論文
3-7-4

컴포넌트 아웃소싱에 기반한 NMS 디자인 모델 (NMS Design Model based on Component Outsourcing)

이 광 형*
(Kwang-Hyung Lee)

요 약

다양한 통신 네트워크를 관리함에 따라 통합되고, 효율적인 통신 네트워크 오퍼레이션 및 유지 관리를 목적으로 한 TMN(Telecommunication Management Network)을 개발하였다. 그러나 TMN은 구현과정에서 서로 다른 플랫폼상에서 개발되어 왔기 때문에 TMN 시스템 에이전트의 클래스를 개발하고 유지하는 단계에서 여러 가지 문제점이 발견되었다. 이러한 문제점을 해결하기 위해서 본 논문에서는 CBD(Component Based Development.)를 기반으로 한 컴포넌트 아웃소싱 모델을 제안한다.

ABSTRACT

By managing various communication networks, we have Telecommunication Management Network(TMN) which has the aim of unified and effective communication network operation and maintenance. However, since TMN has been developed by different platforms in the implementation process, several problems have been found in the step of developing and maintaining the class of TMN system agent. In order to solve these problems, this paper suggests the component outsourcing concept which is based on Component Based Development.

1. 서론

컴포넌트 기반 개발 (CBD : Component Based Development)은 컴포넌트의 생성, 선택, 평가 및 통합으로 구성되는 소프트웨어 개발 방법의 새로운 패러다임이다. 컴포넌트 기반 개발은 컴포넌트의 동작(operation)과 상호동작(interoperability)을 정의하는 명세의 개발, 객체나 컴포넌트들로부터 컴포넌트의 구축, 컴포넌트들을 이용해 응용프로그램(application)을 조립하는 등의 활동을 필요로 한다고 정의하고 있다.[1]

TMN[3][4][5]시스템 개발 방법론인 Farming 방법론[2][6]의 기본 모델인 Farmer 모델은 [7][8] 가 시뮬레이션 및 인공지능 모델로 제안한 시스템 개체구조(System Entity Structure)[7]의 기본개념과 구성원리에 기본을 두고 있다. 본 논문은 ATM, FDDI 등 고속망을 기본망으로 사용하는 TMN 망관리 시스템[9][10][11]의 에이전트[9][10][11][12] 등과 같은 분산객체를 디자인하고 구현하기 위한 새로운 방법론인 Farming 방법론내의 컴포넌트 아웃소싱(Outsourcing)에 대하여 연구하였다. Farming 방법론은 망관리 시스템내의 에이전트를 구성하고 있는 소프트웨어 블록을 플랫폼 독립적인 컴포넌트 형태로 변

* 정회원 : 코딕커뮤니케이션즈 책임연구원

논문접수 : 2002. 6. 11.
심사완료 : 2002. 6. 28.

형한 후 기본망 내에 위치하는 플랫폼 독립형 클래스 저장소(PICR : Platform Independent Class Repository)로 이동시킨 후 분산객체 시스템을 구축하거나 기능을 실행 시에 필요한 컴포넌트들을 PICR로부터 아웃소싱하여 사용하는 것이다. Farming 방법론은 Farmer 모델 형식론(formalism)에 기본을 두고 있다.

2. CBD(Component Based Develop ment)에 기반한 TMN 시스템

2.1 TMN 시스템 분산객체 내 클래스 관리의 문제점

공중전화통신망(PSTN : Public Switching Telephony Network), 공중데이타통신망(PSPDN : Packet Switched Public Data Network), 지능망(IN : Intelligent Network), 개인휴대통신망(PCN : Personal Communication Network) 등의 모든 통신망을 총괄적으로 관리함으로서 일원화되고 효율적인 통신망 운용관리를 지향하는 개념으로 출현한 TMN은 구축 과정에서 여러 가지 서로 다른 상용 플랫폼 툴킷과 이기종의 하드웨어 플랫폼 및 운영체제 등을 이용하여 개발하는 관계로 TMN 시스템의 클래스 관리는 다음과 같은 여러 문제점을 내포하게 된다.

첫째, 다수의 에이전트 및 매니저 등에서 사용하는 분산객체들은 동일한 기능을 수행하는 클래스들을 중복하여 모두 유지해야만 한다. 예를 들어 각 에이전트들로 하여금 신규 관리객체(MO)[10][11][12]들을 생성/삭제하기 위해서 매니저는 M_CREATE나 M_DELETE의 CMIP(Common Management Information Protocol) 메시지를 에이전트로 전송하는데 이러한 CMIP 메시지[11][12]를 처리하기 위한 기능을 수행하는 클래스들을 모든 에이전트들이 유지해야 한다. 이로 인해 동일한 클래스의 버전을 관리하기란 용이한 사항이 아니다.

