COM IDL 인터페이스 정의를 위한 Farmer 모델 변형 알고리즘

박 수 현°, 민 성 기, 김 태 석

* 동의대학교 컴퓨터응용공학부

The Farmer Model Transformation Algorithm for the Definition of COM IDL Interfaces

Soo-Hyun Park°, Sung-Gi Min, Tai-Suk Kim

* Faculty of Computer Application Engineering, Dongeui University

요 약

Farming 이란 Farmer 모델에서 제안하는 컴포넌트 아웃소싱을 의미한다. Farming 개념을 구현하기 위하여 본 논문에서는 컴포넌트 기반 개발(Component-Based Development) 개념을 도입하였다. 컴포넌트는 컴포지션에 의한 소프트웨어 블록 재사용성을 제공하며 컴포넌트는 이들의 수행하는 기능에 대한 명세를 나타내는 인터페이스 명세 모델이 반드시 필요하게 된다. 본 논문에서는 Farmer 모델링 방법론에 의하여 디자인되어진 시스템을 컴포넌트 기반개발의 인터페이스 명세모델로의 변형(transformation) 메카니즘인 FTI 알고리즘 및 COM IDL과의 상호관련성에 대하여 소개한다.

I. 서론

소프트웨어 위기를 극복하고자 도입된 객체지향 프로그래밍은 소프트웨어 재사용이라는 개념을 제안하였으나 상속성에 의한 white-box reuse는 소프트웨어 재사용이라는 본질에는 충실하였을 지라도 최종 사용자 입장에서의 궁극적인 소프트웨어 재사용을 지원하지는 못하였다.[1][2][3] 하지만 컴포지션(composition)에 의한 소프트웨어 재사용 개념인 컴포넌트를 도입함으로서 이와 같은 문제점을 해결할 수 있게 되었다.[4][5][6] Farmer 모델[7][8]은 매우 복잡한 구성을 갖는 Tangled-mess-of-object 형태의 실세계의 사물을 그들이 갖는 여러 속성별로 다중측면[7]에 의하여 분석한 후 Farmer 모델 디자인 스퀘에 의하여 설계한 후 ILB 및 OLB 등의 컴포넌트 Farming 개념에 의하여 실제로 구현하는 개념을 갖는다.[8][9] Farming 개념을 구현하기 위하여 본 논문에서는 컴포넌트 기반 개발(Component-Based Development) 개념[4][10]을 도입하였다. 컴포넌트는 컴포지션에 의한 소프트웨어 블록 재사용성을 제공하며 컴포넌트는 이들의 수행하는 기능에 대한 명세를 나타내는 인터페이스 명세 모델이 반드시 필요하게 된다[9.] 본 논문에서는 Farmer 모델링 방법론에 의하여 디자인되어진 시스템을 컴포넌트 기반개발의 인터페이스 명세모델로의 변형(transformation) 메카니즘인 FTI 알고리즘을 제안한다. 또한 CORBA IDL[6]과의 상호관련성에 대하여 예를 들어 설명하고 있다.

II. COM IDL 인터페이스 정의를 위한 Farmer 모델 변형 알고리즘

Farmer 모델은 Tangled-mess-of-object 형태의 실세계의 사물을 속성별로 다중측면에 의하여 분석한 후 컴포넌트 Farming 개념에 의하여 디자인하는 개념을 설명하고 있다. 그림 1은 이와 같은 Farmer 모델의 문제분석

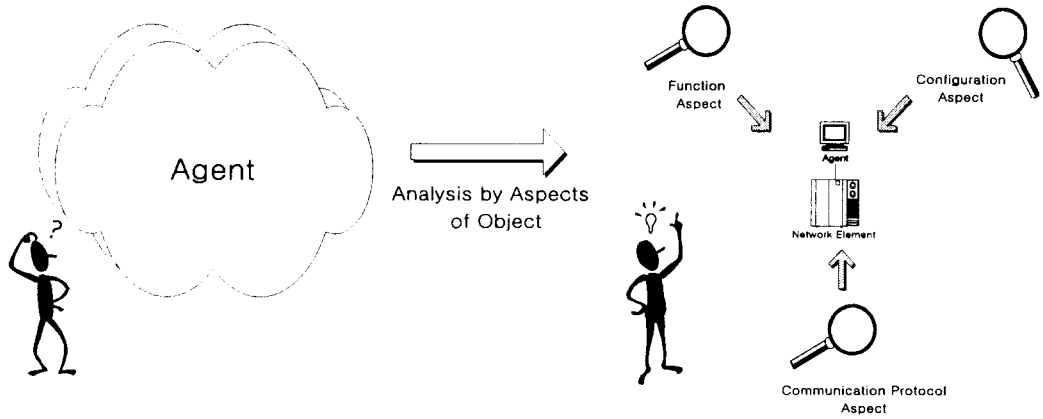

그림 1 Farmer 모델의 문제분석(problem-decomposition) 개념

(problem-decomposition) 개념을 보여주고 있다. 본 절에서는 Farmer 모델에 의하여 정의된 개념을 인터페이스 명세모델에서의 개념으로 변형시키는 방법 및 알고리즘에 대하여 설명하고 있다.

