플랫폼독립형 클래스저장소에 기반한 TMN 분산객체 디자인 방법론

정회원 이 광 형*, 박 수 현**

A Design Methodology of TMN Distributed Object based on Platform Independent Class Repository

Kwang-Hyung Lee*, Soo-Hyun Park** *Regular Members*

요 약

여러 통신망을 총괄적이고 효율적으로 운용하고자 출현한 TMN(Telecommunication Management Network) 은 구축과정에서 서로 다른 하드웨어와 운영체제 등의 상이한 플랫폼 환경 하에서 개발되는 관계로 분산객체내 클래스의 개발 및 유지보수에 여러 문제점을 내포하게 된다. 대표적인 문제점으로는 TMN 시스템내의 모든 에이전트들이 동일한 기능을 수행하는 소프트웨어 및 데이터 블록들을 중복하여 유지해야 한다는 점을 들 수 있다. 이로 인하여 TMN 에이전트의 개발에 있어 Q3 인터페이스 구현상의 표준을 이룰 수 없을 뿐만 아니라 다중 플랫폼을 지원할 수 없게 된다. 이러한 문제들을 해결하기 위하여 Farmer 모델에 기본을 둔 Farming 방법론을 제안하였다. Farming 방법론은 각각의 분산객체에 중복되어 저장되어 있는 소프트웨어 및 데이터 컴포넌트들을 플랫폼에 독립적인 컴포넌트웨어 형태로 변형하여 플랫폼독립형 클래스저장소(PICR)에 저장시켜 놓은 후 각 분산객체내의 프레임워크에 명시된 대로 실행에 필요한 컴포넌트웨어들을 PICR에서 정적 또는 동적으로 로딩하여 사용하는 것이다. Farmer모델을 이용하여 개인휴대통신망의 분산 TMN 에이전트를 디자인하고 구현하였다.

ABSTRACT

The TMN that appears to operate the various communication networks generally and efficiently is developed under the different platform environment such as the different hardware and the different operating system. One of the main problems is that all the agents of the TMN system must be duplicated and maintain the software and the data blocks that perform the identical function. Therefore, the standard of the Q3 interface development cannot be defined and the multi-platform cannot be supported in the development of the TMN agent. In order to overcome these problems, the Farming methodology that is based on the Farmer model has been suggested. With the Farming methodology, the software and the data components which are duplicated and stored in each distributed object are saved in the platform independent class repository(PICR) by converting into the format of the independent componentware in the platform, so that the componentwares that are essential for the execution can be loaded and used statically or dynamically from PICR as described in the framework of each distributed object. The distributed TMN agent of the personal communication network is designed and developed by using the Farmer model.

* 동서대학교 정보시스템공학부(shp@hyomin.dongeui.ac.kr)
** 동의대학교 공과대학 컴퓨터응용공학부(khlee@kowon.dongseo.ac.rk)
논문번호 : 99213-0528 접수일자 : 1999년 5월 28일

Ⅰ. 서 론

여러 통신망을 총괄적으로 관리함으로서 일원화되고 효율적인 통신망 운용관리를 지향하는 개념으로 출현한 TMN은 구축 과정에서 여러 가지 서로 다른 운영체제 및 이기종의 하드웨어 플랫폼을 이용하여 개발되어지는 관계로 TMN 에이전트의 클래스를 개발하고 유지 보수하는 단계에서 여러 문제점을 내포하게 된다[1][2][3]. 첫째, 에이전트들내 클래스들이 자신의 하드웨어나 운영체제 등의 플랫폼에 종속성을 갖고 구현, 유지 보수되어지는 관계로 다수의 에이전트 및 매니저들은 동일한 기능을 수행하는 서로 다른 클래스들을 모두 중복하여 유지해야만 하며 이로 인하여 전체 망관리 입장에서는 동일한 기능을 수행하는 클래스의 버전을 관리하기가 용이하지가 않다. 두 번째, 다중플랫폼(multi-platform)의 지원이 불가능하게 되며 궁극적으로는 DCN(Data Communication Network)과 OS(Operation System), MD(Mediation Device), NE (Network Element) 등과의 Q3 인터페이스를 구현 시 일관된 인터페이스를 제공하기 어렵다. 세 번째, 나아가 서로 다른 망은 각각의 망관리 시스템을 유지하고 있기 때문에 집중화된 망관리 시스템인 TMN을 구축시 이와 같은 서로 다른 망의 유지보수시스템 사이의 호환성이 보장되지 않는다[4][5][6].

임의의 시스템을 설계하는 프로그래밍 기법으로서 80년대 중반 이후 가장 많이 사용되는 방안이 바로 객체지향 분석모델(Object-Oriented Analysis & Modeling)[7][8][9][10][11]이다. 그러나 이러한 OOAM도 실세계의 개체를 단지 하나의 고정된 측면에서만 표현할 수 있기 때문에 그 개체의 설계 및 모델링함에 있어 그 개체를 이해하는 데 도움을 주고 필요한 다른 여러 측면을 무시하게 된다. 이와 같은 문제를 해결하기 위해 측면지향 프로그래밍 (Aspect-Oriented Programming)[12]을 고려할 수 있는 데 이는 실행코드로 구성된 모듈을 통합하여 시스템을 구축시 발생하는 기능혼합(Tangling)[13]의 발생을 막기 위해 측면 요약으로 서술한 후 위버(Weaver) 라는 인테그레이터로 통합하여 최종 실행코드를 생성해 내는 새로운 개념의 프로그래밍 기법이기는 하지만 이 역시 실세계의 개체를 단지 하나의 고정된 측면에서만 표현할 수 있고 실세계가 가지고 있는 계층성을 체계적으로 반영할 수 없는 단점이 있다. 또한 시스템을 분석 접근함에 있어 개체의 측면만을 식별할 수 있고 특수한 언어로 사용하여 나타낸 측면서술들은 측면 위버(Aspect Weaver)라는 통합자(integrator)를 통하여 측면서술이 완료된 시점에서 통합이 이루어진다.

기존의 TMN 에이전트들은 DSET, RETIX, Openview[2][3] 등과 같은 에이전트 생성 툴킷을 이용하여 생성하였으나 여러 가지 서로 다른 운영체제 및 이기종의 하드웨어 플랫폼하에서 생성되는 관계로 앞에서 언급했던 여러 문제점들을 가지게 되었다.

본 논문에서는 매니저와 에이전트 등 분산객체 내의 소프트웨어 컴포넌트들을 컴포넌트웨어 (componentware) 형태로 생성하여 플랫폼독립형 클래스 저장소 (PICR : Platform Independent Class Repository)[4][5]에 유지시켜 필요시 이러한 컴포넌트 웨어들을 분산객체로 동적 또는 정적으로 다운로딩하여 실행시키는 Farming 방법론[4][5][6][14]을 제안하였다. Farming 방법론은 지식표현 모델인 Farmer 모델에 기본을 두고 있다.

본 논문에서는 Farming 방법론을 이용하여 PCN (Personal Communication Network)의 TMN 에이전트를 설계 및 구현을 예시하였다.

Ⅱ. Farmer 모델

Farmer 모델은 실세계의 개체를 각각의 고정된 측면이 아닌 여러 관점의 측면에서 분석한 후 분석 완료된 측면요소들을 측면객체로 정의한다. Farmer 모델은 시스템 개체구조(System Entity Structure)의 개념을 도입한 지식표현을 위해 사용되는 프레임 구조모델로서 지식표현 모델인 EA (Entity-Aspect) 모델[14][15]에 기본을 두고 있다. Farmer 모델의 주요 목적은 실제 디자인 하고자 하는 에이전트를 분석하여 에이전트를 구성하고 있는 컴포넌트 요소들을 측면에 따라 분리, 추출해 내는 데 있다. 이러한 결과 추출된 컴포넌트 요소들은 Farmer 모델 트리의 리프노드에 위치하게 되며 이러한 컴포넌트 요소들은 네트워크를 통하여 최종적으로 PICR에 저장이 된다. 또한 Farmer 모델은 PICR에서 컴포넌트 요소를 에이전트로 동적 또는 정적으로 다운로딩하는 Farming의 개념을 추가한 형식모델이다. Farming 방법론의 핵심 개념은 Farming이며 Farming 방법론의 기본적인 프로그래밍 패러다임으로 개체측면지향 프로그래밍(Entity-Aspect Oriented Programming)[14]을 도입하고 있다. Farmer 모델은