둘째, TMN 구축시 중대한 문제점으로 관리하고자 하는 망들의 주요 관리 시스템들이 서로 상이한 관리체계를 가지고 있다는 것이다. 즉, 각 망의 에이전트 기능을 담당하는 시스템들이 하드웨어나 운영체제 등의 측면에서 현실적으로 서로 상이하다는 점이다. 어떤 망의 에이전트 플랫폼은 UNIX 머신을 기반으로 하는 시스템일 수 있고 다른 망의 경우에는 윈도우 운영체제로 사용하는 PC를 근간으로 하는 시스템일 수도 있다. 이에 따라 동일한 기능을 수행하는 에이전트 내의 클래스들이 자신의 컴퓨터나 시스템에 종속성(dependency)을 가지게 된다. 이러한 하드웨어 및 운영체제 플랫폼 종속성을 줄이는 것이 매우 중대한 문제이다.

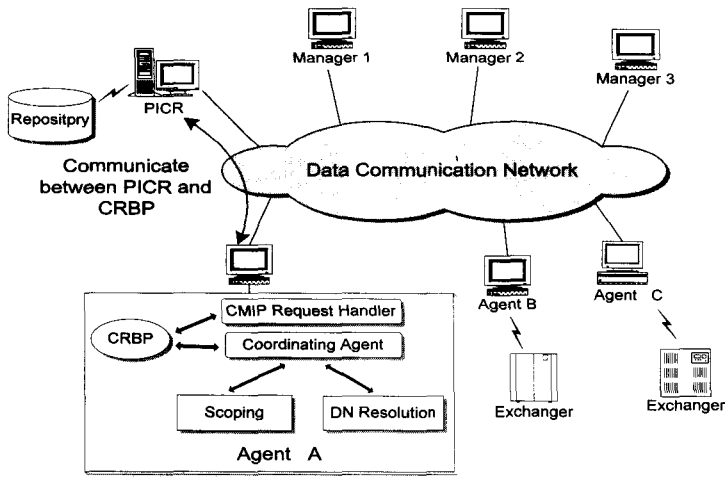
셋째, 다중플랫폼(multi-platform)이 지원되지 않는다. 동일한 기능을 수행하는 에이전트 내의 클래스들이 자신의 하드웨어나 시스템에 종속성을 가지게 됨으로서 다중플랫폼의 지원이 불가능한 현실이다.

넷째, Q3 인터페이스의 구현에 대한 표준화가 이루어지지 않고 있다. 다중플랫폼을 지원하지 않는 관계로 DCN(Data Communication Network)과 OS (Operation System), MD(Mediation Device), NE (Network Element) 등과의 Q3 인터페이스[10][11][12]를 구현시 일관된 인터페이스를 제공하기 어렵다. 현재 서로 다른 망은 각각의 망관리시스템을 유지하고 있기 때문에 집중화된 망관리 시스템인 TMN을 구축 시 이와 같은 서로 다른 망의 유지보수 시스템 사이의 호환성이 보장되지 않는다.

2.2 컴포넌트 저장소(PICR)의 정의 및 구성

분산객체형 네트워크 컴퓨팅 개념을 이용하여 클라이언트(매니저와 에이전트)/서버(PICR : Platform Independent Class Repository)를 구축하기 위해서 본 연구는 라이트사이징(rightsizing)의 방식을 근간으로 하고 있다. 이는 중대형 컴퓨터 등에 영향을 받지 않고 다양한 플랫폼에 가장 적합한 응용프로그램을 구성하는 것을 의미한다.

ATM, FDDI 등 고속망을 기반으로 하는 분산객체 네트워크 컴퓨팅 개념 하에서 TMN 구성요소인 각 분산객체들이 갖는 클래스들을 컴포넌트웨어 형태로 생성하여 이들만을 별도로 관리하는 임의의 서버에 유지시킨다. 즉, 매니저와 에이전트들이 실행하는 기능 (예 : M_CREATE, M_DELETE 등의 CMIP 메시지를 처리하는 기능)들은 매니저와 에이전트 등 분산객체에서 클래스들만을 별도로 분리하여 이를 관리저장하는 임의의 서버로 옮긴다. 이러한 서버를



