J. Astron. Space Sci. 15(2), 475–484 (1998)

위성관제 시스템 개발을 위한 구조적 기법을 재사용한 객체 지향 모델링에 관한 연구

김재훈[1] · 정원찬[1] · 이상범[2]

[1] 한국전자통신연구원 무선방송기술연구소 위성통신시스템연구부

[2] 단국대학교 자연대학 전자계산학과

A Study on an Object-oriented Modeling for the Satellite Control System Development reusing Structured Analysis and Design Approach

Jaehoon Kim[1], Wonchan Jung[1], and Sangbum Lee[2]

[1] Satellite Communications System Department, Radio & Broadcastiong Technology Laboratory, ETRI

[2] Department of Computer Science, Dankook University

email: jhkim@etri.re.kr

(Received October 14, 1998; Accepted November 3, 1998)

요 약

객체지향 소프트웨어 개발은 그 우수성이 입증됨에도 불구하고 기존 개발 방법론에 익숙해진 개발자들에게는 여전히 사용하기엔 어려움이 있다. 본 논문에서는 차세대 위성 관제시스템 개발을 위해 기존의 구조적 분석 기법인 자료흐름도(DFD: Data Flow Diagram)와 그래픽 설계 명세서인 구조도를 재사용하여 객체지향 설계 모델링할 수 있는 방법에 대하여 연구하였다. 즉, 객체지향 기법에서 가장 어렵게 느껴지는 객체 추출을 기존의 구조적 그래픽 기법을 이용함으로써 분석 설계를 손쉽게 할 수 있는 방법을 제시하였다.

ABSTRACT

The object-oriented approach is a difficult method for engineers who are accustomed to other software development methods although it is an excellent software development approach. This paper presents a method for object-orineted modeling re-using DFD(Data Flow Diagram) and SC(Structure Chart) of structured analysis and design approach. This paper suggests an easy method for analysis and design using structured approach for object abstraction, which is one of the most difficult things in object-oriented approach.

1. 서 론

다양한 기법(technique) 또는 방법(method)들이 소프트웨어 개발 단계의 문제 해결에 도움을 주고자하는 시도에서 개발되어 왔으며, 일련의 기법과 방법들을 조합하여 분석에서 유지 보수에 이르는 전 과정에 걸쳐 일관성 있는 원리(principle)에 의해 개발 방향을 제시하는 것을 개발 방법론이라 한다. 현재 사용되는 것들 중 크게 뚜렷이 구별되는 두 가지가 있다: 구조적 방법론(structured approach)은 개발하고자 하는 소프트웨어 시스템을 함수 위주로 구성하는 반면에, 객체지향 방법론(object-oriented approach)에서는 객체와 이들간의 관계로 시스템을 구성하고 있다. 객체지향 방법론이 여러 면에서 우수하지만, 현재 실제 개발에서 많이 사용되지 않는 이유로는 기존 방법론에 익숙해져 있는 개발자에게 새로운 방법론을 이해할 수 있도록 도와주는 효율적인 교육, 기법, 그리고 도구의 개발이 만족스럽지 못하기 때문이다. 그 중에서 가장 큰 원인은 객체 지향 기법에서 큰 난관으로 남아 있는 분석 또는 설계 단계에서 적당한 객체와 행위를 찾아내는데 어려움이 있기 때문이다. 이러한 어려움이 있음에도 불구하고, 객체 지향 개발 방법론은 소프트웨어의 비중이 커지고 요구 사항이 복잡 다양함에 따라 대두된 소프트웨어 위기(software crisis) 해결에 도움을 줄 수 있는 좋은 소프트웨어 개발 방법론으로 인식되어져 가고 있으며, 사용면에서 증가 추세에 있다.

