特輯

ATM 교환 소프트웨어 기술

金 漢 慶
韓國電子通信研究所

비동기식 전달모드(ATM)를 채택한 교환기에서의 소프트웨어 기술을 분야별로 검토함으로서 새로운 기술의 접목과 개발되어야 할 기술을 정리한다. 특히 메가프로그램이라고 지칭되는 교환기 소프트웨어 구조를 객체지향 개념에 입각하여 실현하고, 지능망과의 정합을 고려하여 설정하였으며, 소프트웨어 설계 등에 대한 기술적인 방법론을 제시한다.

Ⅰ 서 론

ATM 교환기는 지능망과 TMN 망에서의 중요한 노드로서의 역할을 담당하여야 하기 때문에 이들과의 정합을 염두에 둔 구조를 가져야 하며 앞으로 여러가지 새로운 기능을 수용할 수 있도록 하기 위해서는 기 개발된 프로그램의 재사용성을 가능한 높이는 것이 필요하다. 이러한 교환기 소프트웨어의 개발에 요구되는 소프트웨어 엔지니어링 기술, 교환 소프트웨어의 기능, 교환 소프트웨어의 구조에 대한 기술을 제시한다.

교환기를 제어하는 교환 프로그램에는 교환기능의 수용능력과 처리능력의 극대화와 함께 프로그램의 유지보수성, 개발 생산성, 각종 OAM 기능이나 새로운 서비스 추가의 유연성 등이 요구된다. 종래의 교환 프로그램은 성능을 중시하였으나 최근의 상황은 새로운 서비스가 신속하게 추가되어야 하고 개발 인건비의 상승이 뚜렷하기 때문에 높은 생산성 및 유지보수성이 요구된다. 이를 해결하는 기본적인 접근방법으로서 객체지향 분석 및 설계방법을 도입하였으며 이 객체지향에 대한 기본적인 개념은 문제의 구조에 따라서 소프트웨어를 구성하고자 하는 방법인데 megaprogramming으로 지칭되는 대형 소프트웨어 개발에 유용하다는 보고가 나오고 있다. 그러나 교환기와 같은 대규모 실시간 시스템을 객체지향 개념에 의해 모델링하는 것은 시스템이 갖는 다양한 기능 때문에 어려운 점이 있으며 기 발표된 기존의 객체지향 설계법은 교환기 시스템 개발에 바로 적용하기에는 구체적이지 못하다. 본 논문에서는 이와같은 요구와 문제 해결을 위해 객체지향형 교환 소프트웨어 기술에 대해 다음과 같이 객체지향 기법의 적용을 위한 개념, 교환기 소프트웨어가 담당해야할 호처리 및 교환기 운용 및 유지보수 기능, 그리고 교환 소프트웨어 구조에 대하여 차례로 기술한다.

Ⅱ. 교환 소프트웨어 기능

What we need is young men with old ideas (- Mort Sahl)

1. 연동 기능

ATM교환기가 연동되어야 할 망은 가입자망, 지능망, N-ISDN 망, PSTN 망, PSDN 망 등이 있으며 이들과의 연동 프로토콜은 (그림.1)과 같이 Meta Signaling, Q.93B, B-ISUP 등이 필요하다. 시스템의 유지보수를 위해서 시스템과 운용자 사이에 대화기능이 필요하며 이를 위해 Man-Machine Interface 메카니즘과 함께 Man-Machine Communication 프로토콜이 필요하다. 또한 망차원의 과금, 장애처리, 통계처리 등을 위해 TMN과의 망연동을 해야 한다.

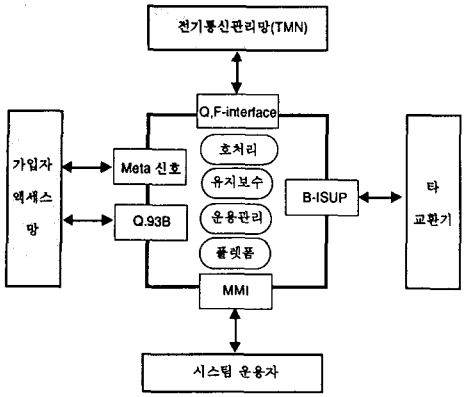

그림 1. ATM 교환 기능

2. 호처리 기능

호처리의 관점에서 볼때, 지능망의 도입이 확산되고 가속화되는 시점에서 B-ISDN 망의 노드 교환기가 가져야할 기능은 기본적인 스위칭 기능을 단위 모듈로 구성하고 이들 모듈과 지능망 노드들의 모듈들 사이에 직접적인 메시지 교환이 가능하도록 소프트웨어 구조를 미리 정립해야 한다. 이와 함께 셀의 라우팅 처리기능, 트래픽에 대한 폭주 및 감시기능, 각종 프로토콜 변환을 통한 중계기능, 멀티미디어 처리기능 등이 포함된다.