PICR : Platform Independent Class Repository
CRBP : Class Repository Broking Processor
CMIP : Common Management Information Protocol
DN : Distinguished Name

[그림 1] 클래스 저장소를 갖는 분산객체형 TMN 구성 모델
<figure 1> Distributed Object Type TMN compositin Model with Class Repository

클래스 저장소(Class Repository)라 정의하며 여기에 저장되어 유지되는 클래스들은 TMN 내의 매니저 및 에이전트를 구성하는 하드웨어나 운영체제 등과는 무관하게 독립적으로 유지할 수 있기 때문에 플랫폼독립형 클래스저장소 (PICR : Platform Independent Class Repository)라 부른다. 이와 같은 PICR를 갖는 분산객체형 TMN 구성모델은 [그림 1]과 같다.

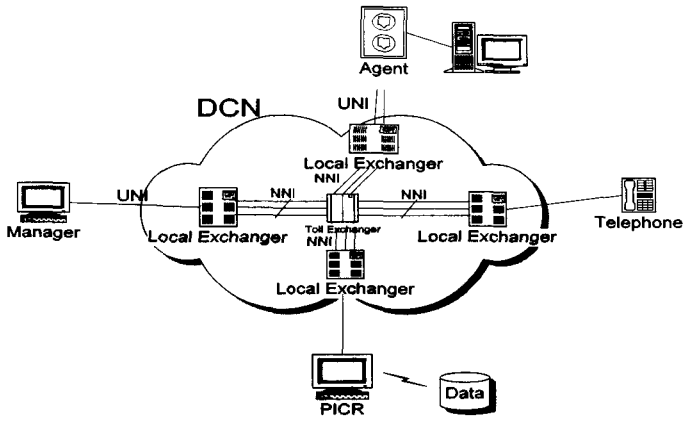
이에 따라 각 에이전트는 매니저로부터 온 CMIP 메시지를 처리하기 위해 PICR에 접근하여 해당 CMIP 메시지를 처리할 수 있는 기능의 컴포넌트웨어를 자신의 하드웨어 플랫폼으로 임포트하여 해당 기능을 실행하는 것이 가능하다.

[그림 1] 내의 CRBP는 분산객체 내에서 다른 분산객체로 부터의 요구를 수행하기 위해 PICR로부터 해당 기능의 클래스를 임포트하는 실질적 역할을 담당하는 프로세서이다. 예를 들어, 신규 관리객체를 생성할 목적으로 매니저로부터 전송된 M_CREATE CMIP 메시지를 처리하기 위해 에이전트는 CRBP에 해당 클래스를 PICR 로부터 가져와 줄 것을 요구한다. CRBP는 자신이 속한 에이전트에서 최적의 루팅(routing)으로 도달가능한 PICR의 주소를 결정(address resolution)한다. 이 주소의 PICR로 원하는 기능의 클래스를 자신의 에이전트에게 보내어 줄 것을 요구한 후, 해당 클래스를 다운로딩 받게되면 이를 관련 기능 블록으로 전송하여 준다.

실제로 교환 시스템에서 PICR를 갖는 TMN 시스템을 구성한다면 [그림 2]와 같이된다. 데이터 통신 네트워크(DCN)는 Local Exchanger, Toll Exchanger 등의 교환 시스템으로 구성이 된다.

Farming 방법론에서 사용하는 PICR내에 저장되는 컴포넌트는 일반적으로 다음과 같이 정의를 내릴 수 있다.

"*컴포넌트란 플랫폼에 독립적으로 생성과 유지보수를 보다 용이하게 수행할 수 있는 일종의 작은 소프트웨어 조각으로서 상호간의 표준화 된 인터페이스를 갖는다.*"[13][14]



UNI : User Network Interface
NNI : Network Node Interface
DCN : Data Communication Network

[그림 2] 교환시스템 내에서의 클래스 저장소를 갖는 TMN 모델
<figure 2> TMN Model with Class Repository in the Exchange System