한편 소프트웨어 공학에서는 오래되거나 좋은 구조를 갖지 못한 소프트웨어를 폐기하고 다시 개발하기보다는 이를 재사용할 수 있도록 하는 재공학(re-engineering)에 대한 관심이 높아지고 있다. 본 논문에서는 구조적 방법론에 의해 개발된 분석 및 설계 명세서를 이용하여 객체 지향 소프트웨어로 변환할 수 있는 방법을 제시함으로써 소프트웨어의 재사용 기법을 소개하고 객체지향에 익숙하지 못한 기존 개발자에게 도움을 주고자 한다. 제 2장에서는 소프트웨어 재공학에 대한 관련 연구를 소개하고, 제 3장에서는 변환 절차를 기술하였으며 결론은 제 4장에 있다.

2.소프트웨어 제공학에 대한 관련 연구

재공학에서 가장 많은 관심을 끌고 있는 분야가 구조적기법으로 개발된 소프트웨어를 객체지향 소프트웨어로 전환시키는 것이다. 아직도 많은 개발자들에게는 객체 지향은 친숙하지 못한 소프트웨어 개발 방법론으로 인식되어져 있다. 이러한 어려움을 극복하고 도움을 주고자 여러 가지 시도가 이루어지고 있는데, 그 중에서 지금까지 많이 사용해 친숙해진 함수 지향적 방법을 이용하고자 하는 연구가 많이 이루어져 있다. 이러한 접근 방법을 분류하면 다음과 같다.

2.1 자료흐름도(DFD)만을 이용하는 기법

여러 다른 레벨의 DFD를 사용하여 경험적인 룰(heuristic rule)을 적용시켜 객체와 관련된 행위를 추출하는 방법을 말한다(Ladden 1988, Rumbaugh *et al* 1991, Lee & Carver 1991, Booch 1987). DFD에서 직접적으로 객체와 행위를 추출할 수 있는 rule이 없으므로 제약과 가정이 수반되지 않을 수가 없다. (Lee & Carver 1991)에서는 여러 Physical DFD을 이용한 객체 지향 모델을 만드는 과정

을 설명하고 있으며, 일반적으로 DFD의 source, sink 그리고 data store가 직접적으로 객체로 전환될 수 있으며, 각 process는 객체와 행위를 동시에 갖고 있다. 그리고 data flow는 객체 지향 모델에서 객체 또는 객체의 속성(attribute)으로 전환된다. 표 1은 DFD들의 요소와 객체모델간의 일반적인 관계를 정리한 것이다. 하지 만 DFD의 각 요소를 직접적으로 객체 지향 모델의 각 요소로 전환시키려는 시도에는 무리 가 있는데, 그 이유는 서로의 관점이 다르기 때문이다.

표 1. DFD 요소와 객체지향 모델 요소와의 관계.

DFD 요소들	객체 모델 요소
data store	object
data flow	object 또는 data
process	operation
source/sink	external object

2.2 DFD와 ERD의 동시 사용법

이 방법에서는 ER(entity-relationship)다이어그램을 이용하여 객체와 객체간의 관계 그리고 속성을 추출하고, DFD을 사용하여 행위와 데이터 그리고 컨트롤을 추출한다. Balin은 ERD와 DFD를 사용하여 EDFD (entity data flow diagram)이라는 객체 지향 분석 모델링 방법을 개발했는데, 이는 DFD와 유사한 형태로 각 노드가 객체 또는 함수로 구성되어져 있다(Bailin 1989). Ward 또한 ERD와 DFD를 이용하여 실시간 객체 지향 모델링에 관한 연구를 발표한 적이 있다(Ward 1989).

2.3 구조적 프로그램의 소스 코드를 직접 이용하는 방법

비객체 지향 방법에 의해 개발된 프로그램의 소스 코드를 사용하여 객체 지향 프로그램 으로 전환하는 연구가 많은 관심을 끌고 있지만, 전반적으로 획기적인 방법이 제시된 것은 없다. 몇 문헌에서 비객체 지향 프로그램의 소스 또는 요구 사항 명세서 그리고 문서에서 객체 지향 소프트웨어로 전환시키는 방법이 소개되어져 있다(Breuser & Lano 1991, Dietrich *et al* 1991, Jacobson 1991, Dunn & Knight 1991, Lee *et al* 1993). 이 때 매개변수나 전역 변 수들 중심으로 이에 관계된 함수들을 모아서 어떤 룰을 적용시켜 객체로 구분한다. 하지만 대부분 방법이 애매 모호하고, 추상적이며 객체 추출에 초점이 맞추어져 있을 뿐만 아니라 뚜렷한 절차 방식을 제안하지 못하고 있다.

