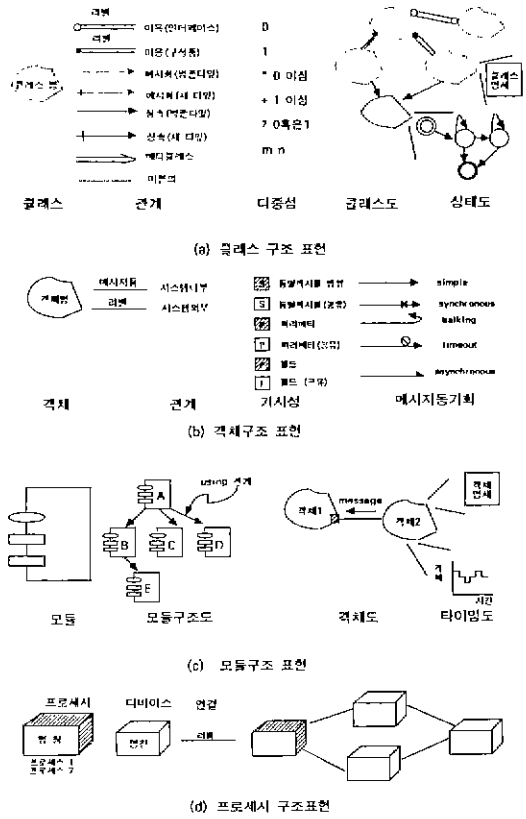
이 그림에서 클래스와 객체는 시스템의 주요 추상체의 식별과 의미 기술을 위해 제공된 도표이고, 시스템의 구현과는 밀접한 관계가 없으므로 시스템의 논리적 관점을 표현한 것도, 모듈 구조나 프로세스 구조는 구현을 위한 소프트웨어와 하드웨어 구성품들을 기술하는데 이용되므로 시스템의 물리적 관점에 해당한다.

이 4가지 모델들 모두 시스템의 정적인 성질만을 표현하고 있으나, 시스템은 정적인 성질만이 있는 것이 아니기 때문에 사건이 발생함으로써 동작해야 일들이 있다. 즉, 객체는 생성, 소멸되며, 객체들도 순서에 입각해서 혹은 병행적으로 다른 객체에 메시지를 전송하기도 한다. 이와 같은 시스템의 동적의미를 표현하기 위해 Booch의 객체지향설계 방법에서는 상태도와 타이밍도를 이용한다.

Booch의 객체지향 활동은 클래스와 객체 식별, 클래스와 객체의 의미 식별, 클래스와 객체의 관계식별, 그리고 구현의 4단계로 구성되어 있다. 이들 각 단계에서는 위 (그림 10)과 같은 관점에서 모델들을 생성하는데 이 모델들의 표기법으로는 (그림 11)과 같다.

클래스와 객체식별 단계에서는 문제 영역으로부터 클래스들과 객체들을 식별하고, 객체들 사이의 행위 메카니즘(그림 11c, 객체도)을 생성하는 활동을 수행한다. 클래스들과 객체들은 응용분야에 전문가에 의해 이용되는 추상체(abstraction)들의 식별로부터 시작하여 정제 과정을 거쳐 만들어진다. 그리고, 행위 메카니즘은 객체들 사이의 협력 구조를 표현한 것으로 자동차의 가속기를 누르게 되면 엔진이 빠르게 작동하게 되고, 가속기를 풀면 엔진이 느리게 작동되는 것처럼 객체들 사이의 행위 의존성을 기술한 것이다. 이로써, 상위 수준의 클래스도와 객체도가 완성된다.

다음 단계는 클래스들과 객체들의 의미 식별 단계로 객체들 사이의 프로토콜(protocol)을 결정한다. 객체들 사이의 프로토콜은 객체의 생성으로부터 소멸까지의



(그림 11) Booch의 객체지향설계 표기법

에 상태와 사건의 흐름으로 표현된다. 객체들 사이의 프로토콜에 의해 클래스나 객체들의 의미가 보다 정제가 되고 상위 수준의 객체도 혹은 클래스도에 새로운 객체나 클래스를 추가하는 기회가 된다. 이때, 위험 요소가 가장 큰 곳에 대해서는 프로토타입(prototype)을 개발할 수 있다.