컴포넌트는 컴퓨터 언어에 중립적인 클라이언트/서버 상호작용 모델에 기반하여 상호동작(interoperation)을 수행한다. 이제까지의 객체와는 달리 컴포넌트는 컴퓨터언어 패키지 및 운영체제와 무관하게 네트워크내에서 상호작용을 수행할 수 있다. 그러나 컴포넌트는 상속성, 폴리모피즘 그리고 인캡슐레이션을 지원한다는 점에서 객체와 유사한 특성을 가지고 있다. Ivar Jacobson [15]은 컴포넌트를 블랙박스의 속성을 갖는 객체로 보았다. 블랙박스 컴포넌트의 경우는 상속성을 통한 컴포넌트의 확장이 불가능하다. OLE 컴포넌트는 바로 이러한 블랙박스의 범주에 속하나 CORBA[16] 및 OpenDoc에서 사용하는 컴포넌트들은 상속성 개념을 지원한다. 화이트박스 컴포넌트는 고전적인 개념의 객체와 같이 행동하는 컴포넌트를 의미한다. 이와 같이 여러 정의가 존재하는 관계로 여기서는 컴포넌트가 지원해야하는 가장 최소의 기능 및 특성에 대하여 서술하기로 한다.

- 개방된 시장에서 구매가 가능해야 한다.
- 완벽한 응용 프로그램이 아니다.
  컴포넌트는 다른 컴포넌트와 연동하여 완벽한 응용프로그램을 형성한다. 따라서 컴포넌트는 응용프로그램 영역내의 한정된 역할을 수행한다. 컴포넌트는 다음과 같은 3가지 종류로 구분할 수 있다.
  - Fine-grained object : 간단한 C++ 클래스 등
  - Medium-grained object : GUI 컨트롤 등
  - Coarse-grained object : applet 등
- 어떠한 조합으로도 사용할 수 있어야 한다.
  컴포넌트는 P&P(Plug & Play) 개념에 의하여 다른 어떠한 컴포넌트와 조합을 이루어 동일한 가계(family)를 이루어야 한다.(이를 수트(suite)라 함).
- 컴포넌트는 잘 서술된(well-specified) 인터페이스를 가져야 한다.
  고전적 개념의 객체와 같이 컴포넌트는 자신이 갖는 인터페이스를 통하여 통제를 할 수 있다. CORBA/OpenDoc의 IDL (Interface Definition Language)이 이에 해당한다.
- 컴포넌트는 상호호환가능한 객체이다.
  컴포넌트는 주소공간, 네트워크, 컴퓨터 언어, 운영체제, 그리고 툴의 한계를 뛰어넘어 상호호환하여 사용할 수 있는 시스템 독립적인 개체로 볼 수 있다.

Farming 방법론에서는 컴포넌트웨어 형태로 변형된 컴포넌트요소들에 대하여 Farmer 모델에서는 다음과 같은 정의를 내린다.

[정의 2.1] 컴포넌트 요소

컴포넌트요소의 집합 C는 다음과 같은 structure로 정의된다.

$\forall c \in C,\ c = <$ Cid, G, SZ, L_AT $>$

where,

Cid : 컴포넌트의 이름

G : 컴포넌트가 속해있는 PICR내의 그룹 이름

SZ : 컴포넌트의 크기(화일 크기 (단위 : byte))

L_AT : 컴포넌트의 로딩과 관련된 속성집합

L_AT = { DYNAMIC, STATIC, DYNAMIC-STATIC } ■

2.3 PICR의 주요기능 및 특징

분산객체 네트워크 컴퓨팅 기반 TMN PICR를 구성 시 PICR에 해당하는 서버 부분과 이 클래스의 실제적 사용자인 클라이언트에 해당하는 분산객체 부분을 완전히 분리하고 이들 사이의 연결기능을 제공하는 CRBP(Class Repository Broking Processor)를 미들웨어(middleware) 개념으로 각 분산객체에 위치시킴으로서 유연하고 확장가능(extensible)한 시스템을 구축할 수 있다.

궁극적으로 네트워크 컴퓨터를 분산객체의 기본 하드웨어 플랫폼으로 사용한다면 CRBP 기능을 하는 클래스 자체도 클래스저장소에 위치시켜야 할 것이며 분산객체 개념을 기반으로 클래스저장소의 구축이 이루어짐에 따라 컴포넌트웨어의 형태로 구성되는 클래스는 재사용이 가능하다.

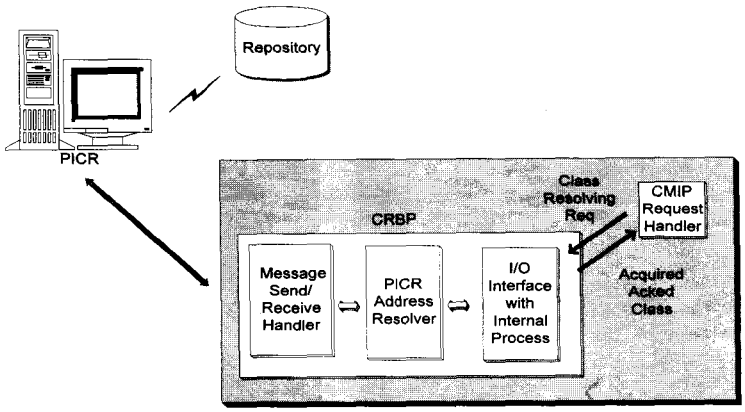
2.3.1 CRBP 주요기능

앞의 [그림 1]에서 분산객체 내에 존재하는 CRBP의 주요 기능은 다음과 같다.