3. OAM 기능

시스템의 운용과 유지보수 관점에서 OAM 기능을 망 OAM 기능, 노드 OAM 기능, 모듈 OAM 기능의 3계층으로 분류한다. 망 OAM 기능은 망 차원의 기능을 담당하는 TMN의 기능으로서 여기에는 고장관리 기능, 성능관리 기능, 형상관리 기능, 과금관리 기능, 보안관리 기능이 있다. 망 OAM 기능을 지원하면서 교환기 시스템의 유지보수를 위한 노드 OAM 기능은 교환기 내의 관리 객체에 따라 여러 분야로 나누어진다. 즉, 상태관리, 시스템 시동, 장애관리, 시험 및 감시, 과부하 제어기능, 소프트웨어 변경에 대한 관리기능이 있으며, 교환기 시스템 운용자를 위한 운용기능으로서는 과금, 데이터 처리, MMC, 측정 및 통계 처리 등의 지원기능이 있다. 위의 노드 OAM 기능은 시스템 내의 각 모듈에 내장되어 있는 모듈 OAM의 구축으로서 가능하다. 모듈 OAM 기능을 잘 구축함으로서 시스템의 self-healing 기능은 극대화될 수 있다.

Ⅲ. 객체지향 분석

Exact scientific knowledge and methods are everywhere, sooner or later to replace rule-of-thumb. (- Frederick Taylor)

지금까지 개발되어온 교환 프로그램에 대한 유지보수의 어려움은 기능추가 또는 변경이 일어날 경우 그에 대한 파급효과가 어느 프로그램에 어떻게 영향을 미치는지 쉽게 파악하기 어렵다는 데에 기인하는데 이것은 교환기 개발에 있어 중요한 요인으로 작용한 것이 소프트웨어의 처리능력 향상과 메모리 용량 절감이었기 때문이다. 이러한 문제는 프로세서의 처리능력과 메모리 기술의 발전으로 외형적으로 해결되어 가고 있으며 이에 따라 교환기 소프트웨어의 개발은 곧바로 교환기 개발에 있어 구조적인 문제 해결에 역점을 두어야 한다. 앞에서 언급한 구조적인 문제를 해결하기 위한 방편으로 제안하는 것은 교환기의 논리모델과 프로그램 구조를 일치시키므로서 시스템 개념모델과 실현모델의 일치를 유도하는데 이는 실세계의 모델이 그대로 프로그램 구조로 실현됨을 의미한다. 이것이 객체지향의 개념이며 이에 따라 프로그램의 유지보수성과 기능추가의 유연성 향상이 가능하다. 논리모델과 프로그램 구조를 일치시키기 위한 방법으로 다음의 기술을 활용한다.

1. 계층화

교환 소프트웨어 구조를 구축하기 위해서는 교환기의 논리적인 구성 요소를 전체적으로 이해해야 하는데 이를 계층화(devide by layer)를 통해 구분함으로서 논리적인 요소의 객체화를 유도한다. 교환 시스템을 물리 계층과 제어 계층 그리고 서비스 계층으로 분리함으로서 우선 상상력과 인식의 어려움이 덜한 물리 계층의 자원에 대한 객체를 결정한다. 물리 계층을 두게되면 실제적인 하드웨어의 구성이 제어 계층과 서비스 계층으로 부터 은폐되어 하드웨어의 추가 또는 변경이 시스템 전체적인 차원에서 소프트웨어를 변경하지않고 물리 계층의 변경만으로 가능해진다. 제어 계층은 교환기가 담당해야할 외부와의 정합 기능을 분산된 제어체계 내에서 수행하기 위해 실현해야할 요소로서 결정된다. 즉, 제어계층은 교환기가 외부와의 인터페이스에 적용할 프로토콜의 처리를 담

당하는 계층이다. 일반적으로 제어계층은 event-driven에 의한 state transition machine의 구조를 취하게 되는데 외부와의 메시지를 주고 받으면서 서비스 계층과는 프리미티브를 통해 정보를 교환하게 되며 서비스 계층은 사용자 또는 운용자 차원의 서비스를 제어하기 위한 객체들로서 구축된다.