세번째 단계는 클래스들과 객체들의 관계성 식별 과정으로 두번째 단계에서 발견한 객체들 사이의 프로토콜을 바탕으로 하여 객체의 활용성(using), 상속성(inheritance), 및 기타 관계성을 확립하고, 시스템의 동적인 의미를 객체의 정적/동적 성질, 메시지 동기화, 가시성, 객체의 실행 타이밍 등을 함께 표현하는 과정이다. 이로써, 객체지향설계 동안에 개발하게 되는 논리적 모델(logical model)을 거의 완성한다.

클래스들과 객체들의 구현단계에서는 객체지향설계의 마지막 단계로 앞선 단계들에서 결정한 설계결정(design decision)을 검토하여 휴리스틱한 방법으로

클래스들을 모듈로 할당하고, 프로그램들을 프로세서(processor)에 할당하는 활동을 수행한다.

IV. 실시간 시스템 개발방법의 연구방향

4.1 실시간 시스템 개발특성

실시간 소프트웨어는 그 시스템에서 요구된 기능들을 주어진 시간제약(time constraint) 범위 안에서 모두 수행하되 시스템의 외부환경과의 상호작용이 올바르고 믿을 만하게 이루어지도록 제어하는 시스템을 말한다. 따라서, 외부세계와 상당히 밀접한 관계를 맺고있는 실시간 시스템은 하드웨어 장치들과 밀접하게 관련하며, 하드웨어 장치들의 수행능력을 최적화시키고 정해진 요구기능들을 만족시켜야 하는 등 실시간 시스템을 설계하고 구현하는 문제는 기존의 순차적 시스템과 비교하여 상대적으로 어렵고 복잡한다.

실시간 소프트웨어는 주어진 수행제약(performance constraints)하에 운영되기 때문에, 소프트웨어 설계는 소프트웨어 구조, 하드웨어 구조, 응용 요구사항, 운영체제의 특징, 구현언어에 따라 달리 적용이 되어야 한다. 그리고, 외부사건에 응답하는 어떤 행위를 생성해야 되므로, 고속의 자료 획득과 엄격한 시간범위와 신뢰성 제약 하에서 제어가 필요하다[3, 11].

또한, 실시간 소프트웨어는 복잡하고 대규모의 시스템일 경우가 많으므로 이 소프트웨어의 유지보수는 상당히 어려워질 것이며, 이 소프트웨어의 개발비용은 상대적으로 많이 들 것이다. 그러므로, 유지보수 문제는 실시간 시스템이 필수적으로 다루어야할 중요한 사항이 된다. 그리고, 일반 소프트웨어들에 비해 신뢰성은 실시간 시스템의 중요한 특징이다. 그러므로, 실시간 시스템의 개발에 있어 소프트웨어의 재사용은 중요하다. 소프트웨어를 재사용하게 됨으로써 직접 개발할 때보다 품질이 인정된 소프트웨어를 이용하게 되므로 신뢰성은 높아질 수 있고, 소프트웨어의 개발 노력을 감소시킴으로 소프트웨어의 개발 비용이 줄게되는 잇점을 갖게될 것이다.

〈표 1〉 실시간 시스템 개발 고려사항

- 병행처리 프로세스
- 병행처리 프로세스 간의 통신과 동기화
- 프로세스 스케줄링
- 우선순위 제어
- 프로세스 타이밍
- 프로세스 시간예측 및 시간제약
- 신뢰성 측면의 오류검출, 오류회복, 재구성(reconfiguration), 예외사항 처리
- 시스템의 안전성
- 소프트웨어 재사용, 소프트웨어 유지보수
- 등등

이와같은 실시간 소프트웨어를 개발함에 있어서 생각해 보아야 할 사항들을 살펴보면 (표 1)과 같이 요약할 수 있다.

4.2 자료흐름중심과 실시간 시스템 개발방법

현재, 자료흐름중심 개발방법은 실시간 시스템 개발영역에 많이 이용되고 있다. 자료흐름중심 개발방법은 1970년대 이후부터 개발된 방법으로 주로 응용시스템(business-system) 개발에 이용되었던 방법이다. 자료흐름중심 개발방법에서는 자료흐름도(DFD)를 이용하여 시스템을 분석한 후, 구조도를 이용하여 자료흐름도를 구조적 분석모델로 변환한다. 그러나, 이러한 방법에는 실시간 특성(real-time characteristc)(4.1절)을 가진 시스템을 취급하는 방법이 부족하여 실시간 시스템 개발을 위한 몇 가지 확장이 필요했다. 2장에서 살펴본 바와같이 Ward & Mellor 방법에서는 기존의 자료흐름도에 제어흐름, 제어프로세스, 자료의 연속성 표현, 상태흐름의 표현 등을 추가하였고, Gomaa의 방법에서는 자료흐름도가 비동기적인 프로세스들을 표시하지 못한점을 해결하기 위해 병행 프로세스 식별 휴리스틱과 함께, 이 프로세스들 간의 통신 및 동기화 표현방법을 제시하였다. 그리고, Nielsen & Shumate 방법에서는 기존 방법들(Mascot, Dart, Structured Design, Layerd Virtual Machine)의 장점을 적극 수용하여 실시간 시스템 개발을 위한 자료흐름중심 개발방법을 거의 완성하게 되었다.