첫째, 클래스저장소 서버의 주소를 결정(address resolution)한다. 다른 분산객체로부터 전송되어 온 요구에 맞는 기능을 수행하는 클래스를 찾기 위해 최단 루팅으로 찾아 갈 수 있는 PICR 서버의 주소를 결정하는 기능이다.

둘째, 클래스저장소 서버로 클래스 전송을 요구하고 해당 응답을 수신한다.

셋째, PICR과 분산객체 매니저/에이전트 사이의 전송 및 수신시의 에러를 예외처리(exception handling)를 이용하여 처리한다.



PICR : Platform Independent Class Repository

CRBP : Class Repository Broking Processor

[그림 3] CRBP의 주요기능

<figure 3> Principle function of the CRBP

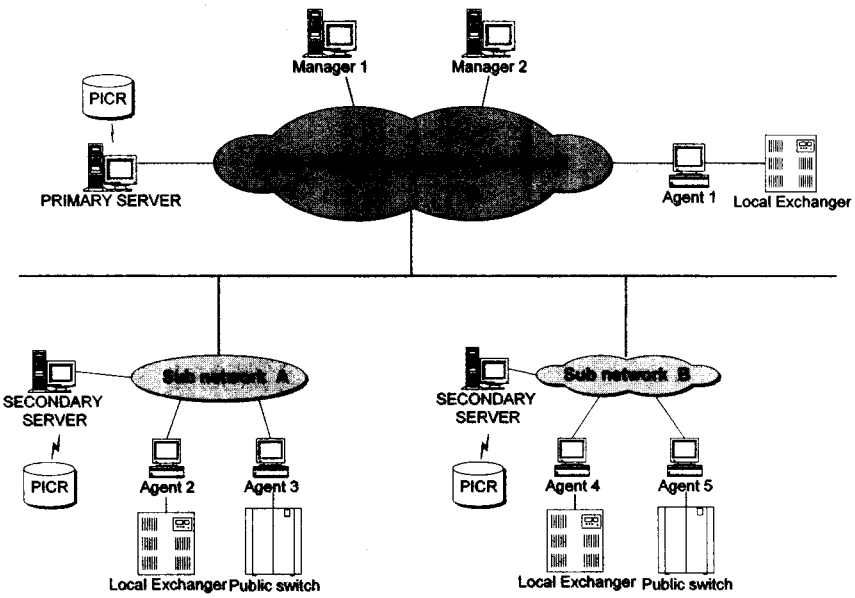
넷째, 최종적으로 PICR로부터 수신한 클래스를 관리객체 내의 해당 관련 블록으로 전송해 준다.

[그림 3]은 이러한 기능을 도식적으로 보여주고 있다.

2.3.2 PICR 특징

TMN 시스템을 구축할 때, PICR을 도입할 경우 다음과 같은 장점을 얻을 수 있다.

1) Q3 인터페이스의 구현에 대한 표준 제공
분산객체형 네트워크 컴퓨팅 개념의 도입으로 PICR에 저장되어 있는 클래스를 여러 OS (Operation System)가 접근하여 이용함으로서 Q3 인터페이스의 구현에 대한 표준이 이루어 질 수 있다.
2) 동적 로딩(dynamic loading) 수행
시스템 운용(run-time) 시에 PICR에 있는 클래스를 동적으로 로딩할 수 있다.
3) 플랫폼 독립성 유지 및 다중플랫폼(multi-platform) 지원
에이전트 생성 툴킷, 하드웨어 및 운영체제 등의 플랫폼과 독립성을 유지할 수 있기 때문에 클래스가 분산객체의 하드웨어나 운영체제 등의 플랫폼에 독립성을 가지게 됨으로서 다중플랫폼을 지원할 수 있게 된다.
4) 포팅(porting)이 용이
PICR에서 클래스를 다루는 사항은 클래스를 단지 PICR에 적재해 주기만 하는 문제로 모아진다. 신규 또는 변경된 클래스를 PICR에 적재한 후 매니저와 에이전트인 분산객체들에게 이러한 변경 사항을 알려주어야 하며 (notification/broadcast) 이에 관련된 메시지의 정의가 필요하다. 이를 수행하기 위해 여러 PICR중의 하나를 [그림 4]와 같이 주 서버 (primary server)로 지정하여 망 관리자는 주 서버에만 클래스를 변경/삭제/신규 등록하면 된다. 이러한 변동 사항은 주 서버가 아닌 다른 부 서버(secondary server)로 변경 내역이 전송되고 이러한 변경 사항에 따라 변경된 클래스를 필요로 하는 부 서버는 해당 클래스를 다운로딩 받아 간다. 이 경우 주 서버에 해당