3. 위성관제 시스템

인공위성이 통신매체로 충분한 기능을 발휘하기 위해서는 이를 제어하고 관리하는 관제 시스템의 역할이 중요시된다. 관제시스템은 위성통신시스템으로 특성적으로는 분산환경에서 실행되는 실시간 시스템으로 여러 개의 서브 시스템으로 구성되어져 있는데, 차세대 관제시스템은 저궤도 및 정지궤도 위성 등 여러 종류의 위성을 동시에 효과적으로 제어할 수 있는 개방형 시스템으로 발전하기 위해서는 소프트웨어 모듈의 재활용 뿐만 아니라 소프트웨어 구조, 즉 frame work이나 설계 명세서의

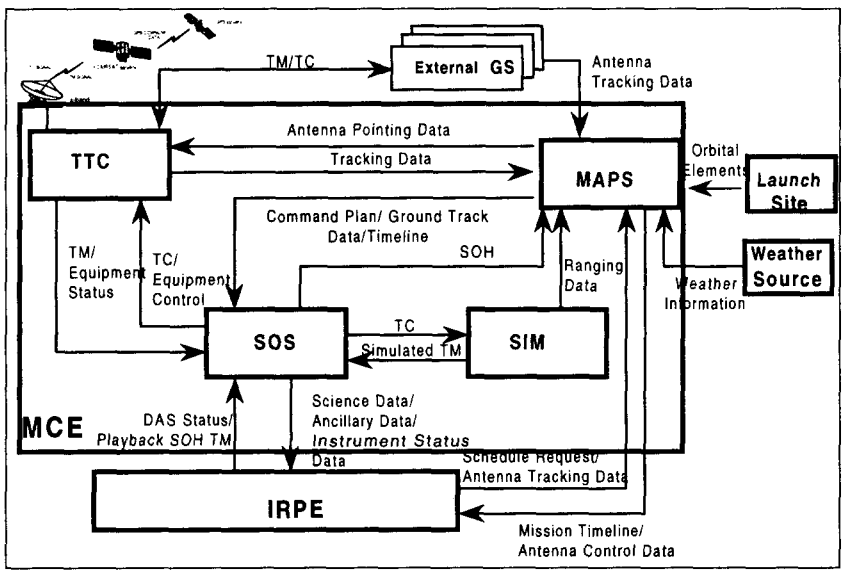

그림 1. 다목적 실용위성 관제시스템의 구성도

재사용이 필수적이다. 이러한 요구사항을 만족하기 위해서는 객체지향적 방법이 가장 적합하다. 그림 1은 현재 한국전자통신연구원에서 개발하고 있는 다목적 실용위성 관제시스템의 구성도이다. 다목적 실용위성 관제시스템은 구조적 분석 및 설계기법을 이용하여 개발하였다. 분석을 위해서는 자료흐름도를, 설계를 위해서는 구조도(structure chart)와 프로세스의 처리 절차를 기술하는 P-Spec을 이용하여 알고리즘을 정의하였다. 이러한 명세서는 객체 모델에 비해 여러 가지 단점이 많고 특히 재사용과 유지보수에 절대적인 취약점을 갖고 있다. 따라서 본 연구에서는 이러한 기존의 분석.설계 명세서를 재사용하여 차세대 관제시스템을 위한 객체 모델링 방안을 제시하고 있다.