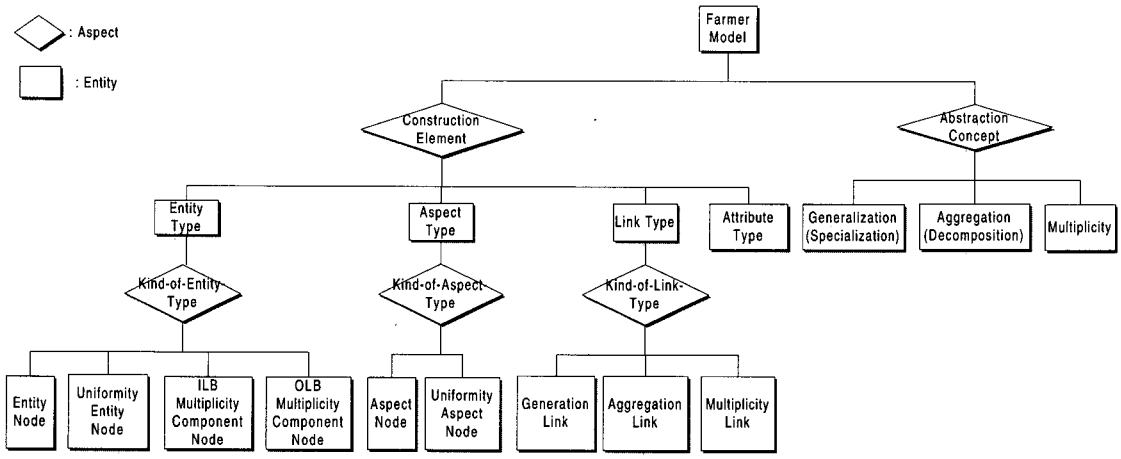


그림 1. Farmer 모델의 기본개념 분류

바로 EAOP의 기본모델이 된다.

Farmer 모델은 실세계의 객관적, 추상적 대상체를 표현하는 개체노드 타입 (entity node type), 이러한 개체를 표현하는 관점인 측면노드 타입 (aspect node type), 개체와 측면간의 관계성을 나타내는 링크타입 (link type) 그리고 개체가 가지는 성질을 나타내는 속성타입 (attribute type), 동일한 이름을 가지는 2 노드는 동일한 속성과 동일한 서브트리를 갖는다는 균일성의 원리에 의해 정의되는 균일성 개체노드 타입 (uniformity entity node type)과 균일성 측면노드 타입(uniformity aspect node type) 그리고 Farming 시 ILB 컴포넌트에 해당하는 속성을 지니는 IM-컴포넌트 타입 노드 (IM component type node : ILB Multiplicity Component type node), OLB 컴포넌트의 속성을 지니는 OM-컴포넌트 타입 노드(OM component type node: OLB Multiplicity component type node) 등의 구성요소들과 generalization, aggregation, multiplicity 등의 3 가지 추상화 개념들로 구성된다. 이들 중 IM-컴포넌트 타입 노드와 OM-컴포넌트 타입 노드는 다중성 개념과 관련되며 다중성 집합(Multiplicity set)의 요소들이 된다. IM-컴포넌트 타입노드와 OM-컴포넌트 타입노드들은 다중성 인스턴스 링크 (Multiplicity Instance Link)에 의해 상위 개체노드와 연결이 된다. 그림 1은 이러한 Farmer 모델의 개념을 Farmer 모델 다이어그램을 이용하여 보여주고 있다.

IS-A 관계를 나타내는 generalization 은 공통의 성질을 갖는 서브클래스들을 하나의 클래스로 통합하는 추상화개념이며 specialization 은 이 개념의 반대 개념으로 하나의 클래스를 상호배제적인 서브클래스들로 분류하는 추상화 개념이다. Specialization 개념은 개체 성질의 상속성(inheritance)을 포함한다.

Aggregation은 A-PART-OF 관계로서 여러 개의 개체나 개념을 서로 통합하여 새로운 하나의 통합된 개체나 개념을 만들어 내는 추상화 개념이며 이 개념의 역관계를 갖는 개념은 decomposition이 된다.

Multiplicity 는 하나의 개체를 구성하기 위하여 동일한 형태의 구성요소가 여러 번 발생하는 경우에 이의 대표적인 요소만을 표시하는 추상화 개념이다. Farmer 모델의 구성요소와 기호는 표 1과 같다.

Farmer 모델에서의 개체노드는 독립적으로 식별되는 구성원소 또는 실세계의 대상체(real world object)를 분할시킬 때 그 대상체를 구성하는 구성원소가 된다. 측면노드는 한 개체를 여러 관점으로 관찰할 때 관점개개가 된다. 따라서 한 측면노드에서의 개체노드들은 그 측면에서의 구조적 관계를 나타내는 분할된 구성원소가 된다.

Farmer 모델을 이용하여 TMN 에이전트 등과 같은 네트워크를 기본으로 하는 시스템을 설계, 구현 시 Farmer 모델의 핵심이 되는 측면(aspect) 개념을 반영하기 위하여 새로운 개념의 객체를 필요로 한다. 이러한 필요성에 의해 창출된 신 개념의 객체를 측면객체(Aspect-Object : AO)이라 부른다. AO는 모델링하고자 하는 실세계의 모든 대상체를 의미하며 실세계의 객체를 이해하는 모델링의 관점에 따라 여러 측면에서 기술되어 질 수 있다.

표 1. Farmer 모델 구성기호

Construction Elements & Abstraction Concepts	Symbols
Entity	Entity Name
Uniformity Entity	Entity Name
Aspect	Aspect Name
Uniformity Aspect	Aspect Name
OM-Component	O Component Name
IM-Component	I Component Name
Attribute	~ Attribute Name
Decomposition Link (Aggregation Link)	
Specialization Link (Generalization Link)	
Multiplicity Link	
Multiplicity Instance Link	
Connector	

Farmer 모델의 주요 특징은 디자인 완료된 시스템을 구현 시 이미 컴포넌트웨어 형태로 만들어져 있는 소프트웨어 및 데이터 요소들을 LAN 이나 인터네트를 통하여 외부의 저장소로부터 아웃소싱(outsourcing)해 가지고 오는 것이다. JAVA머신과 같은 네트워크 컴퓨터 (NC : Network Computer) 의 경우에는 Farmer 모델에 의하여 디자인된 시스템을 구현 시 모든 소프트웨어 블록들을 아웃소싱해야 하며, 아웃소싱이 필요없는 경우에는 OM-컴포넌트 타입 노드와 IM-컴포넌트 타입노드가 필요없게 된다.

Ⅲ. Farming 방법론

TMN 에이전트를 디자인하고 구현함에 있어 Farming 방법론을 도입할 경우 다음과 같은 점들을 고려해야만 한다. 우선 임의의 네트워크에 TMN을 구축할 경우 네트워크내의 어떠한 요소들을 관리대상으로 할 지에 대하여 결정을 내려야 한다. 다음으로 관리대상으로 정한 네트워크 요소들을 구성하는 소프트웨어자체도 TMN내의 관리 요소로 고려해야 한다. 이를 위하여 PICR 개념의 도입이 바람직하다.

마지막으로 소프트웨어 아키텍처의 개념을 TMN 에이전트 개발과 관련된 방법론으로 고려해 볼 수 있다. Farming 방법론의 프레임워크 및 Farming의 개념은 소프트웨어 아키텍처의 개념을 충실히 반영하고 있다.

다음과 같은 Farming 방법론의 6 단계는 이상의 고려사항들은 반영하고 있다.

3.1 1 단계 : 네트워크 분석 및 관리대상 네트워크 요소의 선택

본 단계에서는 네트워크내의 어떠한 요소들을 관리대상으로 할 지에 대하여 결정을 내려야 한다. 이와 관련한 해답은 ITU-T 권고안 "M.3100 Generic Network Information Model"에서 찾을 수 있다.

이 모델에서는 관리대상이 되는 관리객체에 대하여 다음과 같은 여러 관점을 가지고 있다.

- Network Element Level Viewpoint
- Network Level Viewpoint
- Service Level Viewpoint

현실적으로 네트워크 구성요소들간의 관계나 위상(topographical) 관계까지 TMN에서 관리할 수는 없기 때문에 본 단계에서는 Network Element Level Viewpoint의 관점을 채택하였다.

예를 들어 PCN TMN 에이전트 구축의 예를 보기로 하자. PCN 분석의 결과로서, 그림 2에서 보는 바와 같이 관리대상 네트워크 요소로서 Mobile Switching Center (MSC), Base Station Center (BSC), Home Location Register (HLR) 그리고 InterGate Subsystem (IGS)를 들 수 있다. 이에 따라 PCN TMN 시스템은 MSC 에이전트, BSC 에이전트, HLR 에이전트, 그리고 IGS 에이전트로 구성이 되며 이들은 다음과 같은 3가지 기능들을 수행한다.