[그림 4] DCN내에서의 PICR의 위치

<figure 4> Position of the PICR in the DCN

하는 PICR의 주소를 찾아내야 하는 주소해결 (address resolution) 방안이 필요하다.

3. NMS 디자인 모델 - Farmer 모델

Farmer 모델은 실세계의 개체를 각각의 고정된 측면이 아닌 여러 관점의 측면에서 분석한 후 분석 완료된 측면요소들을 측면객체로 정의한다. 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 leaf 노드에 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. 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 3가지 추상화 개념들로 구성된다.

Farmer 모델의 형식구조는 다음과 같이 정의된다.

[정의 3.1] Farmer 모델 형식구조

Farmer 시멘틱 structure 는 ordered sequence로 정의된다[9].

Farmer 모델 형식구조 F는 다음과 같이 정의된다.

$$\mathbf{F = < C,\ AO,\ E,\ S,\ \lambda_1,\ \lambda_2,\ \lambda_3,\ CNST >}$$

where, C : 컴포넌트 집합
AO : 측면객체 집합
E : 개체 집합
S : 측면구조들의 집합
측면객체 AO의 측면을 $s_1, s_2, \ldots, s_n$이라 하면, AO의 측면집합 S는 다음과 같이 표기된다.

$$S = \bigcup_{i=1}^{n} \{ s_i \}$$

λ_1 : $C \rightarrow AO$,
$\lambda_1(c)$ = **Aspect_Object (c)**,

λ_2 : $E \cup S \rightarrow AO$,
for $\forall h \in E \cup S$,
$\lambda_2(h)$ = **Aspect_Object (h)**

λ_3 : $C \rightarrow AO$
for $\exists c \in C$ in PICR
$\lambda 3(c)$ = moveto(**AO** in an Agent_platform)

CNST : Farmer 모델구조에서 사용하는 상수 (constant)들의 집합 ■

4. 컴포넌트 아웃소싱 알고리즘

분석대상 시스템이 수행하는 메소드들이 컴포넌트웨어 형태로 변형된 컴포넌트들의 집합 C가 PICR 형태로 존재하고 임의 분석대상시스템인 AGENT가 다음과 같다고 가정하자.

- 시스템명 : Agent_id
- AGENT를 분석하는 측면 : Agent_aspecti
- AGENT가 가지는 속성 : Agent_attributej
- AGENT내에서 수행되는 오퍼레이션 : Agent_methodk
- AGENT의 상위클래스 : Agent_upper_class
- AGENT의 하위클래스 : Agent_lower_classh (단, i, j, k, h는 자연수)

Farmer 모델형식 구조에 따른 컴포넌트 아웃소싱 알고리즘은 다음과 같다.

1. 분산객체 플랫폼으로 소프트웨어 컴포넌트 및 데이터 컴포넌트를 이식하기 위한 환경을 조성한다.

1.1 개체구조집합 E에 대하여
∀e ∈ **E**, e.Eid ← init_value for Eid
e의 속성집합 A에 대하여
∀a ∈ **A**, a.Aid ← init_value for Aid
a.**AT** ← init_value for **AT**
e.**S** ← none
e.**LT** ← none < ① >

1.2 측면구조집합 S에 대하여
∀s ∈ **S**, s.ASPid ← null
OWNER(s) ← null set < ② >

1.3 메소드 도메인 구조들의 집합 **M**에 대하여
∀m ∈ **M**, m.S ← null set /* ②에 의하여 */
m.Mid ← init_value for Mid
m.**T** ← init_value for **T** < ③ >

1.4 측면객체구조들의 집합 **AO**에 대하여
∀o ∈ **AO**,
o = < initial aspect object id,
null aspect set, /* ① */
null attribute set, /* ② */
null data,
null method set, /* ③ */
null super set,
null sub set >

2. 분산객체 시스템 기능분석 및 컴포넌트웨어 요소 추출의 단계에서 디자인된 프레임워크 리스트에 따라 개체구조집합, 균일성개체, 측면구조집합 및 균일성측면을 생성한다.