4.구조적 명세서에서의 객체 모델로의 전환 기법

본 논문에서 제시하는 전환 기법은 다음과 같은 가정에 그 기초를 두고 있다. 즉, 구조적 기법에 의한 분석 및 설계 명세서가 함수 중심으로 구성되어져 있더라도 이러한 함수들은 객체지향 소프트웨어에 역시 필요한 함수라는 점이다. 물론 이들이 객체지향에 필요한 모든 정보를 포함하고 있지는 않다. (Jacobson 1991)에서 지적한 것과 같이 구조적 프로그램에서는 각 객체의 행위와 데이터가 관련된 객체 단위로 묶어져 있기보다는 기능별로 흩어져 있다. 이러한 비객체 지향적 방법에 의해 작성 된 명세서에서 객체 다이어그램으로 전환시키는 데에 있어서 가장 어려운 일은 흩어져 있는 함수들 을 어떤 규칙이나 방법에 의해 객체 단위로 묶어서 재구성시키는 가에 있다. 구조적 방법론의 모

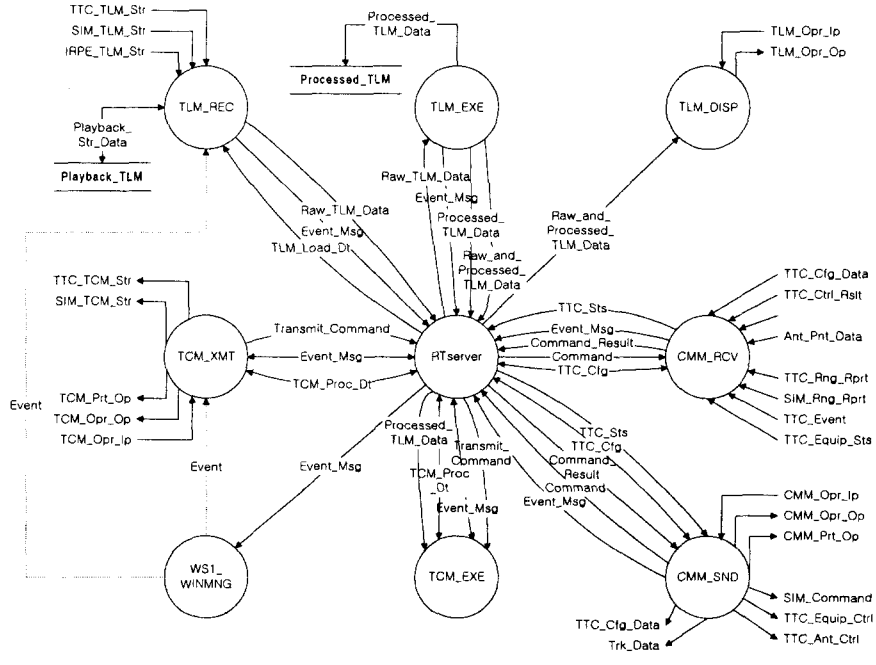

그림 2. 위성운용시스템의 레벨 1DFD

델들 중에서 객체 모델 구성에 사용되어지는 것은 그래픽 분석 명세 서인 상위 레벨의 physical DFD, 그래픽 설계 명세서인 구조도와 P-Spec 이다. 구조도는 DFD를 이용하여 함수들의 관계를 계층적으로 도식화한 것으로 DFD가 여러 레벨의 다이어 그램으로 구성된 것과는 달리 하나의 그래프로 전체 시스템의 구조를 표현해주는데, 함수간 의 호출관계 및 매개변수의 전달 등 자세한 정보를 포함하고 있다. DFD와 구조도를 이용한 객체모델 추출 절차는 아래와 같이 요약될 수가 있다.