- MSC, BSC, HLR, 그리고 IGS의 TMN 에이전트 기능
- MSC와 IGS의 과금데이타 전송기능
- HLR 의 고객데이타 통신기능 및 HLR 시스템 관리기능

PCN교환기 내의 MSC, BSC, HLR 및 IGS 에이전트들은 오류처리, 형상, 과금, 성능관련 서비스 품질 유지, 고객정보에 대한 데이터 보호 (FCAPS) 등을 CMIP, FTAM 프로토콜을 이용 종합적으로

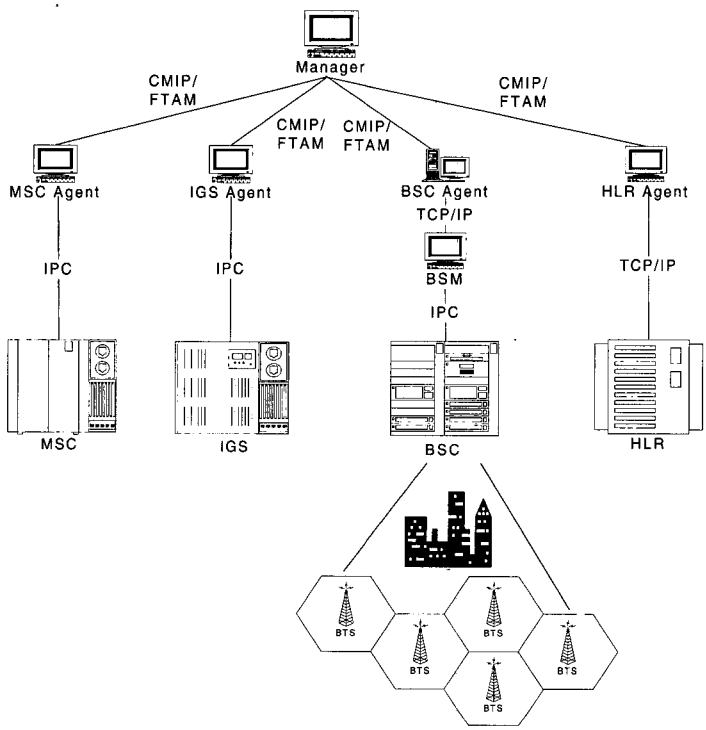


그림 2. PCN TMN 구성

관리함으로서 통화품질 및 서비스 품질 향상에 기여 관리하는 기능을 수행한다. 그림 3의 step 1은 이러한 과정을 보여주고 있다.

3.2 2 단계 : 분산객체 시스템의 분석 및 컴포넌트웨어 요소 추출

이 단계에서는 TMN 에이전트가 요구하는 기능영역을 정의함으로서 설계에 필요한 요구사항을 확정짓는 단계로서 개발하고자 하는 분산객체 시스템이 요구하는 기능영역을 정의함으로서 설계에 필요한 요구사항을 확정한다. 또한 현재의 시스템 스펙을 소프트웨어 재사용 개념을 사용하여 미래의 요구사항에 대응할 수 있도록 확장이 가능하게 하고 다른 사용자의 재사용을 지원할 수 있도록 분석한다. 기능영역을 분석할 때 분산객체가 수행해야 하는 기능을 초기로딩 기능 블록 (ILB : Initial-Loading function Block)과 요구부 로딩 기능블럭 (OLB : On-demand Loading function Block)으로 구분하여 분석을 하는데 프레임워크의 구현과 관련된다.

ILB는 시스팀의 근간을 이루는 주요 모듈로 I/O 인터페이스, 시스팀의 동작을 드라이브하는 부분을 의미한다. 이 블록은 필요할 때마다 동적으로 다운로딩 하여 사용하기에는 파일의 크기가 너무 크고 호출되는 빈도가 매우 많기 때문에 프레임워크의 구성 시에만 로딩이 이루어지는 블록이다. ILB는 다음과 같이 정의된다.

【정의 1】 초기로딩 기능블록(Initial-Loading function Block)

초기로딩 기능블록(Initial-Loading function Block)은 다음과 같이 정의된다.

$\mathbf{M} \subset \mathbf{C}, \forall c \in \mathbf{C}, \forall b \in I$ 에 대하여

$\mu_1 : C \rightarrow I,\ \mu_1(c) = b$

이때 b는 다음의 조건을 만족해야 한다.

1) 블록 b의 크기와 관련된 조건 : Size(b) $\geq \mu$
2) 블록 b의 로딩되는 빈도수 : Load_frequency (b) $\geq \zeta$
3) 중요도 : Degree_of_importance(b) $\geq \delta$

where, C : 컴포넌트 집합

μ : 주요기능 블록 크기의 임계값

ζ : 로딩빈도수 임계값

δ : 에이전트 내에서의 중요도 ■

즉 ILB b로 정의되기 위하여는 block b의 파일

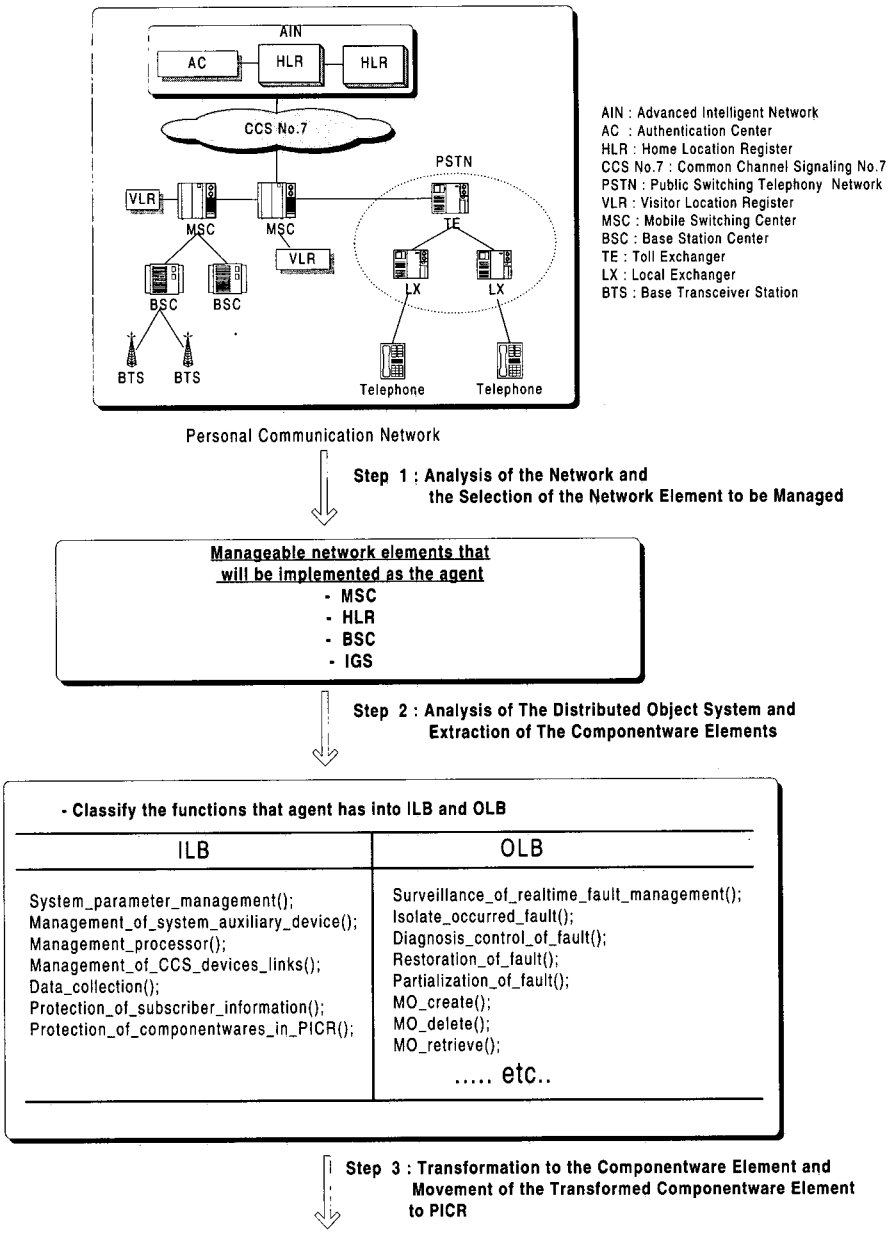


그림 3. Farming 방법론에 의한 PCN TMN 시스템의 설계 및 구현 (계속)

크기가 μ이상인 경우와 운영 중에 PICR로부터 에이전트로 다운로딩되는 빈도수가 ζ 보다 큰 경우가 된다. δ는 block b의 중요도와 관련된 값으로 다운로딩 실패시 재전송의 허용여부 등과 관련된 값이다. μ, ζ, δ의 값은 시스템의 상황에 따라 가변적으로 결정된다.