2.1 개체구조 집합 **E**의 원소를 e1, e2, ... ,en 이라 할 때 (n은 자연수)
for ∀e ∈ **E**,
e_1.Eid ← root class agent id (Agent_id)
e_1내의 속성집합 A에 대하여
for ∀a ∈ **A**,
a.Aid ← root class agent의 속성 (Agent_attributej)
a.AT ← root class agent의 속성타입
e_1.**S** ← none
e_1.**LT** ← none

2.2 $\mathbf{E_1}$ = { e_1 }
$\mathbf{E_{sub}} = \bigcup_{i=2}^{n} \{ ei \}$ 라 정의할 때,
$\mathbf{E} = \mathbf{E_1} \cup \mathbf{E_{sub}}$가 되는 ∀e ∈ $\mathbf{E_{sub}}$에 대하여
FOR i = 2 **TO** h
ei.Eid ← Agent_lower_classi의 id
IF for ∃e ∈ **E**, e.Eid = ei.Eid
THEN
/* 균일성 개체의 경우 */
Copy all attributes of e and subtree structure to ei
END IF
ei 내의 속성집합 **A**에 대하여,
∀a ∈ **A**에 대하여
a.Aid ← Agent_lower_classi의 속성
a.AT ← Agent_lower_classi의 속성타입
e_i.S ← Agent_lower_class$_i$의 측면집합
e_i.LT ← Agent_lower_class$_i$의 loading 타입
END FOR

3. 측면구조집합 **S**의 원소를 s_1, s_2, ... ,s_m이라 할 때 (m은 자연수)
$\mathbf{S} = \bigcup_{i=1}^{n} \{s_i\}$와 같이 정의할 수 있음.
∀s ∈ S에 대하여
FOR i = 1 **TO** m (m은 자연수)
IF for ∃s ∈ **S**, s. ASPid = s_i.ASPid and **OWNER**(s) = OWNER(s_i)
THEN /* 균일성 측면의 경우 */
Copy all attributes of s and subtree structure to s_i
END IF
s_i.ASPid ← Agent_aspect$_i$
OWNER(s_i) ← s_1을 측면으로 하는 AOid
END FOR

4. 측면객체를 정의하며 측면객체내의 메소드들이 ILB(Initial-Loading function Block)[5]인지 OLB(On-demand Loading function Block)[5]인지의 여부를 확인한다.

4.1 측면객체들의 구조집합을 **AO**라 할 때

$\forall o \in$ **AO**, o.AOid ← Agent_id

4.2 **AO** 내의 측면구조집합 **S**의 원소를 s_1, s_2, ..., s_n이라 할 때 (n은 자연수)

$\mathbf{S} = \bigcup_{i=1}^{n} \{s_i\}$, $\forall s_i \in$ **S**에 대하여

FOR i = 1 **TO** n

for $\forall s_i \in$ **S**,

s_i.ASPid ← Agent_aspect$_i$의 id

OWNER(s_i) ← s_i를 측면으로 하는 AOid

END FOR

AO 내의 속성집합 **A**에 대하여

FOR j = 1 **TO** m

for $\forall a \in$ **A**,

a_j ← Agent_attribute$_i$

END FOR

4.3 **AO** 내의 메소드 도메인 구조집합 **M**과 **M** 내의 측면구조들의 집합 **S**, **M**내의 메소드 타입 집합 **T**에 대하여

for $\forall o \in$ **AO**, $\forall m \in$ **M**, $\forall s \in$ **S**,

m_k.Mid ← Agent_method$_k$

m.s_i.ASPid ← Agent_aspect$_i$

m.s.**OWNER**(s) ← s를 측면으로 갖는 **AO** 집합

$\forall t \in$ **T**,

IF Size(m) $\geq \mu$ **and**

Load_frequency(b) $\geq \zeta$ **and**

Degree_of_importance(b) $\geq \delta$

THEN

/* m이 ILB인 경우 */

t ← s /* static loading mode */

ELSE /* dynamic loading mode */

t ← d

END IF

o.p ← Agent_upper_class

4.4 AO가 갖는 하위클래스 집합을 **B**라 할 때,

for $\forall b \in$ B, b_h ← Agent_lower_class$_h$

5. 개체구조집합 **E**, 측면구조집합 **S**, 측면구조집합 **AO**에 대하여

$\exists e \in$ **E**, $\exists s \in$ **S**, $\exists o \in$ **AO**,

λ_2(e $\vee$ s) → o

6. 4, 5에서 정의된 측면객체 구조집합 **AO**와 **AO** 내의 메소드 도메인 구조들의 집합 **M** 및 **M** 내의 메소드 타입집합 **T**에 대하여