2. 세분화 (devide and conquer)

교환기 개발에 있어 문제영역을 나누어 각각의 문제를 해결하는 domain analysis technique이 필요하다. ATM교환기 소프트웨어 기술에서는 문제 영역을 세분하여 단위 문제를 function으로 정의한다. 따라서 이러한 function은 중복된 설계가 일어나지 않도록 분류되어야 하며 단일 주소공간 내에서 독립적으로 구현되는 것을 유도하기 위하여 한 사람의 개발자에 의해 설계 및 실현이 가능하여야 한다. 이러한 function은 계층화, 상세화를 통해서 정의된 객체를 기준으로 정형적으로 표현되고 설계되어야 한다. 이를 위해 sequence chart 및 state transition diagram을 function 단위로 작성하여 계층화 또는 상세화를 거쳐 정리된 객체를 서로 엮어 procedural하게 문제를 분석한다.

3. 상세화 (refinement)

시스템의 기능을 호처리, 운용, 보전, 운영체계, 데이타베이스 관리 등으로 나누고 이러한 분야에 대하여 또다시 구체적인 기능을 세분하여 서브시스템으로, 이를 다시 하위의 구체적인 설계 단위인 블럭으로 정의를 하고 이를 실현시키기 위하여 유니트로 상세화시켜 나가는 작업이 필요하다. 이와 같이 상세화를 통하여 논리 모델을 구축하고 이를 객체화하는 것은 앞에서 설명한 계층화 및 세분화하는 방법을 보완해 주는 것으로서 복합 객체의 확인과 계승 체계의 수립을 지원한다. 설계가 진행됨에 따라 추가로 객체를 정의해야 하는 경우와 기 정의된 객체의 기능이 복합적이어야 하는 경우가 발생한다. 이러한 과정을 거쳐 복합객체, 능동객체, 병렬객체, 수동객체 등이 정리된다.

Ⅳ. 교환 소프트웨어 구조

Unavoidable price of reliability is simplicity (- C. Hoare)

1. 시스템 모델링

1) 개념 구조

(그림.2)에서 보여주듯이 ATM 교환기의 개념적인 구조는 고전적인 교환방식 개념과 마찬가지로 정합(interface), 집선(concentration), 분배(distribution), 확대(expansion) 부분으로 구성되며 분배부분을 기준으로하여 서로 대칭의 형태를 가짐으로서 half-call 처리 구조가 유용하게 실현될 수 있음을 보여주며 이들의 기능을 제어할 소프트웨어가 실장될 프로세서를 분산 구조로 구성하였다. 정합부분에 실현되어야 할 소프트웨어 기능으로서는 Meta signaling 처리기능과 Q.93B, CCS No.7, TMN과의 정합을 위한 F 및 Q 인터페이스 프로토콜 처리기능, 호수락 제어(call admission control), 사용자 파라메타 제어(usage parameter control), 가입자 감시 및 시험(supervision and test), 폭주제어(congestion control) 기능이 있으며, 집선부분에는 self-routing 처리기능, 우선제어(priority control) 기능이 실현되어야 하며, 분배부분에 스위칭 기능, 번호번역 기능, 자원할당(network resource management) 기능, OAM 기능이 필요하다. OAM 기능은 시스템의 유지보수를 위한 각종 지원기능들로서 시스템 로딩, 운용자 정합, 과금 및 통계, 시험 및 감시 기능의 지시와 종합 관리 역할을 담당한다.