4.1 Active 객체에 대한 추출

일반적으로 physical DFD의 상위 레벨의 프로세스는 함수적인 것이기보다는 행위를 수행하는 주체(actor) 또는 행해지는 장소(place), 즉, 객체로 표현되기가 쉽다 (Lee & Carver 1991). 그림 2는 관제시스템의 서브 시스템인 위성운용시스템(Satellite Operational Systems)에 대한 레벨-1의 DFD로 여러 개의 프로세스와 데이터 흐름(data flow) 및 데이 터 저장(data store)으로 구성되어져 있다. 이때 각 프로세스는 객체로 간주할 수가 있는데 이들은 특성상 active 객체라 할 수 있다. Active 객체란 논리적인 기계같이 어떤 일을 능동적으로 수행하는 단위가 된다. 한편 여기에 나타난 프로세스를 자세히 살펴보면 시스템에 수행되는 함수 단위라기 보다는 어떤 함수를 수행하는 주체로 파악할 수 있다. 일반적으로 대형 시스템을 DFD로 분석 할 때, 상위 레벨의 DFD의 프로세스는 함수이기보다는

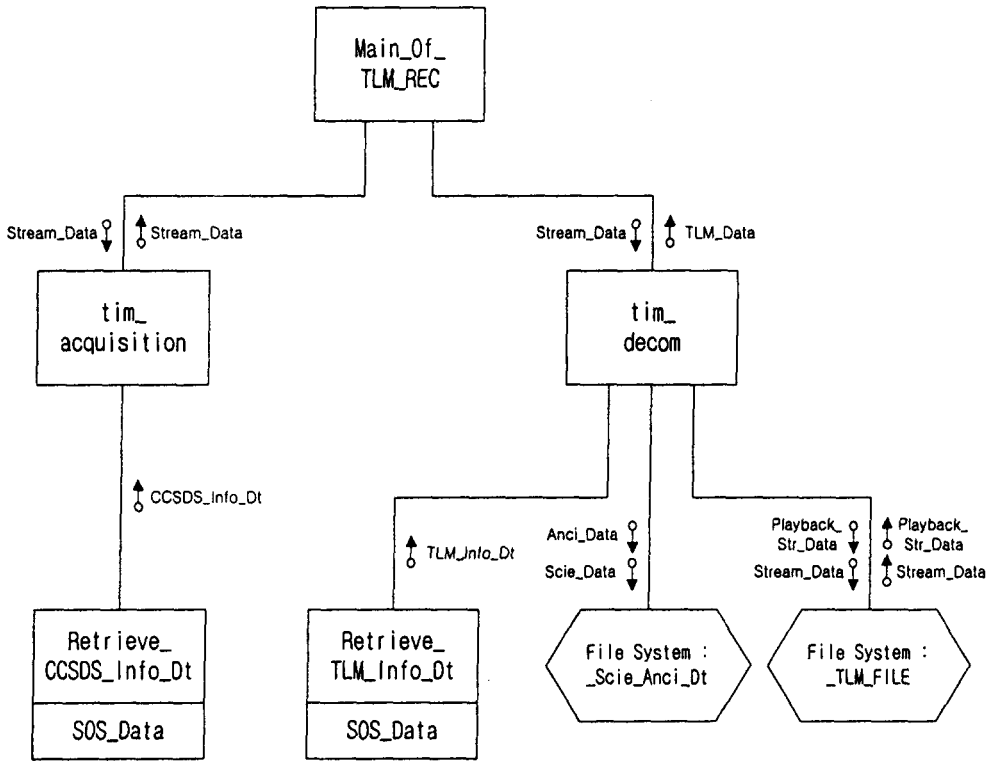

그림 3. TLM_REC 프로세스에 대한 구조도

객체로 정의되는 경향이 있다. 극단적인 예로는 최상위 레벨이 context 다이어그램에 나타난 하나의 프로세스는 개발하고자 하는 전체 시스템을 나타내는 하나의 avtive 객체로 정의할 수 있다. 이러한 관점에서 보더라도 객체지향 소프트웨어 개발 방식이 구조적 개발 방식에 비해 인간의 사고 논리방식에 적합하다고 할 수가 있다.