이 방법들은 분석동안에는 자료흐름도, 자료사전, 상태도, 프로세스 명세, 개체관계도를 개발하고 있으며, 설계동안에는 다수의 프로세서를 가진 실시간 시스템을 개발하기 위해 시스템을 서브시스템과 태스크로 기능분해를 하고 있다. 서브 시스템은 하나의 이상의 태스크로 구성되어 있으며, 태스크들은 한의 프로세서(processor)에서 할당된 작업을 수행한다. 시스템을 서브 시스템으로 분할하는데는 서브 시스템들의 기능적 응집성(functional cohesion)을 높이면서, 서브 시스템들 간의 통신 오버헤드(communication overhead)를 줄이기 위해 가능하면 약결합(weak coupling)을 가지도록 하고 있다. 또한, 서브 시스템의

기능적 응집성과 더불어 시간적(temporal), 순서적(sequential) 응집성도 고려한다. 서브 시스템 혹은 태스크들 사이에는 어떤 기능을 수행하기 위해 통신을 하게 되는데 이를 위하여 통신 동기화(communication synchronization) 모듈을 마련하고, 이것을 이용하여 서브 시스템들 혹은 태스크들 간의 통신결합(약결합, 강결합)을 표현하고 있다. 태스크들은 자료흐름도의 기능들로부터 추출이 되는데, 자료흐름중심의 실시간 개발방법에서는 이것들을 식별하는 휴리스틱들을 마련하고 있다.

이처럼 자료흐름중심 개발방법에서는 캡슐화(encapsulation)의 단위로 자료흐름에 따른 기능 혹은 프로세스로 보고 있으며, 이러한 기능들을 모듈로 묶어 전체 프로세스 중의 일부분의 기능을 담당하도록 한다. 즉, 시스템 전체로 관심있는 자료들, 지식(knowledge)들을 분산시킨다. 이것이 소프트웨어의 환경변화에 따른 수정(modification)을 어렵게하는 요인이 되고 있다. 소프트웨어의 모듈들이 실세계 "객체"로 직접 사상이 되지 않았기 때문에 문제의 변경이 어떤 모듈에 가해지면 많은 모듈에서 변경의 파급효과를 초래하게 되고, 관련된 다수의 모듈들에 변경을 해야만 된다. 이것이 크고, 복잡한 실시간 시스템의 유지보수와 재사용 측면에서 어려움을 증가시키는 요인이 되고 있다. 따라서, 자료흐름중심의 개발방법의 이러한 문제들은 계속적인 연구를 통해 해결되어야 할 필수과제가 되고 있다.

4.3 객체지향과 실시간 시스템 개발방법

현재까지의 객체지향 개발 방법론들에서는 객체지향 개념을 적용하여 시스템을 체계적으로 분석/설계하므로써 실시간 시스템과 같은 크고, 복잡한 시스템의 복잡성을 상당히 해결하고 있다. 객체지향 개념은 1980년대 이후의 소프트웨어 개발 프로젝트의 대규모화, 프로그래밍 언어의 고수준화, 유지보수 해야할 소프트웨어 양의 증가 등의 소프트웨어 위기를 극복하는데 중요한 역할을 하고 있다. 객체지향 개념이 소프트웨어 위기를 극복하는데 중요한 역할을 할 수 있었던 것은 첫째, 객체지향 개념이 실세계의 객체를 시스템으로 모델화하였기 때문에 분석가와 사용자 양측의 상호이해를 돕고, 둘째, 시스템의 병행행동을 모델화하고 구현하는 자연스러운 방법을 제공하고, 세째, 하나의 객체내로 자료와 오퍼레이션을 캡슐화 하므로서 시스템의 변경으로 인한 비용을 감소시키고, 네째, 개발자가 시스템을 빠르게 이해하도록 하므로서 국면이전(paradigm shift)이 발생하므로서 나타나는 설계과정상의 불연속성을 해소하는 등 다양한 장점들을 포함하고 있었기 때문이다[6].