$\exists o \in$ **AO**, $\exists m \in$ **M**, t를 **M**이 가지는 메소드 타입이라고 할 때

read(o.m.t) /* o.m.t를 읽어냄 */

7. IF o.m.t = d **THEN**

/* dynamic loading block 인 경우 */

for $\exists c \in$ **C** in PICR,

λ_3(c) → moveto(**AO** in an Agent_platform)

ELSE /* static loading block 인 경우 */

search local library

IF o.m 이 local library에 존재 = **O.K.**

THEN

execute the method

ELSE

/* CRBP가 middleware 역할수행/

send components_load_request_signal to PICR

receive the requested components from PICR execute the method

END IF

END IF

5. 결론

컴포넌트 기반 개발 (CBD : Component Based Development)에 기반하여 망관리 시스템을 설계 시 필요한 컴포넌트의 생성, 선택, 평가 및 통합 등의 여러 과정 중 본 논문에서는 컴포넌트 저장소(PICR)

에 저장된 컴포넌트의 선택 및 실제 분산객체로의 아웃소싱과 관련된 알고리즘에 대하여 설명하였다. 본 알고리즘에 대한 적합성(correctness)에 대한 증명은 매우 간단(trivial)하다.

컴포넌트 아웃소싱(Outsourcing)은 본 논문에서 Farming 이라고 부르고 있으며 Farming은 데이터 및 컴포넌트 마이닝과는 반대의 개념을 지니게 되므로 단순한 아웃소싱의 개념과는 차이가 있다.

향후 연구해야 할 부분으로는 프레임워크 개념에 의한 컴포넌트의 통합 및 컴포넌트간의 인터페이스 정의 등이 있다.

※ 참고문헌

[1] 권오천, 신규상, 전진옥, "컴포넌트 기반 개발 기술 검토 및 동향", 정보통신 기술, 정책 및 산업 주간기술동향, 제 900 호, 한국전자통신연구원, 1999

[2] 이광형, 박수현, "분산시스템상에서 Farmer Model을 이용한 통신망 관리 에이전트 개발", 한국정보처리학회 논문지 제6권 제9호, 1999

[3] ITU-T Recommendation M.3010, "Princi ples for a TMN", 1992.

[4] ITU-T Recommendation M.3020, "TMN Interface Specification Methodology", 1992.

[5] ITU-T Recommendation M.3100, "Gene ric Network Information Model", 1992.

[6] Soo-Hyun Park, Doo-Kwon Baik, "Plat form Independent TMN Agents Based on the Farming Methodology", The IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, The Institute of Electronics, Information and Communication Engineers (IEICE), pp. 1152 - 1163, Japan, 1998

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

[8] In-Kee Jeong and Doo-Kwon Baik, "Data Reengineering Methodology using ESR Model", Ph.d Thesis, Korea University, 1996.

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

[10] ISO CMIP Reference Guide, Retix, 1996.

[11] ISO CMIP Programmer Guide, Retix

[12] OSI Management Toolkit Program mer Reference Guide, Retix

[13] Robert Orfali, Dan Harkey, and Jeri Edwards, *The Essential Distributed Objects, Survival Guide*, John Wiley & Sons, Canada, 1996.

[14] Robert Orfali and Dan Harkey, *Client/Server Programming with JAVA and CORBA*, John Wiley & Sons, Canada, 1996.

[15] Ivar Jacobson, *Object-Oriented Software Engineering, A Use Case Driven Approach*, Addison-Wesley, 1992.

[16] Thomas J. Mowbray and Ron Zahavi, *The Essential CORBA Systems Integration using Distributed Objects*, OMG, 1995.

이 광 형

1999.5 고려대학교 컴퓨터학과 박사
1996.2 - 1997.2 한국산업표준원 선임연구원
1997.3 - 2000.2 동서대학교 컴퓨터학과 교수
2000.3 - 2002.3 (주)ECO 시스템개발실 부장
2002.4 - 현재 (주)코딕커뮤니케이션즈 개발실 책임연구원