4.2 Passive 객체에 대한 추출

그림 3은 TLM_REC라는 프로세스에 대한 구조도로 여러 개의 서브 모듈 즉 함수와 이들간의 관계 그리고 매개변수를 포함하고 있다. 여기에서 우선 main 모듈인 "TLM_REC"는 위 과정에 정의된 능동 객체가 된다. 소스 코드를 기준으로 객체 모델을 찾아낼 경우 매개 변수 또는 전역변수가 객체에 사용되는 데이터로 간주되기 때문에 여기서는 매개변수인 Stream_data가 중심이 되고 이를 처리하는 tlm_acquisition()과 tlm_decomposition()이라는 함수(행위)들의 모임인 Stream_data_Processor라는 passive 타입의 객체를 추출할 수가 있 다. 만약 매개변수가 여러 개 있다면 4-4과정에서 언급된 알고리즘을 이용하여 객체를 정의 할 수가 있다. 이 때 passive 객체란 자료의 처리를 담당하는 것으로

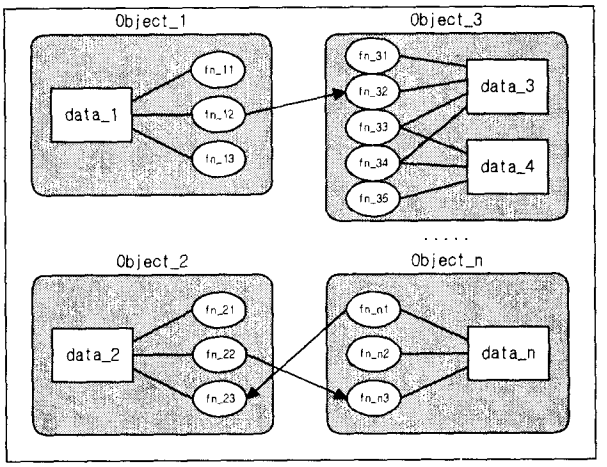

그림 4. 관계 그래프

예를 들면 스택(stack) 이나 큐(queue) 같은 것이 된다. 구조도에 상세한 P-Spec이 첨부된 경우는 객체 추출에 더 많은 정보를 얻을 수가 있게 된다.

4.3 관계 그래프의 설정

함수와 데이터의 관계를 그래프 형태로 표현하면, 함수 지향 프로그램에서는 함수가 중심이 되고 주위에 관련된 데이터가 나타나는 반면, 객체 지향 프로그램에서는 데이터 중심에 관계된 함수들이 나타나기 마련이다. 함수 중심의 DFD와 구조도에서 데이터 중심의 형식으로 전환시키기 위해서 관계그래프(relation graph)를 이용하여 임시적으로 표현할 수가 있다. 관계그래프를 만들기 위해서는 4-2에서와 같이 매개변수와 전역변수를 추출하여 중복된 것은 제거시킨 후 이들을 중심 노드로 지정하고, 각 변수 또는 파라메트에 관련된 함수를 주변 노드로 지정한다. 한 set의 관계 그래프가 설정되면 호출자와 비호출자 간의 호출 관계도 관계 그래프간의 호출 관계를 설정하게 되는데, 그 형태는 그림 4와 같다.

4.4 해결 객체 (Solution object)의 파악

이 과정에서의 주요 관점은 객체들의 설정이다. 소프트웨어 관점에서 객체는 단지 한 모 듈로서 데이터들와 그 데이터를 처리하는 행위(operation)로 구현되어져 있다. 경우에 따라 선 정확한 객체 이름을 추출하거나 의미 있는 이름을 부여하기가 불가능하기 때문에 임의의 의미를 갖지 않는 이름을 부여하게 된다. 여기서 주지할 것은 객체의 이름이 분석이나 설계 단계에서는 중요할 지는 모르나 (객체 이름이 그 속성과 행위의 특성을 어느 정도 의미하고 있기 때문에), 구현된 프로그램 상에서는

```
Collect all parameters and global variables in the Structure Chart
        and define them as data
Collect all functions from Sturcture Chart
Sort the data in sequence order
Eliminate the duplicaticated data
Construct a set of relation graphs using data and functions
FOR each data DO
    get the set of related functions
IF (set of functions of DATA_A ⊇ set of functions of DATA_B) OR
  (set of functions of DATA_A ⊆ set of functions of DATA_B)
THEN
   DATA-A and DATA_B belong to the same object A
ELSEIF (any member function of DATA_B ∋ set of functions of DATA_A) OR
         (any member function of DATA_A ∋ set of functions of DATA_B)
THEN
      determine DATA_A and DATA_B belong to the same object B or not
 ELSE
      DATA_A belongs to Object A
      DATA_B belongs to Object B (A\=B)
```