2) 소프트웨어 플랫폼(Platform)

시스템에 대해 외부로 부터 요구되는 기능을 개발하기 위해서는 시스템의 기본적인 토대 위에서 사용자가 요구하는 사항만 응용하는 것이 재사용의 측면에서 요구된다. 이러한 재사용은 임의의 한 프로젝트 내에서는 재사용의 효율을 기대하기 힘들지만 여러가지의 통신 시스템 또는 여러가지의 프로젝트를 지속적으로 추진하는 관점에서는 매우 필요하다. 이러한 점을 고려하여 소프트웨어 플랫폼을 구축하고 모든 응용 소프트웨어를 이 플랫폼 위에서 구축한다. 이 플랫폼에 포함되는 내용은 시스템 기능수행을 위해 내부적으로 정의되어야 하는 기능들로서 운영체제, DBMS, 화일처리기, 객체간 통신처리기 등이 있다. 일반적으로 운영체제는 시스템 개발 또는 구조와 관련하여 설계하지 않으며 운영체계가 담당해야 하는 기능도 통신시스템의 서비스 소프트웨어의 관리이거나 또는 하드웨어를 관리하는 소프트웨어들에 대한

관리이다. 즉, 이중화 관리, 메시지 통신관리, 프로세스 스케쥴링 기능등이 운영체계의 기능으로 제공된다. 주기억장치에 상주하는 데이타베이스 관리시스템으로서 실시간 처리가 보장되는 관계형 데이타베이스 관리 시스템이 데이타의 일치성, 보안성 등의 기능지원을 위해 소프트웨어 플랫폼으로서 제공된다.

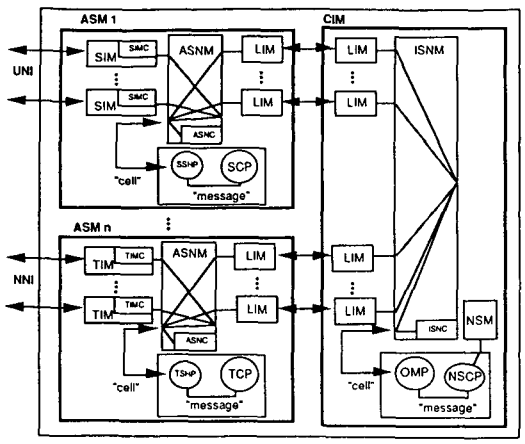

그림 2. ATM 교환기의 개념 구조도

2. 응용 소프트웨어 구조

1) 계층적 구조

교환기 소프트웨어의 관리상 계층 구조는 크게 두가지로 나눈다. 첫째는 설계 및 개발 측면에 중점을 둔 구조(S-con)이고, 둘째는 제품 관점에서 바라본 구조(P-con)이다. 전자의 경우는 최고 상위레벨인 시스템으로 부터, 논리적 및 물리적 resources를 관장하는 서브시스템, 기능적 특성을 고려한 블럭, 소프트웨어 개발결과인 real product에 해당하는 유니트들로 계층화한다. 블럭들은 실제 설계 및 구현되는 논리적 단위로서 각 설계자들에 의하여 개발된다. 후자의 경우는 설계 및 개발된 소프트웨어 제품이 실제 구성되어져 있는 계층적 구조를 표현한 것으로 최상위 레벨에 프로세서를 두며, 그 아래에 그 프로세서에 포함되어 있는 즉 로딩 단위인 블럭을 둔다. 그리고 각 블럭들을 이루고 있는 다양한 화일들이 마지막 레벨로 계층화된다.

실시간 시스템의 객체지향 설계기술은 대상시스템의 모델링에 많은 영향을 받는다. 그 때문에 객체지향 설계에서는 객체를 선택하는 것이 가장 중요하며 어려운 문제이다. 객체의 선택을 효율적으로 그리고 쉽게 하기 위해서 (그림.3)에 ATM교환기 연구시제품에 적용할 소프트웨어의 계층모델을 나타내었다. 통신서비스는 통신 하드웨어를 통해 제공이 되는데 이 하드웨어에 무관하게 통신 서비스가 제공될 수 있도록 통신 하드웨어에 대한 information hiding이 요구된다. 즉 통신 하드웨어를 추상화하여 하드웨어의 물리적인 특성을 은폐시키므로서 하드웨어와 서비스 제어를 분리시킨다. 하드웨어에 대한 추상화는 하드웨어 자체에 대한 형상관리와 하드웨어 제어를 위한 제어방식으로 분리 되는데 제어방식을 프로토콜로 추상화시키고 하드웨어 형상관리를 하드웨어 객체로 추상화시켜 이들을 각각 프로토콜 계층과 물리계층에 둔다. 이와 같은 물리계층과 프로토콜 계층의 분리 및 하위계층과 서비스 계층의 분리를 통하여 virtual hardware를 구축함으로서 시스템의 형상이 바뀌어도 소프트웨어에는 영향을 받지 않도록 투명성을 보장한다.