〈표 2〉 객체지향 분석/설계 활동

객체지향분석	객체지향설계
	객체, 동적, 기능 모델의 결합
	서브시스템으로 분할
	서브시스템 통신및 관계성 식별
객체, 클래스 식별	객체 병행성 식별
객체 구조화	자료구조 설계
객체 메시지 정의	객체 성질 분류
객체 메시지 접속관계 정의	(actor, server, agent)
객체 자료접근 관계 정의	객체 메시지 분류
상태도 개발	(비동기, 동기)
자료흐름도 개발	객체 가시성 결정
	객체의 실행 타이밍 분석
	설계의 최적화
	설계 결정 검토

이러한 이점들을 활용해서 병행행동을 모델링하고 프로그래밍하는 환경구축을 위해 노력해 오고 있다[4, 5,6,12,16,17]. 이 방법들의 객체지향분석/설계 활동에서 수행되어야할 작업내용을 살펴보면 〈표 2〉와 같이 요약할 수 있다.

그러나, 현재까지의 객체지향 분석/설계 방법들에서는 설계방법보다는 분석방법에 치중하고 있으며, 분석방법 중에서도 객체식별과, 캡슐화, 분류화, 상속개념을 적용한 객체구조화에 주로 관심을 쏟고 있는 형편이다. 따라서, 현재의 객체지향 방법들을 잘 활용하게 되면, 소프트웨어의 복잡성 문제, 유지보수 문제, 재사용 문제들이 상당부분 해결되리라 예상된다.

그렇지만, 객체의 동적인 특성분석이 미약하기 때문에, 〈표 1〉과 같은 객체들의 메시지, 메소드의 병행성, 객체들 사이의 메시지 통신 및 동기화, 객체의 실행 타이밍 등의 실시간 시스템 개발에 있어 문제가 되는 사항들에 대한 객체지향 기법에 의한 해결 방법은 미흡하다. Bo Sanden은 Booch의 "Object Oriented Development"에서 실시간 시스템을 개발할 때 주요하게 고려되어야 할 태스크의 두 가지 병행행동 즉, 주기적 행동과 비주기적 행동을 다루는 구체적인 방법에 대해서 언급하지 않음을 지적하고 있다[13]. 이에 따라, Booch의 객체지향 설계 방법론[5]에서는 실시간 시스템의 시간적인 행위를 취급하도록 상태도와 객체의 타이밍도, 기타 객체의 동적/정적 성질, 객체들 사이의 메시지를 표시하는 등의 일반 소프트웨어 개발 뿐만아니라, 실시간 소프트웨어를 개발하

는데 유용한 도구(tool)와 기술(technique)들을 마련하였다. 그렇지만, 실시간 처리 문제들을 해결함에 있어 이 도구들을 적절히 이용하는 구체적인 방법(method)에 대해서는 소홀히 다루고 있다. 실시간 시스템의 개발에 있어서 해결해야할 주요문제인 오류발생시의 대응책, 병행 프로세스들의 통신 및 동기화 문제, 우선처리 문제를 해결할 수 있도록 하는 적절한 방법이 이 방법들에는 취급하지 않는다. Booch의 객체지향 설계 방법론에서처럼 현재의 다른 객체지향 방법들[6,12,16,17]에서도 마찬가지로 실세계 객체의 캡슐화, 분류화, 상속 등의 개념을 적절히 적용하여 시스템을 체계적으로 분석/설계하는 방법에 대해서는 논하고 있지만 앞서와 같은 실시간 처리 문제에 대한 적절한 방법, 도구, 기술을 제공 못하고 있다.

그러므로, 현재의 객체지향 방법론의 재사용, 유지보수, 복잡성 측면의 문제해결 장점을 살리면서 실시간 시스템 개발 상의 문제점을 해결하기 위한 객체의 동적인 특성을 현재의 것보다 더 강조하고, 객체지향설계 방법을 보완할 필요가 있다.

4.4 객체지향적 실시간 시스템 개발방법

품질이 좋은 소프트웨어를 경제적으로 개발하기 위해서는 효과적인 개발방법론을 적용해야하고 이를 위해 여러가지 방법론들이 제안되었으며 또, 사용되고 있다. 현재 가장 널리 사용되고 있는 방법론으로는 구조적 분석/설계 기법이다[20]. 그러나, 새로운 움직임으로 객체중심 개발방법론이 앞으로는 구조적 방법론을 대체할 것으로 예상되고 있다. 이는 객체중심 방법론의 접근방법이 현실세계에서 사물을 보는 것과 흡사한 시작에서 시스템을 분석/설계하기 때문이며, 이 방법에 의하면 요구명세의 이해가 쉽고, 설계가 명확하며 유지보수가 용이하고 재사용성이 높다는 장점을 갖고 있기 때문이다.