그림 5. solution 객체 추출 알고리즘

객체 이름 그 자체는 중요하지 않다. 그러므로 여 기서 의미 있는 객체 이름을 정의하는 것보다는 데이터와 행위의 묶음으로서의 객체 구성 (clustering)에 중점을 두고 있다. 이름 자체가 갖는 의미는 프로그램 상에서 크지 않고 단지 모듈 이름으로만 사용되기 때문이다. 여기서 두 타입의 객체가 존재하게 되는데, 첫 번째가 DFD에서 추출된 능동 객체이며, 두 번째는 구조도의 매개변수와 전역변수를 이용한 데이터 와 관련 행위를 대표하는 관계 그래프에서 추출된 무명의 객체들이다. 그림 5는 위 과정에 서 정의한 candidate 객체들을 이용하여 solution 객체를 정의하는 알고리즘이다.

4.5 객체 모델에서의 메시지 루틴 설정

관계 그래프에서 다른 함수간에 설정된 화살표는 두 함수들간에 호출자(caller)와 피호출자(callee)관계에 있음을 나타내고 있다. 이러한 정보가 객체간의 정보 교환 루틴을 설정하는데 결정적인 도움을 주게 된다. 가령 어떤 함수 A와 함수 B간에 호출 관계가 있다면, 우리는 쉽게 함수 A를 포함하고 있는 객체와 함수 B를 포함하고 있는 객체간에 메시지 루틴이 설정되어야 한다는 사실을 알 수가 있다. 하지만 두 함수 A, B모두가 한 객체의 행위로 존재한다면 같은 모듈 안에서는 정보를 데이터를 통하여 서로 공유할 수 있기 때문에 메시지 전달 경로는 필요가 없게 된다.