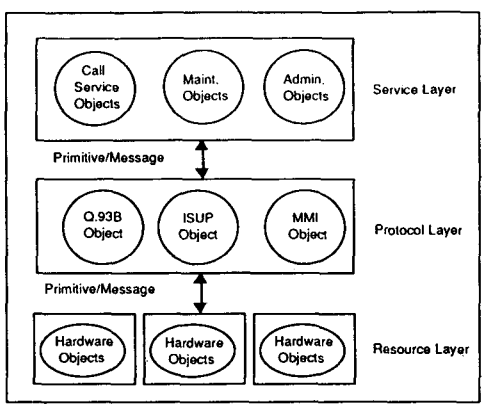

그림 3. ATM 교환기 소프트웨어 계층 모델

2) 동시처리 및 분산처리 구조

일반적으로 병렬처리를 하기 위한 프로그램 구조는 두가지로 생각할 수 있다. 첫째는, 응용 프로그램을 여러개의 순차적인 프로세스로 구성하고 프로그램 모니터를 만들어 이를 통해 다중처리를 하는 방법이 있으며 둘째는, 스래드 개념에 입각한 복수개의 스래드를 객체안에 정의하고 수행시에 병렬처리하는 방법이다. ATM 교환 소프트웨어에서는 이 두가지 방법이 다 활용된다. 호처리와 같은 다중처리의 성격이 분명한 기능과 관련된 객체는 복합객체로 구성하는데 복합객체란 객체내에 동시처리되어야 할 병렬객체를 encapsulation시키고 로더에 의해 로딩이 된 다음

run time시에 복합객체가 wake-up되면 복합객체의 내부처리에 의해 병렬객체가 동적으로 creation 및 binding 된다. 병렬객체는 다른 복합객체의 병렬객체와 비동기 통신을 하는데 이 메시지에 의해 병렬객체의 method가 invoke된다. 반면에 프로그램 모니터를 이용하는 방식은 소프트웨어 엔지니어에게 보다 많은 설계능력을 요구하는 방법으로서 하나의 프로그램을 여러개의 순차처리 객체로 구성하여야 하며 내부의 조건 변수 또는 동기화 변수를 두어 각 객체에서 이 변수에 조건을 기록하면 모니터가 이에 따라 queue에 있는 객체를 concurrent하게 invocation 한다.

3. 교환 소프트웨어 객체 설계

1) 능동객체(Active Object)

교환기는 동시에 수많은 호를 실시간으로 다중 동시처리해야 하므로 동시처리(concurrent processing)가 가능한 객체를 필요로 한다. 병렬객체의 도입은 소프트웨어 모델링에 있어 시스템을 간결하게 구축할 수 있으나 context switching 등으로 인한 성능저하가 초래된다. 동시처리가 가능한 프로세스를 정의할 수 있는 객체를 능동객체라고 하는데 이 능동객체는 독립적인 주소공간을 갖게 되며 서로 다른 주소공간에 있는 프로세스와는 병렬로 수행되므로 병렬객체간의 통신은 비동기 통신방식을 취한다. 이는 프로세스가 스케줄링되면 자신의 메시지 버퍼에서 메시지를 읽어내어 메시지가 요구하는 처리를 수행한다. 프로세스가 preemption 되었다가 다시 스케줄링된 경우에는 인터럽트 당한 지점부터 처리를 재개한다. 프로세스는 자신의 프로세스 제어 블록(Process Control Block)과 스택을 가지므로 해서 실행시에 context switching이 가능하다. 이것은 메모리 양과 실행시간에 대한 오버헤드가 발생하므로 교환 프로그램에 요구되는 소프트웨어를 전부 능동객체만으로 실현하는 것은 어렵다.