비교적 적은 비중을 차지하는 기능을 수행하는 기능블럭들은 요구부 로딩 기능블럭 (OLB : On-demand Loading function Block)으로 정의되며 이들 블록들은 에이전트내의 드라이버인 Farmer의 요구에 의하여 PICR로부터 다운로딩되어 수행된다. OLB의 정의는 다음과 같다. OLB는 ILB와 비교시 비교적 작은 크기의 컴포넌트웨어로 분산객체의 기능 수행 시 동적으로 로딩이 이루어지는 블록으로

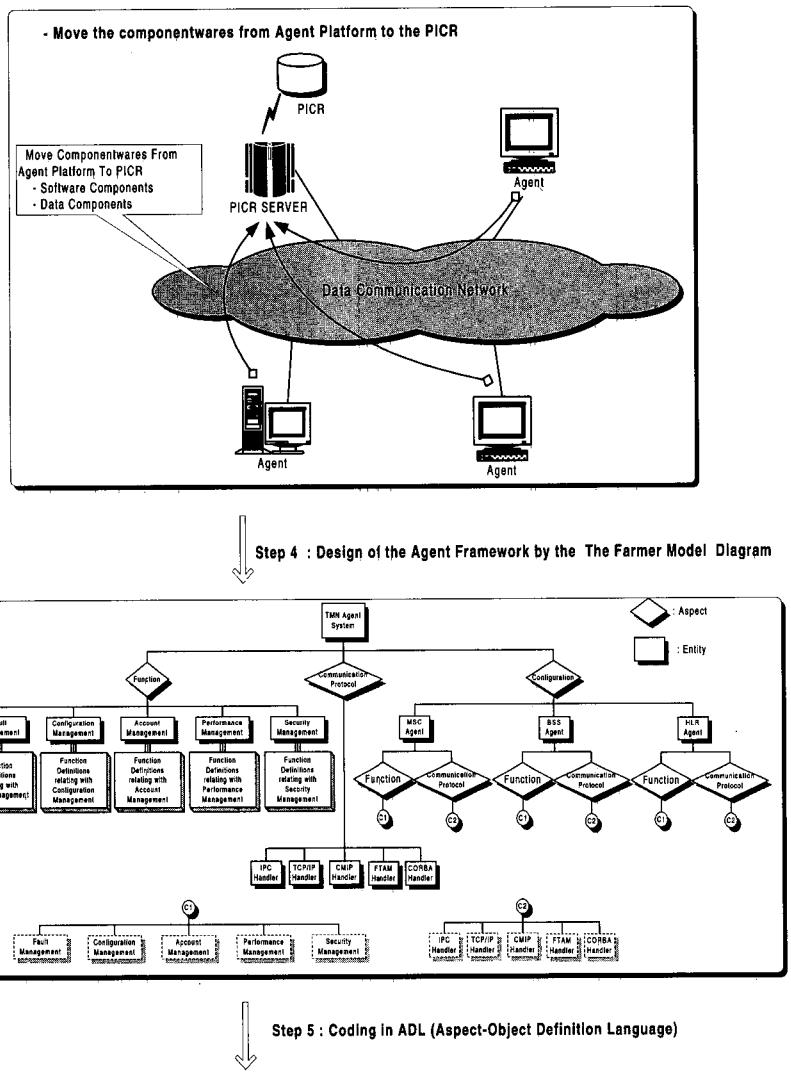


그림 3. Farming 방법론에 의한 PCN TMN 시스템의 설계 및 구현 (계속)

호출되는 빈도가 ILB에 비해 비교적 적다.

【정의 2】 요구부 로딩 기능블럭 (OLB : On-demand Loading function Block)

요구부 로딩 기능블럭 (OLB : On-demand Loading function Block) O는 다음과 같이 정의된다.

O $\subset$ **C**, $\forall c \in$ **C**, $\forall b \in$ O 에 대하여

$g : C \rightarrow O$, $g(c) = b$

이때 b는 다음의 조건을 만족해야 한다.

1) 블록 b의 크기와 관련된 조건 : Size(b) $< \mu$

2) 블록 b의 로딩되는 빈도수 : Load_frequency(b) $< \zeta$

3) 중요도 : Degree_of_importance(b) $< \delta$

where, C, μ, ζ, δ : **정의 1** 참조 ■

[예 1] ILB와 OLB

실제 PCN TMN 에이전트에서 사용되는 대표적인 컴포넌트들을 **정의 1**과 **정의 2**에 의해 ILB와 OLB로 분류하면 **표 2**와 같다. 이러한 기능블럭들은 PICR에 저장되어 있다. ■

본 단계에서는 Farmer 모델에 의하여 TMN 에이전트 시스템을 3 가지 측면(aspect)-기능측면(Function Aspect), 통신프로토콜 측면(Communication Protocol Aspect) 그리고 형상 측면(Configuration

```
CLASS
 PCS_TMN_Agent = A-OBJECT;
  ASPECT : Function, Communication_protocol, Configuration;
  ATTRIBUTE : ;
  DATA-DCL : ;
  METHODS :
       Fault_management ( Function : ASPECT; ); {
         Surveillance_of_realtime_fault_management(sub);
         Isolate_occurred_fault(sub);
                  ...
       }
       Configuration_management ( Function : ASPECT; ); {
         System_parameter_management(main);
         MO_create(sub);
                  ...
       }
       Account_management ( Function : ASPECT;); {
         Data_collection(main);
         Billing_data_transmission_to manager_system(sub);
                  ...
       }
                  ...
       IPC_Handler ( Communication_protocol : ASPECT;);
       TCP/IP_Handler ( Communication_protocol : ASPECT;);
       CMIP_Handler ( Communication_protocol : ASPECT;);
       FTAM_Handler ( Communication_protocol : ASPECT;);
       CORBA_Handler ( Communication_protocol : ASPECT;);
  SUPER :
  SUB :
       IGS_Agent(Configuration);
       MSC_Agent(Configuration);
       BSC_Agent(Configuration);
       HLR_Agent(Configuration);
END PCS_TMN_Agent;
```

Step 6 : Implementation of The Framework

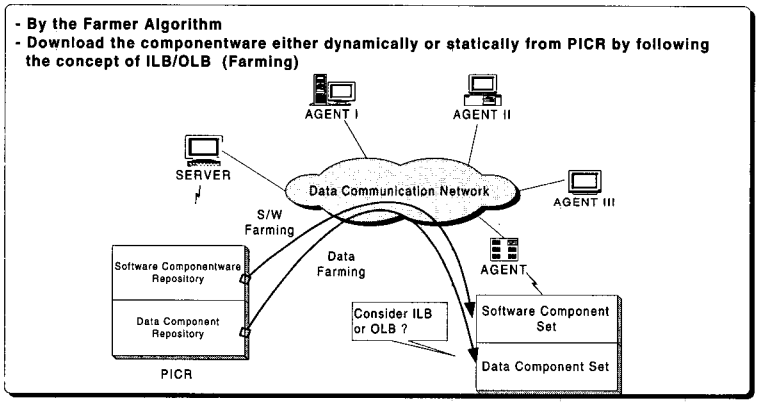


그림 3. Farming 방법론에 의한 PCN TMN 시스템의 설계 및 구현

Aspect)-으로 기능영역을 정의한다. 그림 4는 TMN 에이전트 시스템을 바라보는 3 가지 측면을 보여주고 있다.

1) 기능측면 (Function Aspect)

PCN교환기 내의 MSC, BSC, HLR 및 IGS 에이전트들은 오류처리, 형상, 과금, 성능관련 서비스 품질 유지, 고객정보에 대한 데이터 보호 (FCAPS) 등을 CMIP, FTAM 프로토콜을 이용 종합적으로 관리함으로서 통화품질 및 서비스 품질 향상에 기여 관리하는 기능을 수행한다.

기능측면에서 고려해야 할 기능 및 컴포넌트웨어 요소는 다음과 같다.