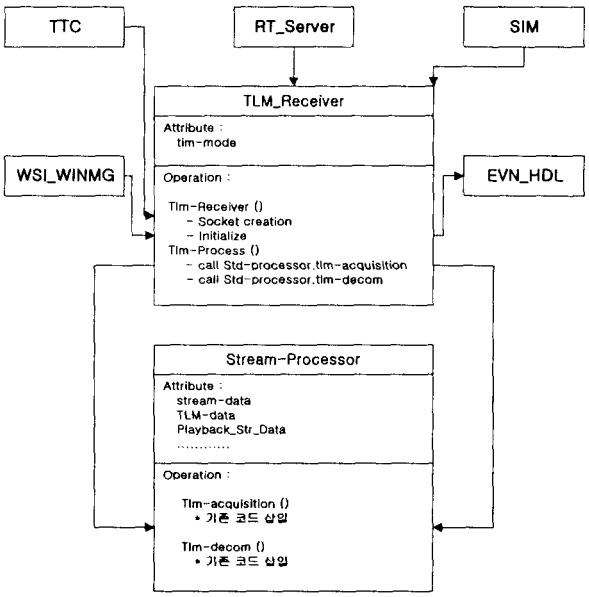

그림 6. Telemetry Receiver에 대한 OMT 객체 모델

4.6 객체 모델 구현

위에서 제시한 과정을 거치게 되면 객체 모델 구현에 필요한 대부분 요소들은 파악이 된다. 본고에서 제시한 방법에서는 객체간의 관계를 설정하기 어렵기 때문에 객체 지향 방법론에서 아주 중요시되는 객체간의 상속(inheritance), 결합(aggregation)관계 등은 직접적으로 파악하기가 어렵다. 물론 포함된 행위들 간의 관계에 대한 보다 많은 정보를 갖고 있다면 상속 관계의 설정이 불가능한 것은 아니다. 따라서 위에서 추출한 정보를 이용하여 이를 OMT의 객체 모델이나 Booch 모델 또는 UML 등으로 표현할 수가 있다. 그림 6은 위 과정에서 만들어진 OMT기법을 이용한 객체 모델이다. 물론 최종 산출물인 객체 모델은 draft한 것이기 때문에 개발자가 필요한 내용을 더 많이 추가하여 좀 더 발전된 객체 모델로 발전시킬 수가 있다.

5. 결론

소프트웨어의 크기가 증가하고 복잡해짐에 따라 유지 보수가 그 이전보다 점점 더 어렵게 되어 가고 있는데, 그 이유는 요구 사양의 변화에 프로그램의 규모가 커짐에 따라 수정하기가 훨씬 어렵기도 하지만, 한편으로는 함수 지향적 소프트웨어가 구조적으로 변화에 적응이 어렵게 되어져 있다. 1980년 후반부터 각광을 받고 있는 객체 지향 소프트웨어 개발 방법론은 여러 가지 면에서 우수하

다고 인정되었지만, 여전히 기존 시스템 개발자에게는 생소하고 어렵게만 느껴지고 있는 게 현실이다. 본 논문에서는 함수 지향적 프로그램을 객체 지향적 프로그램으로 전환할 수 있는 재공학 도구를 소개하고 있는데, 그 목적은 두 가지로 요약할 수 있다. 먼저 변화에 적응하지 못해 폐기 위기에 있는 함수 지향적 프로그램을 재사용하여 객체지향 프로그램으로 전환시킴으로써 재개발보다는 경제적으로 이익을 가져다 줄 수 있다. 또한, 부차적인 면으로는 객체 지향 프로그램 개발에 어려움을 겪고 있는 기존 프로그래머들이 자신이 개발한 함수 지향적 프로그램이 객체 지향 프로그램으로 변화하는 과정을 보면서 자연스럽게 객체 지향에 의한 방법론을 익히게 함으로써 새로운 기법에 적응해 가는데 도움을 줄 수가 있을 것이다. 이 방법 역시 경험적인 룰에 의존하기 때문에 완전한 전환을 기대하기는 어려워, 그 결과물을 프로토타입 또는 중간 생산물로 사용하여 시스템 개발자가 좀 더 보완해야 하는 단점이 있다.

감사의 글: 본 연구는 정보통신부 출연금의 지원에 의한 것입니다.

참고문헌

Ladden, R. M. 1988, Soft. Eng. Notes, 13, 3, pp.24-31

Rumbaugh, J., Blaha, M., Premerlani, M., Eddy, F. and Lorensin, W. 1991, Object-Oriented Modeling and Design, Prentice-Hall Inc.

Lee, S.B. & Carver, D. 1991, Journal of Object-Oriented Programming, Jan., pp. 35-43. Booch G. 1987, Software Engineering with ADA (2nd ed.), The Benjamin-Cummings Publishing Inc.

Bailin, S. 1989, Comm. of ACM, 32, 5. pp.608-623

Ward, P.T. 1989, IEEE Software, pp.74-82

Breuer, T. & Lano, K. 1991, Journal Software Maintenance: Research and Practice, 21, 12

Dietrich, W. C., Nackman, L. R., & Gracer, F. 1989, Proc. of OOPSLA

Jacobson, I. 1991, Proc. of OOPLSA, Association of Computing Machinery, N.Y.

Dunn, M. F. & Knight, J. C. 1991, Proc. of 13th Int'l Conf. On S.E., IEEE Computer Society Press, Los Alamitos, CA., 1991

Lee, S. B., Lee, J., Chi, D., & Lim, K. 1991, Proc. of InfoScience'93, (Seoul), pp.211-217