2) 소프트웨어 백플레인 (Backplane)

추상화를 통한 시스템 어트리뷰트를 객체화시킴으로서 소프트웨어 재사용을 유도하고 또한 개발된 소프트웨어 객체의 정확도를 활용하여 새로운 기능의 추가나 변경을 용이하게 한다. 이와같은 개념을 지원하는 소프트웨어 백플레인은 수동객체로 구성되는데 수동객체로 구현되는 대상은 일반적으로 시스템의 기능에서 procedural abstraction이 가능한 부분을 1차적으로 선택하며 2차적으로는 시스템 데이타로서 다수의 능동객체가 공유해야 하는 부분을 선택한다. 이러한 소프트웨어 백플레인은 기능과 활용도에 따라 2가지의 라이브러리로 구분이 되는데 첫째, application domain 내에서의 재사용을 위한 delegate object library 둘째, 프로세서 boundary내에서의 재사용을 위한 object library 등으로 구분한다. 이러한 지역성은 분산처리 구조, 동시 및 병행처리 구조를 지원해야 하는 교환 소프트웨어의 특징을 반영한 것이다. Application domain 내에서 객체의 호출은 procedure call과 동일하며 object library는 프로세서에서 shared memory에 위치한다. Object library에 존재하는 객체의 호출은 객체 naming table을 참조하여 delegation jump를 통해 제어가 이전된다. Object library에 있는 객체가 수행중일 때에는 preemption되지 않는다.

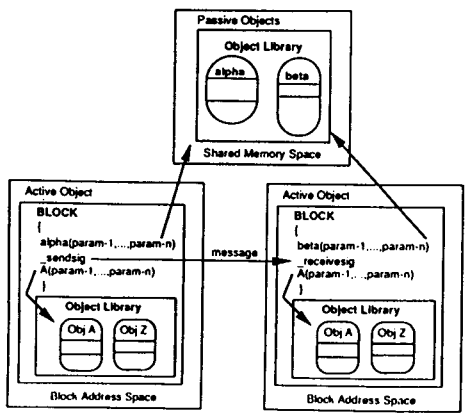

그림 4. 객체의 구성

3) Message Passing

병렬객체에 의해 생성된 프로세스 사이에 교환되는 메시지의 종류는 대체적으로 3가지가 있다. 첫째는 수신측 프로세스에서 접수될 메시지의 type을 지정하여 자신에게로 전달을 요구하는 initial message가 있으며 이 initial message에 의해 발신 프로세스를 확인하고 착신측의 프로세스 id를 통보함으로서 발신 및 착신 프로세스 상호간에 필요한 정보교환을 가능하게 하는 normal message가 있다. 이와 같은 메시지 교환은 비동기적으로 이루어지게 된다. 반면에 공유 메모리 방식으로 정보교환을 수행할 수 있는데 이것은 critical region에 발신 프로세스가 메시지를

기록한 다음에 수신 프로세스가 읽어갈 수 있도록 하는 방식이다. 그러나 공유 메모리 방식에서 더 나아가 메시지를 기록하거나 읽어내는 것을 operation을 통하여 수행하도록 내부 규정을 만들고 이를 객체화함으로서 재사용이 가능하도록 하여 initial 또는 normal message에 의한 통신이 갖는 성능상의 단점을 개선하며 이들을 이용한 통신은 동기적으로 수행되도록 하였다. 이러한 통신 방식들이 프로세서 영역을 넘어서게 되면 프로세서간 통신으로 확대해야 하는데 이를 위해 메시지 송신 프리미티브에 프로세서 번지를 지정할 수 있도록 구성한다. 이외에도 메시지의 종류로는 timeout 처리를 위한 timesig가 있다.

4. 소프트웨어 개발방법론

블럭으로 체계화되는 능동객체의 설계는 블럭 내부의 프로세스, 프로시듀어를 소프트웨어 유니트 개념의 물리적인 화일로 정의하는 과정이다. 여기에는 초안의 프로세스, 프로시듀어, 메시지 및 DB relation을 상세화하는 과정이 포함된다. 블럭 설계를 위해 CCITT 표준 언어인 SDL을 사용하여 블럭을 기술하며, 이렇게 SDL로 표현된 블럭들을 모으면 실현되어질 소프트웨어 모델이 생성된다. 이러한 SDL 모델을 CCITT에서 권고한 CHILL 언어로 코딩하여 화일을 구성한 다음 컴파일 및 링크를 통해 시스팀에 로딩되는 단위인 실행모듈을 구축한다. 프로세서의 로딩 단위인 실행모듈의 구성을 보면 프로세서별로 한 개 이상의 실행모듈로 실현하며 실행모듈간의 공통 데이타는 허용하지 않는다. 각 로딩 단위간의 통신은 메시지에 의하여 가능하도록 한다. 또한 프로세서간 공통 블럭은 별도 실행모듈을 구성하며, 서브시스팀간의 실행모듈은 분리하여 실현한다. 실행모듈의 구성은 실시간 시스팀에서 프로세스 스택 구성을 위한 메모리의 증가, 프로세스 switching에 따르는 오버헤드 증가 등으로 인한 시스팀의 성능에 영향을 미치므로 실행모듈 정의시 세심한 고려를 하여야 한다.