그렇지만, 현재까지의 객체지향 분석/설계방법은 캡슐화, 추상화, 이해도 측면에서 기존의 구조적분석/설계 방법에 비해서 상당히 나아졌으나, 실시간 시스템과 같은 상황에서 필요로 하는 시간성(temporal characteritic) 문제에 대해서는 소홀히 다루고 있다. 현재의 객체지향 분석/설계 방법들에서는 객체의 통신, 객체 행위 메카니즘, 객체의 시간 예측성, 메소드의 시간 예측성, 메소드의 병행 수행성, 메시지의 시간 예측성, 메시지의 병행성 등에 관한 사항을 소홀히 취급하고 있다[9]. 그러므로, 객체지향 실시간 방법론에서는 시스템의 구조적인 면을 표현하는 정적인 관점과 더불어, 병행성(병행 태스크 식별, 병행 태스크들 간의 통신 및 통신화), 적응성(객체의 동적인 생성과 소멸), 실시간 실행(real-time execution)을 다루는 동적인 관점이 적절히 포함되도록 해야한다.

그리고, 앞절에서 살펴본 바와같이 근래까지의 실시간 소프트웨어 개발방법론들 모두 구조적 개발기법에 그 바탕을 두고 있고, 이 방법들 모두 요구사항의 정확한 분석보다는 실시간 시스템을 어떻게 설계할지에 주요관심을 갖고 있다. 다시 말하면, 자료흐름도로부터 프로세스들의 식별, 병행 프로세스들 간의 통신 및 동기화, 그리고, 프로세스 스케줄링 등의 방법에 대한 주요 해결방법들을 제시하고 있다. 그러나, 이 방법들의 주요문제점은 자료흐름도로부터 프로세스들을 식별하려는 시도에 있다. 이는 자료흐름도의 모든 연산들이 결국에는 어느 프로세스에 포함되도록 하였기 때문에, 결과적으로는 병행 프로세스의 수가 많아지고, 이로써 프로세스의 통신과 동기화에 수반되는 오버헤드가 커져 시스템 전체의 수행능력에 영향을 미치는 경우가 많다. 이러한 문제점을 Bo Sanden은 지적하고 있다[14,15].

그러므로, 객체지향 실시간 방법론에서는 분석단계에서는 객체 모델을 중심으로 동적모델, 기능모델을 개발하나 시간 특성들이 적절히 분석되도록 해야하고, 설계단계에서는 객체모델을 기반으로 동적모델과 기능모델을 병행 프로세스의 식별, 프로세스 간의 통신 및 동기화, 프로세스 스케줄링 등의 〈표 1〉과 같은 실시간 시스템 문제들을 해결하는데 적극 활용하도록 해야한다.

이러한 관점에서, 실시간 객체지향 모델이 갖추어야 할 요건은 비 실시간 객체지향 모델과는 두가지 관점에서 차이가 있다[2]. 첫째, 객체모델에서는 캡슐화, 분류화, 상속성은 물론 객체의 시간 특성을 표기하는 방법을 지원해 주어야 한다. 둘째, 객체, 클래스, 메소드, 그리고 메시지에 대해서만이 아니라, 이와 관련된 실행 매카니즘에 대한 예측 가능한 시간특성(predictable temporal characteristic)을 갖도록 해야한다.

특히, 실시간 객체지향 설계 모델에는 실시간 시스템의 여러 문제점들을 분석하고 해결하는 표현방법을 갖추고 있어야 한다. 예를들면, 이 모델은 서브시스템으로 모듈화하는 방법, 객체의 본질적 병행 프로세스를 표시하는 방법, 객체 프로세스의 통신과 동기화 표시방법, 객체 프로세스의 우선순위 표시방

법, 프로세스들간의 메시지 우선순위 표시방법 등 여러가지 실시간 시스템의 문제를 해결하기 위한 모델이 되어야 한다.