■ 장애관리기능 (Fault Management)
- 실시간 장애 감시기능 (Surveillance of Real Time Fault)
- 발생장애에 대한 격리 (Isolation of Occurred Fault)
- 장애진단 및 제어기능 (Diagnosis & control of fault)
- 발생장애에 대한 신속하고 정확한 복구기능 (Restoration of fault)
- 장애 국부화 기능 (Partialization of fault) 등

■ 구성관리기능 (Configuration Management)
- 시스템 파라미터 관리기능 (Management Sys-

표 2. ILB와 OLB

ILB	OLB
System_parameter_management(); Management_of_system_auxiliary_device(); Management_processor(); Management_of_CCS_devices_links(); Data_collection(); Protection_of_subscriber_information(); Protection_of_componentwares_in_PICR();	Surveillance_of_realtime_fault_management(); Isolate_occurred_fault(); Diagnosis_control_of_fault(); Restoration_of_fault(); Partialization_of_fault(); MO_create(); MO_delete(); MO_retrieve(); Billing_data_transmission_to manager_system(); Billing_data_retransmission_to_manager_system(); Asynchronous_report_of_extempore_billing(); Surveillance_of_performance(); Print_out_collected_dat_relating_with_performance(); Measuring_quality_of_service();

tem Parameter)
- 시스템 주변장치 관리기능 (Management of System Auxiliary Device) - MTU, DISK, I/O port등
- 구성하는 프로세스에 대한 관리기능 (Management Process)
- No.7 신호장비 및 링크관리 기능 (Management of Common Channel Signaling Devices and Links) 등

■ 과금관리기능 (Account Management)
- 관리시스템의 과금 전송요구에 대한 전송기능 (Billing Data Transmission to Manager System)
- 관리시스템의 과금 재전송 요구에 대한 재전송 기능 (Billing Data Re-transmission to Manager System for The Response of The Re-transmission Request)
- 즉시과금 비동기적 통지기능 (Asynchronous Report of Extempore Billing) 등

■ 성능관리기능(Performance Management)
- 성능관련 데이터를 수집하는 성능감시기능 (Surveillance of Performance)
- 성능관련 수집 데이터 출력기능 (Print-out Collected Data Relating with Performance)
- 서비스 품질 측정기능 (Measuring Quality of Service) 등

■ 보안관리기능
- 가입자 정보보호 (Protection of Subscriber Information)
- 플랫폼 독립형 클래스저장소내의 컴포넌트웨어 보호 및 버전관리 (Protection of Componentwares in PICR) 등

2) 통신프로토콜 측면 (Communication Protocol Aspect)

이 측면은 인터페이스와 관련하여 2 종류의 하위 측면으로 구분할 수 있다. 관리객체 에이전트 인터페이스와 관련된 측면과 매니저 인터페이스와 관련된 측면으로 구분할 수 있다. 관리객체 에이전트 인터페이스는 MSC 인터페이스, BSC 인터페이스, HLR 인터페이스 그리고 IGS 인터페이스로 구분할 수 있다.

MSC 인터페이스는 MSC와 MSC 에이전트 사이에서 IPC (Inter Process Communication) 프로토콜을 이용하여 MSC 관리정보와 과금 데이터를 상호 중계하는 역할을 하며 BSC 인터페이스는 BSC 에이전트와 BSM 사이에서는 TCP/IP 프로토콜을 이용하여 그리고 BSM과 BSC 사이에서는 IPC를 이용하여 BSC 관리정보를 상호 인터페이스시켜준다. 또한 TCP/IP 프로토콜은 HLR과 HLR 에이전트사이에서 HLR 관리정보를 송수신하기 위하여 사용된

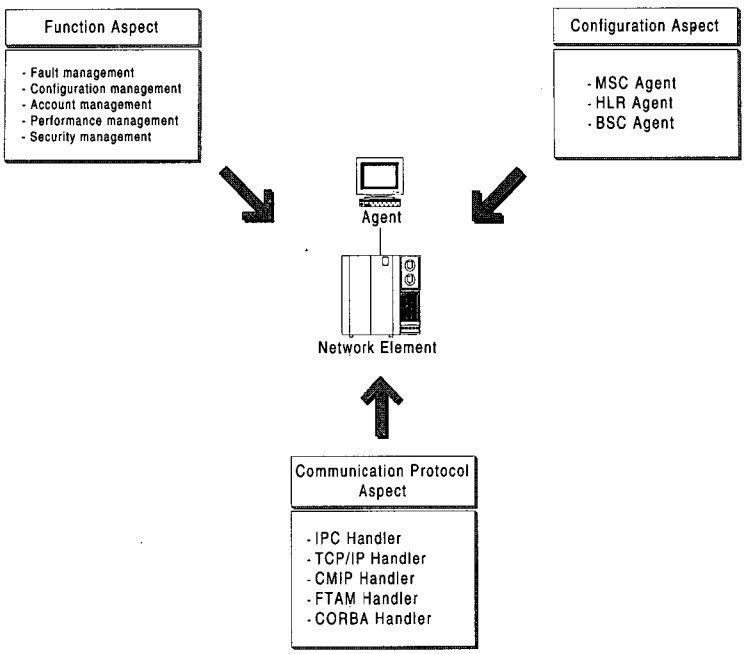


그림 4. TMN 에이전트의 관찰측면

다. IGS 관리정보와 과금데이터는 IPC에 의하여 IGS 에이전트와 IGS 사이에서 송수신 되어진다. 매니저와 에이전트사이에서 네트워크 관리 시스템과 과금관리시스템 그리고 고객등록 시스템을 위한 인터페이스는 CMIP (Common Management Information Protocol), FTAM(File Transfer Access and Management) 등의 프로토콜을 사용한다.

3) 형상측면 (Configuration Aspect)

이 측면은 PCN TMN 에이전트를 구성형상 측면에서 보여주고 있다. PCN TMN 시스템은 형상측면에서 분석하여 보면 MSC 에이전트, BSC 에이전트 그리고 HLR 에이전트로 특수화(specialize) 시킬 수 있다. 각각의 이러한 에이전트들은 측면요소로서 기능측면과 통신프로토콜 측면을 갖는다. 기능측면과 통신프로토콜 측면 요소들은 상위 레이어에 이미 존재하는 측면요소들과 동일한 이름을 가지기 때문에 Farmer 모델의 균일성 공리에 의하여 상위의 동일한 이름을 가지는 측면요소들의 서브트리들은 하위의 기능측면 요소와 프로토콜 측면요소로 상속되어진다.

그림 3의 step 2는 이러한 과정을 보여주고 있다.

3.3 3 단계 : 컴포넌트웨어 요소로의 변환 및 변환된 컴포넌트웨어 요소의 PICR로의 이동

2 단계에서 ILB와 OLB로 분류된 기능 블록들을 컴포넌트웨어로 변환하여 망관리 시스템내에 존재하는 PICR로 이동시켜 저장시키는 단계이다. 이전 단계에서 영역분석을 통하여 객체가 추출되면 추출된 객체를 기능요소별 컴포넌트웨어와 이에 필요한 데이터 요소로 매핑을 수행한다.

컴포넌트웨어는 소프트웨어 컴포넌트와 데이터 컴포넌트로 분류가 된다. 기존의 기능블럭들을 컴포넌트웨어화하는 단계에서는 CWC[16][17] 등의 재공학(re-engineering), 역공학(reverse-engineering) 등의 개념을 이용하며 실제 컴포넌트화 작업은 본 연구 범위에 포함시키지 않는다.

이 단계에서 PCN TMN 에이전트의 기능을 수행하는 데 필요로 되어지는 모든 기능 블록들은 컴포넌트웨어 형태로 변형되어지며 변형된 컴포넌트웨어들은 PICR 로 이동되어 저장된다.

PICR에서 저장되어 있는 컴포넌트웨어들도 또한 Farmer 모델에 의하여 3 가지 측면 - 기능 측면(function aspect), 레이어측면(layer aspect) 그리고 통신프로토콜 측면(communication protocol aspect) - 으로 특수화되어 있다.

그림 3의 step 3은 이러한 과정을 보여주고 있다.

3.4 4 단계 : Farmer 모델 다이어그램에 의한 에이전트 프레임워크의 디자인

본 단계에서는 에이전트 프레임워크를 Farmer 모

델 다이어그램을 이용하여 디자인한다. 이 논문에서는 PCN TMN 시스템을 주요기능 측면, 통신프로토콜 측면 그리고 형상측면 등과 같은 3가지 측면으로 분석하였으며 의해 Farmer 모델 다이어그램을 이용하여 디자인하였다.

PCN TMN 시스템을 구축시 MSC, BSC, HLR 그리고 IGS의 기능 및 하드웨어 디바이스 등은 관리객체로 정의되며 기간망(backbone)으로서의 DCN은 전용회선이나 LAN 등의 통신망을 활용하게 된다.

그림 3의 step 4는 이러한 과정을 보여주고 있다.

3.5 5 단계 : 측면객체 정의언어(ADL)로 코딩

본 단계에서는 4 단계에서 Farmer 모델 다이어그램에 의해 디자인된 에이전트 프레임워크를 Farmer 모델을 지원하는 측면객체정의언어(ADL : Aspect-Object Definition Language)[14] 에 의해 코드화 시킨다. 컴포넌트웨어들은 결국에는 각각의 분류된 기능요소들에 해당하며 이러한 프레임워크들이 측면객체(Aspect-Object)로 정의된다.

그림 3의 step 5는 이러한 과정을 보여주고 있다.