V. 결 론

... the question is not "Will there be a management elite?" but "What sort of elite will it be?" (- Sidney Webb)

ATM 교환 소프트웨어 개발을 위하여 교환기능의 실현을 위한 객체지향 분석, 설계 및 실현에 대한 기본적인 개념을 정리하였다. 이는 기존의 교환 소프트웨어 개발에서 시스팀의 성능향상을 위해 구조적이지 못하고 모듈화하지 못한 부분에 대하여 객체지향 개념을 이용하여 구조화와 모듈화를 확고히 하고 하드웨어 기술의 발전을 이용하여 소프트웨어 공학적인 관점을 견지함으로서 소프트웨어 유지보수성을 향상시키고, 기능의 추가 및 새로운 프로젝트를 추진할 경우에 소프트웨어 재사용성을 극대화시키기 위한 방편을 제공하고자 한 것이다. 앞에서 제시된 구조는 아직도 정리되어야할 부분이 많으며 앞으로도 수정 및 보완이 요구되는 것임을 밝힌다.

용어정리

ATM	Asynchronous Transfer Mode
B-ISDN	Broadband Integrated Services Digital Network
CCITT	International Telegraph and Telephone Consultative Committee
CCS	Common Channel Signalling
CHILL	CCITT High Level Language
DBMS	Database Management System
OAM	Operation And Maintenance
PSTN	Public Switched Telephone Network
PSDN	Packet Switched Data Network
N-ISDN	Narrowband Integrated Services Digital Network
SDL	Functional Specification and Description Language
TMN	Telecommunication Management Network

參 考 文 獻

[1] Katsumi Maruyama, et al., "Concurrent object-oriented switching program in CHILL," 13th ISS proceedings, 1990.

[2] E.C. Arnold, D.W. Brown, "Object-oriented software technologies applied

to switching system architectures and software development processes," 13th ISS proceedings, 1990.
[3] 한국전자통신연구소 TDX개발단, "TDX-10 소프트웨어 설계 지침서," 1990. 10.
[4] G. Booch, "Object-oriented design with applications," Benjamin/Cummings, 1990.
[5] 고광호, 김봉수, 김한경, "광대역 통신망의 OAM 기능을 위한 개념 정립," 한국전자통신연구소 전자통신동향분석 제8권 제2호, 1993. 7.
[6] 이병윤, 김봉수, 김한경, "객체지향 개발방법론," 한국전자통신연구소 주간기술동향 93-13, 1993.
[7] 김영부, 이성창, 한치문, "완전분산형 구조를 갖는 ATM 교환 시스템," 제3회 통신정보합동학술대회(JCCI-'93) 논문집 제3권, 1993. 4.
[8] 한국전기통신공사 광대역ISDN개발사업추진단, "ATM교환기 사용자 요구사항," 1993.8.

筆者紹介

金 漢 慶

1951年 2月 15日生
1973年 2月 서울대학교 공과대학 원자력공학과 학사
1987年 8月 충북대학교 대학원 전자계산학과 석사
1992年 ~ 현재 충북대학교 대학원 전자계산학과 박사과정

1973年 3月 ~ 1973年 10月 한국스페리 주식회사
1973年 10月 ~ 1976年 5月 군중앙경리단
1976年 5月 ~ 1977年 9月 한국스페리주식회사
1977年 9月 ~ 1983年 2月 한국전자통신주식회사
1983年 3月 ~ 현재 한국전자통신연구소 ATM 운용보전연구실장

주관심 분야 : 객체지향 프로그래밍, 소프트웨어 규격명세, 통신 프로토콜