객체는 다른 객체와의 관계에서 독립성을 지니고 있으므로 근본적으로 병행성(concurrency)를 지니고 있다. 따라서, 실시간 시스템에서 객체들간의 병행성은 자연스러운 것이며, 이러한 시스템의 분석과 설계에서 그 개념이 쉽게 적용이 될 수 있다. 그리고, 앞서 연구한데로 일반적으로 실시간 시스템의 분석과 설계에서도 프로세스의 개념을 도입하는 것과 프로세스 간의 통신과 동기화가 가장 중요한 문제로 꼽히고 있다. 시스템의 구현상 Ada를 위시한 기존의 고급언어를 이용하는 환경에서는 설계과정에서 프로세스를 활용하여 시스템을 구성하는 것이 더 효율적이다. 그러므로, 객체중심설계에서도 구현상의 효율을 위해 프로세스를 설계하도록 해야한다.

프로세스와 객체와의 관계를 살펴보면, 하나의 프로세스는 여러개의 객체간의 활동을 포함할 수도 있고 하나의 객체의 활동들이 여러 개의 프로세스들로 이루어질 수 있다. 그러나, 대다수의 경우에는 하나의 객체의 활동들로 하나의 프로세스가 형성됨이 일반적이다. 따라서, 객체들의 속성이나 연산 등의 객체 특성들을 적절히 검토함으로써 객체들의 연관이 프로세스간의 통신 성격을 띈다면 이들을 병행 프로세스로 식별할 수 있을 것이다. 이때, 두 객체간의 연관이 강결합(tight coupling) 형태이면, 두개의 프로세스로 식별할 것이 아니라 하나로 통합할 수도 있으며, 하나의 객체내에 다른 성격의 연관 등이 모여 있을 경우에는 복수개의 프로세스로 식별할 수 있다.

이와같이, 객체지향 실시간 방법론은 객체지향(object orientation) 개념을 중심으로하여 실시간 시스템의 여러 주요문제점들을 해결하는 분석/설계 방법이 될 것이다. 이때, 이용되는 모델이 〈표 3〉과 같은 실시간 객체지향 모델이다.

실시간 객체지향 분석모델은 객체, 동적, 기능모델로 구성되어 있다. 객체모델은 실세계에서의 개체(entity)들에 대응되는 객체(object)와 관계(relationship)의 견지에서 한 시스템의 정적인 구조에 대해 기술한 모델로서 시스템내의 객체들과 그들간의 관계에 대해 설명해준다. 동적모델은 사건과 상태의 관점에서 시스템의 제어구조에 대해 묘사해 놓은 모델로서 시스템내에서 객체들간의 상호작용(interaction)에 대해 나타내는 것이며, 시간에 따라 변화하는 시스템의 여러 측면에 대해서 설명한다. 기능모델은 자료값(data value)과 기능(function)의 시각에서 시스템의 계산처리구조에 대해 설명해 놓은 것으로서 한 시스템 내에서 자료값과 변환에 대해 다루고 있다.

〈표 3〉 실시간 객체지향 모델

분석	실시간 객체지향 분석모델 - 객체 모델 - 동적 모델 - 기능 모델
설계	실시간 객체지향 설계모델 - 객체 인터페이스 모델 - 프로세스 인터페이스 모델 - 객체 구조화 모델 - 프로세스 스케쥴링 모델 - 기타 (신뢰성, 안전성 점검모델)

한편, 실시간 소프트웨어를 개발하기 위해서는 분석보다는 설계에 어려움이 많아 기존의 실시간 소프트웨어 개발방법들이 설계에 중점을 둔것과 같이 실시간 객체지향 설계모델은 객체지향 개념을 적용한 실시간 소프트웨어의 주요모델이다. 이 모델에는 객체지향 설계 활동에 따라 객체 인터페이스 모델, 프로세스 인터페이스 모델, 객체 구조화 모델, 프로세스 스케쥴링 모델 등으로 구분될 수 있다. 객체 인터페이스 모델은 분석동안에 식별한 객체, 동적, 기능 모델을 하나의 모델로 통합(integration)하는 초기 모델로 객체들간의 통신 및 관계성을 표현한다. 객체들간의 통신은 동적모델과 기능모델로부터 각각 사건과 자료 흐름을 추출하여 객체모델에 표시한다. 프로세스 인터페이스 모델은 객체가 가지고 있는 본질적 병행성을 식별하여 객체들간의 메시지 우선순위, 메시지 실행 타이밍 등을 결정하기 위하여 이용하는 모델이다. 이때, 객체의 본질적 병행 프로세스는 객체간의 인터페이스 특성을 고려하여 병행 프로세스로 처리되어야 할 부분과 그렇지 않아도 될 부분으로 구분된다. 이와같이 처리하는 이유는 불필요한 병행 프로세스 취급에 의해 수반되는 프로세스 통신과 동기화 오버헤드 유발을 막기 위해서이다. 객체 구조화 모델은 객체별로 다른 객체에 보여줄 수 있는 부분과 캡슐화 시켜야 하는 부분을 결정하여 표시한 객체 가시성에 대한 모델이다. 프로세스 스케쥴링 모델은 객체 구조화 모델에 따라 실시간 소프트웨어의 실행 구조가 확립됨에 따라 프로세스들이 마감시간(deadline)을 충족시키는 범위내에서 각 프로세스들에게 처리기(processor)를 할당하고 프로세스 실행 우선순위를 부여하기 위한 수단이 되는 모델이다.