3.6 6 단계 : 프레임워크의 구현

이전 단계에서 ILB와 OLB로 분류, 추출된 블록들은 PICR에서 소프트웨어 컴포넌트를 로딩하는 단계에서 차이를 보인다.

프레임워크 리스트를 디자인 시에 ILB로 분류된 블록들은 소프트웨어 컴포넌트와 이와 관련된 데이터 컴포넌트의 Farming에 의하여 실제로 프레임워크를 구현 시에 응용프로그램의 코드로 반영이 되어 분산객체 내에 위치하게 된다.

OLB로 정의되는 세부기능들은 담당하는 블록들은 분산개체의 요구에 의해(on-demand) 클래스 또는 메소드 저장소에서 동적으로 로딩받아 수행이 이루어진다. 이와 같은 기능블럭들은 JAVA[18]를 이용하여 컴포넌트웨어로 변형되어 PICR에 위치하게 된다. 그림 3의 step 6은 이러한 과정을 보여주고 있다. Framework을 구현하고 실행되는 동안에 에이전트내의 드라이버인 Farmer는 다음과 같은 Farmer Algorithm을 이용한다. Farmer 알고리즘을 간략하게 서술하면 다음과 같다.

```
1     read ILB/OLB table
2     download ILB from PICR to the local library in
      the agent
3     Do Forever
3.1       receive a request(the CMIP message) from
          Manager
3.2       request와 일치하는 componentware의 속성
          을 check 한다.
3.3       if componentware의 속성이 ILB의 속성과
          같다면
3.3.1       local class repository 에서 해당하는 컴
            포넌트웨어를 찾아Farmer로 넘겨준다.
3.4       else if componentware의 속성이 OLB 이면
3.4.1       해당하는 컴포넌트웨어를 PICR로 on-
            demand loading요구를 함
3.4.2       PICR 로부터 수신한 컴포넌트웨어를
            Farmer로 넘겨준다.
3.5       end if
3.6       수신받은 componentware를 수행
3.7     end receive
4     End forever
```

< Farmer Algorithm >

Ⅳ. PCN TMN 에이전트 시스템의 구현

Farming 방법론의 5 단계에서 생성된 ADL은 에이전트를 구현하기 위한 프레임워크로서 사용되며 이러한 프레임워크를 읽어 실행시키는 에이전트 구성자(constructor)를 구현한다. 에이전트 구성자는 다음과 같은 기능을 수행한다. 그림 5는 이러한 에이전트 구성자의 메인 윈도우를 보여주고 있다.

에이전트 내에서 신규로 필요한 에이전트를 J++ Builder를 이용하여 생성해낸 다음 생성된 ILB/OLB 컴포넌트를 PICR에 이동(move)시키기 전에 에이전트 플랫폼 내의 로컬 라이브러리에 저장해두는 기능을 수행한다.

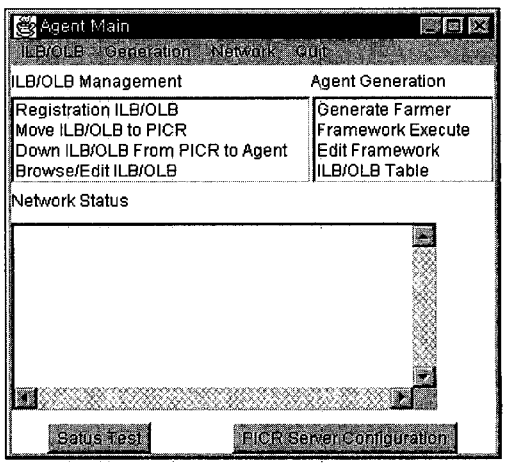


그림 5. 에이전트 Constructor

그림 5의 ILB/OLB Management window 내의 **"Registration ILB/OLB"** 항목이 이에 해당하며 이는 그림 6으로 연결된다. 이밖에 TMN 시스템 관리자와 협의하여 PICR에 저장할 수 있는 허가를 받은 ILB/OLB 컴포넌트들을 PICR 서버로 전송하는 기능**(Move ILB/OLB to PICR)**, PICR 에 저장되어 있는 ILB/OLB 컴포넌트를 에이전트 플랫폼으로 다운로딩하는 기능, 수정되거나 신규로 등록되는 컴포넌트를 PICR에 등록하는 기능들이 있다. 그림 5의 메인 윈도우내의 **"Generate Farmer"** 항목은 Farmer class를 생성해낸다. Farmer class 또한 컴포넌트 형태로 PICR 에서 에이전트로 가져와 실행을 시키게 된다. Farmer의 생성결과는 그림 5에서 보여주는 에이전트 constructor main window의 status message window 에 display 된다. 이 밖에 프레임워크 실행 기능, ILB/OLB 테이블 리스트 기능, 에이전트와 PICR server사이의 통신채널의 상태를 확인하는 기능, TMN시스템내에 존재하는 모든 PICR server에 대한 정보를 보여주는 기능 등을 제공한다.

본 연구에서는 TMN PICR로서 Solaris 2.5를 운영체제로 사용하는 SUN Ultra SPARC I를 이용하여 구현하였다. 또한 매니저 및 에이전트로서 AMD -PR 166 (Windows 95), Pentium 166 (Windows NT), Pentium PRO 200 (Windows NT), SUN Ultra SPARC I (Solaris 2.5)를 플랫폼으로 사용하였다.

V. 성능평가

이번 장에서 Farming 방법론을 실제 TMN 에이전트의 구축에 적용(이하 Farming 방식)하였을 때 나타나는 결과에 대하여 Farming의 개념을 적용하지 않고 개발(이하 non-Farming 방식)된 TMN 시스템과 비교하여 평가하였다. 평가의 기준으로는 망에서 발생하는 CMIP 메시지, 온라인 통지 메시지(on-line notification message) 등과 같은 트랜잭션 처리에 대한 응답시간으로서 전체 시스템의 성능을 고려하였다.

5.1 트랜잭션 응답시간에 따른 성능분석

CMIP 메시지 처리 및 에이전트에서 발생하는 통지 메시지 등을 처리하는 트랜잭션 t의 실행과 관련된 시뮬레이션 평가모델을 정의한다. 본 논문에서는 성능요소로서 각 분산객체 내에서의 CPU 처리시간, PICR 접근시간, 네트워크 요소들간의 통신시간 등을 고려하였으며 네트워크 토폴로지는 고려하지 않았다. 본 시뮬레이션의 모델은 그림 7과 같은 분산환경이 된다.

그림 7에서 매니저가 임의의 CMIP 메시지를 TMN 내의 임의의 에이전트로 전송하는 경우 이를 처리완료하는 데 소요되는 시간을 CT_{t_CMIP} 으로 정의하였다. CT_{t_CMIP} 는 매니저, 에이전트, PICR 서버와 같은 각 노드들의 CPU시간과 네트워크를 통한 노드들 사이의 전송시간, PICR 디스크 입출력 시간, 에이전트와 실제 리소스(real resource) 사이의 통신시간, 실제 리소스의 처리시간의 합으로 표현할 수 있다.

TMN 에이전트들은 실제 리소스에 오류가 발생하는 경우라든가 주기적인 상태보고 등과 같은 온라

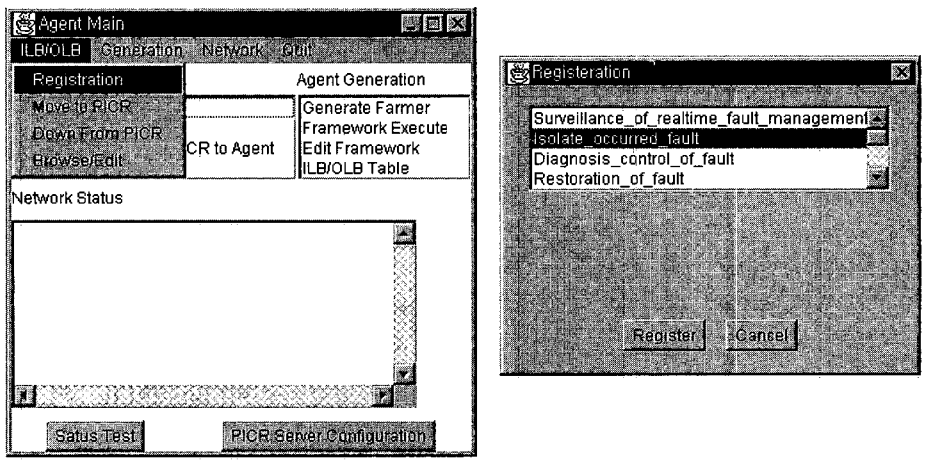


그림 6. Registration ILB/OLB

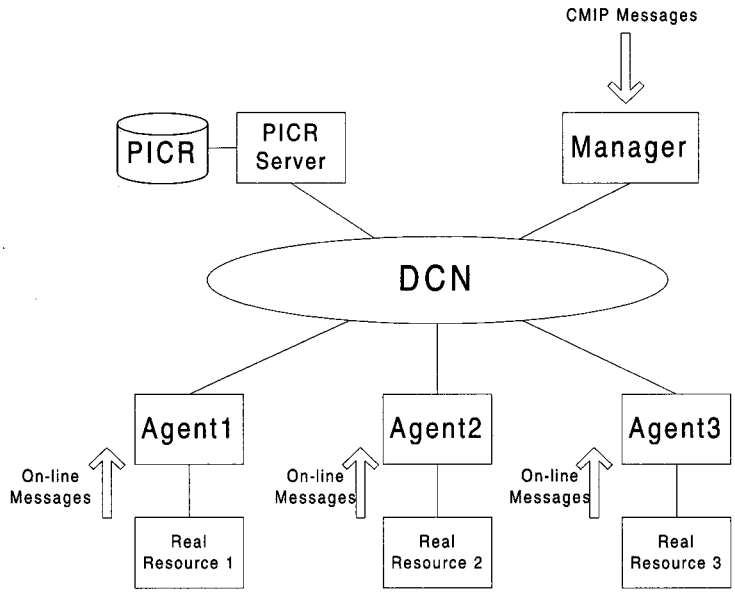


그림 7. Simulation 환경

인 통지 메시지를 만들어 매니저로 보고를 한다. $CT_{t_notification}$은 이러한 온라인 통지 메시지를 처리완료하는 데 소요되는 시간으로 정의된다. $CT_{t_notification}$는 매니저, 에이전트, PICR 서버와 같은 각 노드들의 CPU시간과 네트워크를 통한 노드들 사이의 전송시간, PICR 디스크 입출력 시간, 에이전트와 실제 리소스사이의 통신시간의 합으로 표현할 수 있다.

따라서 트랜잭션 t 의 실행을 완료하는 데 소요되는 전체 처리시간을 의미하는 CT_t 는 다음과 같이 정의된다.

$$CT_t = CT_{t\_cmip} + CT_{t\_notification}$$

본 논문에서 제안한 Farming 방식을 이용하여 구현된 TMN 시스템의 시뮬레이션을 위해 Sun-Sparc 3에서 실행되는 SLAM II와 FORTRAN 77을 이용하였다. 적용대상으로 고려된 TMN 시스템은 그림 7과 같이 1 개의 매니저와 3 개의 에이전트 그리고 1 대의 PICR들을 연결하는 통신링크로 구성되어 있다. 본 적용대상에서는 매니저에서 발생하는 CMIP 메시지, 실제 리소스 1, 2, 3에서 발생하는 통지 메시지 등이 있다. 본 실험에서 사용되는 에이전트 별 통지 메시지 발생 빈도 R_O 와 매니저에서 발생된 CMIP 메시지의 에이전트별 처리 빈도 R_C 및 시스템 매개변수의 값은 표 3, 표 4와 같다.

표 3. R_C 및 R_O

발생 메시지 타입	빈도수 관련				
	발생장소	발생장소별 RO	처리장소 별 RC		발생메시지 갯수
CMIP	Manager	0.0	Agent 1	0.5	10^3
			Agent 2	0.3	
			Agent 3	0.2	
On-line	Agent 1	0.46	0.0		10^3
	Agent 2	0.29			
	Agent 3	0.25			

표 4. 실험에서 사용된 시스템 매개변수의 값

(단위 : ms)

매개변수	manager	agent1	agent2	agent3	PICR server
CPUT(x)	2.0	2.5	2.0	3.0	2.0
ACC(p)	0.0	0.0	0.0	0.0	1.0
Tm(r)	0.0	0.0	0.0	0.0	0.23

본 수치들은 ATM TMN 시스템의 실측치로부터 얻었다. $COM_T(s, d)$와 같은 노드사이의 통신시간은 네트워크 종속값으로서 본 실험에서는 DCN을 Frame Relay(1.544 Mbps or 2.048 Mbps), FDDI or FDDI-II (100 Mbps), ATM(155 Mbps), Giga-Bit Net(2G)를 사용하는 경우로 조정해 가면서 시

표 5. COMT(s, d)의 매개변수 값

(단위 : ms)

DCN	전송속도
Frame Relay	2,590 1,953
FDDI FDDI-II	40
ATM	25.0
Giga-Bit Net	2.0

스템의 전체 성능을 비교하였다. 네트워크에서 통용되는 메시지의 평균 사이즈를 500K로 가정할 경우 각 프로토콜별로 다음과 같은 $COM_T(s, d)$ 값은 표 5와 같다. PICR과 PICR server 사이의 통신 시간인 $COM_T(p, s)$ 값은 1 ms로 정의하였다. 그림 8은 표 5에 나타난 대로 $COM_T(s, d)$의 매개변수 값의 변경하여 가며 실험한 결과 나타난 CMIP 메시지 처리 및 에이전트에서 발생한 통지 메시지를 처리하는 전체 트랜잭션의 평균 응답시간을 비교한 그래프이다. 실험결과 Farming 방식을 이용하는 경우 저속망에서는 non-Farming 방식과 많은 응답시간의 차이를 보이지만 고속망으로 갈수록 그 격차는 줄어드는 것을 볼 수 있다. 그림 9와 그림 10은 에이전트에서의 CMIP 메시지 처리 시의 평균 응답시간 및 온라인 메시지 처리 발생 시의 평균응답 시간을 보여주고 있다. 이들 그림 또한 그림 8에서와 같이 고속망으로 갈수록 Farming 방식과 non-Farming 방식의 격차가 줄어들음을 알 수 있다.

5.2 결과분석

그림 8, 그림 9 그리고 그림 10에서 보는 바와 같이 DCN을 구성하는 망이 ATM 이상의 고속망으로 발전 시 Farming 방식과 non-Farming 방식사이에는 성능에 있어 큰 차이를 보이지 않게 됨을 알 수 있다. 하지만 개발비용 면에서 본다면 우선 하드웨어 플랫폼 측면에서 저가의 네트워크-접속 컴퓨터(Network-Connected PC)를 하드웨어 플랫폼으로 사용하는 Farming 방법론이 현재와 같이 고가의 워크스테이션을 하드웨어 플랫폼으로 사용하는 non-Farming 방식에 비하여 비교우위에 있게 된다. 또한 플랫폼 독립적인 언어를 이용하여 TMN 에이전트를 구현하는 관계로 non-Farming 방식에 비해 소프트웨어 개발비용 측면에서도 Farming 방법론이 비교우위에 있게 된다. 따라서 100Mbps 이상의 고속망이 DCN으로 구성이 된다면 Farming 방법론에 의한 시스템의 개발이 바람직하다는 결론을 얻게 된다.

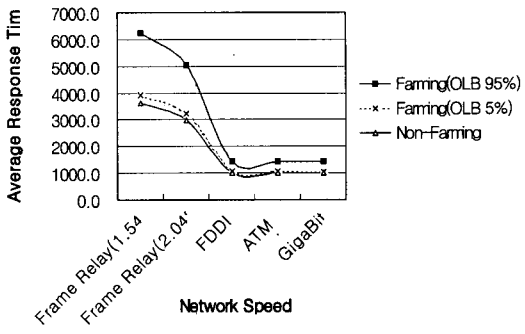


그림 8. 전체 트랜잭션 처리시의 평균 응답시간

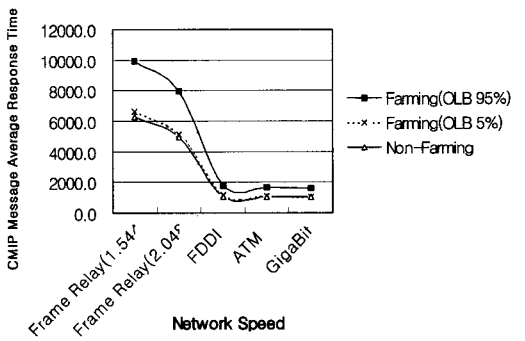


그림 9. CMIP 메시지 처리시의 평균 응답시간

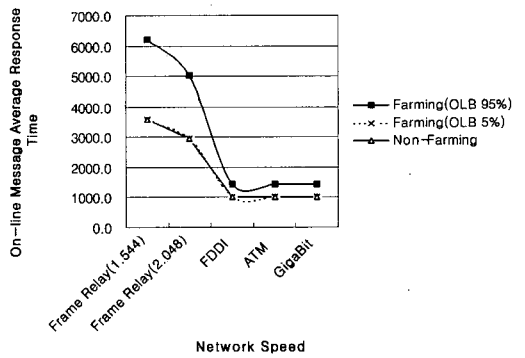


그림 10. 통지 메시지 처리시의 평균 응답시간

Ⅵ. 결 론

TMN을 비롯한 네트워크상의 분산객체 시스템의

구축 시 문제가 되는 사항은 관리하고자 하는 네트워크들의 주요 관리 시스템들이 서로 상이한 관리 체계를 가지고 있기 때문에 분산객체 내의 클래스들이 자신의 하드웨어나 시스템에 종속성(dependency)을 가지게 된다는 점이다. 이로 인하여 전체 망관리입장에서는 동일한 기능을 수행하는 클래스의 버전을 관리하기가 용이하지가 않고 플랫폼 종속성으로 인하여 다중플랫폼(multi-platform)의 지원이 불가능하게 되며 궁극적으로는 Q3 인터페이스의 구현에 대한 표준화에 도달할 수 없게 된다.