이러한 실시간 객체지향 설계 모델들은 객체지향 개념을 적용하여 실시간 시스템을 개발하기 위해 실시간 실행 특성들을 모두 해결하기 위한 방안을 제시하는 모델이다. 따라서, 객체지향적 실시간 시스템 개발방법에서는 실시간 소프트웨어를 개발하면서 발생하는 여러 문제점들을 해결함과 동시에 객체지향 개념의 이점을 살리는 방법이어야 하며, 이때 분석/설계의 복잡성을 감소시켜 신뢰성과 안전성이 있는 시스템을 개발하도록 해야한다.

V. 결 론

본 고에서는 자료흐름 중심의 실시간 소프트웨어 개발 방법론들과 객체지향적 실시간 시스템 개발 방법들에서 적용되는 기본생각, 표기법, 절차에 대하여 고찰해 보았으며, 실시간 소프트웨어 개발을 위한 객체지향 방법의 보완점에 대하여 살펴보았다.

기존의 실시간 시스템 개발방법들이 구조적 분석/설계 기법에 바탕을 두고 있다는 점과 중요한 실시간 시스템 특성들을 모두 방법론에 소화시키지 못하고 있다는 점은 새로운 방법론의 필요성을 증대시켜 왔다.

현재 잘 정립되어 있는 객체지향 방법들은 크고, 복잡한 시스템을 객체지향 개념을 적용해서 체계적으로 모델링하므로서 복잡성이 많이 경감된 모델을 형성한다. 그러나, 기존의 구조적 분석/설계 기법과 같이 실시간 시스템을 개발하기 위해서는 방법론상의 보완이 더 필요하다.

새로운 객체지향 실시간 시스템 개발방법에서는 분석 동안에는 객체모델을 중심으로 한 동적, 기능 모델을 개발하고, 설계 단계에서는 객체모델을 기반으로 동적모델과 기능모델이 통합되어 실시간 시스템 개발을 위한 객체의 병행프로세스 식별, 객체간의 통신 및 동기화 처리, 객체 프로세스들의 스케쥴링 등을 수행하도록 한다.

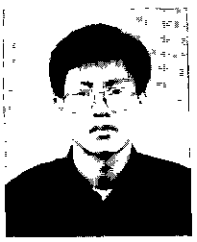
객체지향 개념을 바탕으로 실시간 소프트웨어 개발에 있어 해겨해야할 여러가지 무제들을 분석과 설계단계로 적절히 구분하여 문제를 분류하고, 실시간 소프트웨어를 개발하기 위한 특수목적용 모델들을 적극활용한다. 이 모델들을 실시간 객체지향 모델들이라고 하는데, 실시간 객체지향 분석모델들(객체, 동적, 기능 모델)은 기존의 비실시간 객체지향 분석 모델들과 유사하나 실시간 특성을 표현하는 동적모델에 시간특성이 더 강조되어 표시되는 것이 다른점이고, 실시간 객체지향 설계모델들은 실시간 소프트웨어의 병행행위 특성들을 적극 수용하여 실시간 실행특성들을 설계할 수 있도록한 모델들이다. 실시간 객체지향 설계모델들은 객체 인터페이스 모델, 프로세스 인터페이스 모델, 객체 구조화 모델, 프로세스 스케쥴링 모델, 기타 신뢰성 점검모델, 안전성 점검모델들로 구성될 것이다.

앞으로, 객체지향 실시간 시스템 개발방법론은 실시간 시스템을 개발함에 있어 발생하는 여러가지 문제들을 활용할 수 있도록 적절히 체계화된 방법을 갖게될 것이며, 이 방법들 각각에서는 객체 지향 실시간 모델들을 활용해서 문제들을 하나하나 해결할 것이다. 더 나아가서는 이들 방법들을 CASE Tool화하여 실시간 시스템을 개발함에 있어, 보다 적은 비용으로 커다란 생산성을 가져올 수 있도록 해야 하겠다.