이와 같은 문제를 해결하기 위해 본 논문에서는 매니저와 에이전트 등 분산객체 내의 소프트웨어 컴포넌트들을 컴포넌트웨어형태로 생성하여 PICR에 유지시켜 필요시 이러한 컴포넌트웨어들을 분산객체로 동적 또는 정적으로 다운로딩하여 실행시키는 Farming 방법론을 제안하였다. 이 방법론은 TMN 시스템 내에서 에이전트를 구성시 주어진 역할을 수행하기 위하여 분산객체의 주요기능을 담당하는 블록들을 프레임워크 리스트로 정의하고 이 프레임워크 리스트를 구성하고 있는 소프트웨어 컴포넌트들을 PICR로부터 이식하여 분산객체 프레임워크를 구성하고 에이전트의 실행에 필요로 하는 데이터 컴포넌트까지도 이식하여 전체 에이전트를 구성하는 것을 의미한다.

Farming 방법론을 이용함으로서 네트워크내의 분산객체를 설계 및 구현함에 있어 Q3 인터페이스의 구현에 대한 표준을 제공할 수 있으며 컴포넌트웨어 대한 동적로딩 수행이 가능하게 된다. 또한 에이전트의 구축에 있어 플랫폼 독립성 유지할 수 있게 됨으로서 다중플랫폼(multi-platform) 지원이 가능하게 된다. 또한 신규 또는 변경된 컴포넌트웨어들을 PICR에 유지시킴으로서 네트워크 전체에 보다 효율적으로 포팅이 가능하게 되며 소프트웨어의 생명주기(lifecycle)의 적기에 적용이 가능하다. 나아가 에이전트 하드웨어 플랫폼으로 보다 경제적인 JAVA 스테이션과 같은 네트워크 컴퓨터 (NC : Network Computer)의 사용이 가능하게 되었다.

본 논문에서는 PICR과 Farming 방법론을 지원하기 위하여 Farmer 모델을 제안하였다. Farmer 모델은 개체의 수평/수직적인 측면을 다양하게 반영할 수 있을 뿐만 아니라 시스템을 분석접근시 개체와 측면간의 관계성식별에 중심을 두고 있으며 또한 측면의 개념을 반영한 측면객체(Aspect-Object)를 이용함으로서 측면지향 프로그래밍에서와 같은 통합자가 불필요하다는 장점이 있다. 나아가 객체의 통합시점도 측면객체를 정의시에 이루어지는 차이점이 있다. Farmer 모델을 이용하여 본 논문에서는 PCN (Personal Communication Network) TMN 에이전트를 설계 및 구현하였다. 구현 결과를 실제 환경에 맞추어 시뮬레이션 해본 결과 DCN을 구성하는 망이 ATM 이상의 고속망으로 발전시 Farming 방식과 non-Farming 방식사이에는 성능에 있어 큰 차이를 보이지 않게 됨을 알 수 있었다. 하지만 개발비용 면에서 본다면 하드웨어 플랫폼 측면이나 소프트웨어 개발비용 측면에서도 Farming 방법론이 비교 우위에 있게 된다. 따라서 100Mbps 이상의 고속망이 DCN으로 구성이 된다면 Farming 방법론에 의한 시스템의 개발이 바람직하다는 결론을 얻게 되었다.

참 고 문 헌

[1] ITU-T Recommendation M.3010, "Principles for a TMN", 1992.

[2] Salah Aidarous, Thomas Plevyak, "TMN into the 21st Century, Techniques, Standards, Technologies and Applications", *IEEE PRESS*, 1994.

[3] DSET Corporation Version 1.2, "GDMO Agent Toolkit User Guide", 1995.

[4] Soo-Hyun Park, Doo-Kwon Baik, Sang-Hoon Park, "TMN Distributed Object Design Methodology based on Platform Independent Class Repository", *In Proceedings of The 1997 International Technical Conference on Circuit/ Systems, Computers and Communications (ITC-CSCC'97)*, Volume I, pp.571 - 574, Okinawa, Japan, 1997.

[5] Soo-Hyun Park, Doo-Kwon Baik, Sang-Hoon Park, "Platform Independent TMN Componentware and Data Element Repository Based on Software Farming Methodology". *In Proceedings of The Sixth International Conference on Information Systems Development (ISD'97)*, pp.169 - 184, Boise, ID, USA, 1997.

[6] Soo-Hyun Park, Doo-Kwon Baik, Sang-Hoon Park, "Farming Methodology for TMN Platform Independent Class Repository Design", *In Proceedings of The 21st Annual Inter-*

national Computer Software & Applications Conference (COMPSAC'97), IEEE Computer Society, pp.352 - 355, Washington D.C, USA, 1997.

[7] Peter Coad, *Object Models : Strategies, Pattern, and Applications*, Prentice Hall, 1994.

[8] Alfons Kemper, Guido Moerkotte, *Object-oriented Database Management : Applications in Engineering and Computer Science*, Prentice Hall, 1994.

[9] Richard C. Lee, William M.Tepfenhart, *UML and C++ - A Practical Guide to Object-Oriented Development*, Prentice Hall, 1997.

[10] James Martin, James J. Odell, *Object Oriented Software Analysis & Design*, Prentice-Hall, 1992.

[11] Ann L.Winblad, Samuel D.Edwards and Davis R.King, *Object-Oriented Software,* Addison-Wesley, 1992.

[12] Gregor Kiczales, John Irwin, "Aspect-Oriented Programming, A Position Paper", *Xerox Parc*, 1995.

[13] Berlin, A., et al., "Distributed Information Systems for MEMS, ISAT Study", *ISAT*, 1995.

[14] Soo-Hyun Park, Doo-Kwon Baik, Sang-Hoon Park, "The TMN Agent in PCN based on Entity-Aspect Model", *In Proceeding of 1997 IEEE International Conference on Personal Wireless Communications (ICPWC'97)*, pp.394 - 398, Bombay, India, 1997.

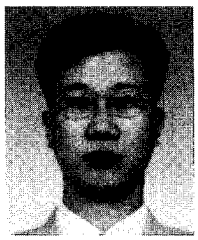
[15] Zeigler B.P, "Multifaceted Modeling and Discrete Event Simulation", Academic Press, 1984.

[16] ComponentWare Consortium, "A technical product description", *I-Kinetics, Inc.*,1995.

[17] ComponentWare Consortium Technical Plan Statement of Work, *I-Kinetics, Inc.*,1995.

[18] David Flanagan, "Java in a Nutshell, A Desktop Quick Reference for Java Programmers", *O'Reilly & Associates*, Inc. 1996.

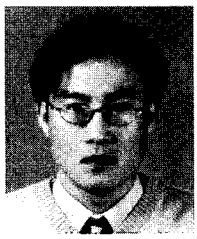
박 수 현(Soo-Hyun Park)



1988년 : 고려대학교 이과대학 컴퓨터학과 (학사)
1990년 : 고려대학교 대학원 수학과 (석사)
1998년 : 고려대학교 대학원 컴퓨터학과 (박사)

1990년 ~ 1999년 : LG정보통신(주) 중앙연구소 선임연구원
1999년 ~ 현재 : 동의대학교 공과대학 컴퓨터응용공학부 전임강사
<주관심 분야> 컴포넌트기반 개발, 컴포넌트 관리 시스템, TINA, 지능망
e-mail : shp@hyomin.dongeui.ac.kr

이 광 형(Kwang-Hyung Lee)



1988년 : 순천향대학교 공과대학 전산학과 (학사)
1990년 : 고려대학교 대학원 컴퓨터학과 (석사)
1995년 : 고려대학교 대학원 컴퓨터학과 (박사)

1995년 ~ 1996년 : 한국산업표준원 선임연구원
1997년 ~ 현재 : 동서대학교 정보시스템공학부 컴퓨터공학전공 전임강사
<주관심 분야> 분산시스템, 컴포넌트 기반개발, 컴포넌트 관리시스템