참 고 문 헌

1. Birrel, N. D. and Ould, M. A., A Practical Handbook for Software Development, Cambridge University Press, 1985.
2. Thomas E. Bihari, and Prabha Gopinath, "Object-Oriented Real-Tıme Systems: Concepts and Examples," IEEE Computer Society, Dec., 1992, pp. 25~32.
3. Alan Burns, and Andv Wellıngs, Real-Time Systems and Their Programming languages, Addison Wesley, 1990.
4. Grady Booch, "Object Oriented Development," IEEE Trans. Software Eng., Vol. SE-12, No. 2, Feb. 1986, pp. 211~221.
5. Grady Booch, Object Oriented Design with Applications, Benjamin Cummings, 1991.
6. P. Coad, and E. Yourdon, Object Oriented Analysis, Yourdon Press, 1990.
7. Gomaa, H., "Software Design Method for Real-Time System," CACM, Vol. 27, No. 9, Sept. 1984.
8. Gomaa, H., "Structuring Criteria for Real-Time System Design," Proceeding 11th International Conference on Software Engineering, 1989, pp. 290~301.
9. Davie E. Monarchi, and Gretchen I. Puhr, "A Research Typology for Object-Oriented Analysis and Design," CACM, Vol. 35, No. 9, Sep. 1992, pp. 35~47.

10. Kjell Nielsen, Ken Shumate, Designing Large Real-Time Systems with Ada, McGraw-Hill, 1988.
11. Roger S. Pressman, Software Engineering: A practitioner's Approach, Third Ed., McGraw-Hill, Inc., 1992.
12. James Rumbaugh, Michael Blaha, William Premeriani, Fredrick Eddy, William Lorensen, Object Oriented Modeling and Design, Prentice-Hall, 1991.
13. Bo Sanden, "The Case for Electric Design of Real-Time Software," IEEE Trans. Software Eng., Vol. 15, No. 3, Mar. 1989.
14. Bo Sanden, "An Entity-Life Modeling Approach to the Design of Concurent Software," CACM, Vol. 32, Mar. 1989, pp. 330~346.
15. Bo Sanden, "Entity-Life Modeling and Structured Analysis in Real-Time Software Desing-A Comparison," CACM, Vol. 32, No. 12, Dec. 1989, pp. 1485~1466.
16. S. Shlaer and S. J. Mellor, Object-Oriented Systems Analysis: Modeling the World in Data, Prentice Hall, 1989.
17. S. Shlaer and S. J. Mellor, Object Lifecycles: Modeling the World in States, Yourdon Press, 1992.
18. Ward, P. T. and Mellor, S. J. Structured Development for Real Time Systems I, II, III. Yourdon Press, 1985.
19. Ward, P. T., "The Transformation Schema: An Extention of the Data Flow Diagram to Represent Control and Timing," IEEE Trnas. Software Eng., Vol SE-12, No. 2, Feb. 1986, pp. 198~210.
20. E. Yourdon, Modern Structured Analysis, Prentice-Hall, 1989.

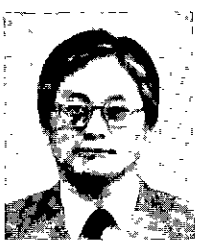


이 광 용

1991 숭실대학교 전자계산학과(학사)
1993 숭실대학교 대학원 전자계산학과(석사)
1993 ~현재 숭실대학교 대학원 전자계산학과 박사과정
관심 분야 : 소프트웨이 공학, 실시간 시스템, 인공지능, 분산처리

류 성 열

1976 숭실대학교 전자계산학과(학사)
1980 연세대학교 산업대학원 전자계산학과(석사)
1988 아주대학교 대학원 박사과정 수료
1981 ~1982 전국은행협회 어음교환기술 고문
1987 ~1988 숭실전자계산원 부원장
1985 ~숭실대학교 중앙전자계산소 소장
현재 숭실대학교 전자계산학과 교수
관심 분야 : 소프트웨어공학, 자료구조



정 기 원

1967 서울대학교 전기공학과 졸업
1981 미국 일라비마주립대학(헌츠빌) 전산학 석사
1983 미국 텍사스 주립대학(알링턴) 전산학 박사
1966 ~1968 미팔군 전자기사
1969 ~1971 대한전자공업주(자료처리 과장)
1971 ~1975 한국과학기술연구소 전자계산실
1975 ~1990 국방과학연구소 책임연구원
1990 ~현재 숭실대학교 전자계산학과 부교수
관심 분야 : 소프트웨어 공학, 분산처리, 모델링/시뮬레이션, 실시간 시스템, 인공지능