표 1은 이러한 양쪽 모델의 개념상의 매핑관계를 보여주고 있다. Farmer모델에서의 측면노드의 개념을 지원하기 위하여 측면 인터페이스(Aspect Interface)의 개념을 두었으며 Aspect Interface에 대한 상세정보를 보관하기 위하여 측면 인터페이스 카탈로그(Aspect Interface Catalog)를 별도로 유지하고 있다. 균일성 개체노드(uniformity entity node)는 인터페이스 카탈로그내의 동일한 이름을 갖는 인터페이스로 매핑이 되고 균일성 측면개체노드는 측면 인터페이스 카탈로그내의 동일한 이름을 갖는 측면 인터페이스로 매핑이 이루어진다. Farmer 모델에서의 다중 추상화(multiplicity abstraction)개념에 의한 대표개체노드(representative entity node)는 대표 인터페이스(Representative interface)로 변형되며 ILB 컴포넌트는 정적인터페이스 호출(Static Interface Invocation)을 지원하는 인터페이스로 매핑되는 데 속성 중 Type_Of_BasicComponent값을 ILB로 정한다. OLB 컴포넌트의 경우 동적인터페이스 호출(DII : Dynamic Interface Invocation)을 지원하는 인터페이스로 매핑되는 데 속성 중 Type_Of_Basic Component 값을 OLB로 정한다.

Farmer 모델에 의하여 정의된 개념을 인터페이스 명세모델에서의 개념으로 매핑시키기 위하여 Farmer모델의 구성요소를 기준으로 다음과 같은 사항들을 고려하여야 한다.

1) Farmer 모델에서의 Aspect entity node

- ISM의 Aspect Interface로 매핑
- Aspect Interface Catalog 운영
- IFR(Interface Repository)에 AIFR(Aspect IFR)을 별도로 유지
- Aspect interface에 추가되는 기본속성
  Agent_ID
  Number_Of_Entity
  Successor_Node_Type
  Supported_Function
  Type_Of_Successor_Assocication :
  [Decompose, Specialize, Multiplicity]
  Serviced_Protocol
  Configuration_Of_Node

- Aspect interface에 추가되는 메소드
  Set_Aspect_ID (in aAspect_ID)
  Select_Association
  (out Type_Of_Succssor_Assocication)
  Is_Type_Of (in Successor_Interface)
  Add_New_Function (in aFunction)
  Delete_Function (in aFunction)
  Add_Protocol_Service (in Protocol_Type)
  Delete_Protocol_Service (in Protocol_Type)
  Add_New_DeviceElement (in aAgent, aDeviceElement)
  Delete_DeviceElement (in aAgent, aDeviceElement)

그림 2는 이와 같은 측면 인터페이스(aspect interface)의 예를 보여주고 있으며 이는 다음과 같은 CORBA IDL로 프리프로세싱된다.

- IDL Description : 표 2 참조

The Farmer Model	Interface Specification Model
Entity node	Interface
Aspect entity node	Aspect Interface
Uniformity entity node	Interface which has the same name in the interface catalog
Uniformity aspect entity node	Aspect Interface which has the same name in the aspect interface catalog
IM-Component	Interface which supports static Interface Invocation(SII), and attribute Type_Of_BasicComponent has "ILB" value
OM-Component	Interface which supports Dynamic Interface Invocation(DII), and attribute Type_Of_BasicComponent has "OLB" value
Decomposition	Decomposition
Specialization	Specialization
Multiplicity Abstraction	Refinement & Representative interface

표 1 Farmer 모델과 Interface Specification Model 모델의 Concept Mapping관계

2) Farmer 모델에서의 Multiplicity abstraction에 의한 representative entity node
- interface로 매핑, interface이름을 representative_NODE_ID로 정함
- 기본속성
Attribute_Of_BasicComponents: [ILB, OLB]
Number_Of_BasicComponents
Type_Of_Succssor_Association :
[Decompose, Specialize, Multiplicity]
- 기본 operations
Assign_Attribute_To_BasicComponent
(in BasicComponent)
Is_ComponentType_Of (in BasicComponent)
Is_InterfaceType_Of (in Successor_Interface)
Add_BasicComponent_To_MultiplicityLink
(in aComponent)
Delete_BasicComponent_To_MultiplicityLink
(in aComponent)
Select_Association
(out Type_Of_Succssor_Association)

3) Farmer 모델에서의 ILB
- Static Interface Invocation
- Interface로 매핑
- 속성 중 Type_Of_BasicComponent값이 ILB인 것으로 판단

4) OLB
- Dynamic Interface Invocation 수행
- Interface로 매핑
- 속성 중 Type_Of_BasicComponent 값이OLB인 것으로 판단

5) 개체노드(entity node)
- Interface로 매핑

Farmer 모델을 ISM모델로 변환시키는 FTI 알고리즘의 설명을 위하여 여기서 Farmer Model에서 사용하는 기본적인 definition에 대하여 알아볼 필요가 있다. 먼저 개체노드에 대한 정의는 다음과 같이 구조체(structure)로서 정의할 수 있다.

【 Definition 1 】 Entity Node Structure
If **E** is ***the set of the entity node structure,*** E can be defined as follows.

$$\forall e \in \mathbf{E},\quad e = < Eid, A, S, LT >$$

where, Eid : Name of the entity node e
A : Attribute Set (Refer to **Definition 2**)
S : It is the view that e has, and it signifies

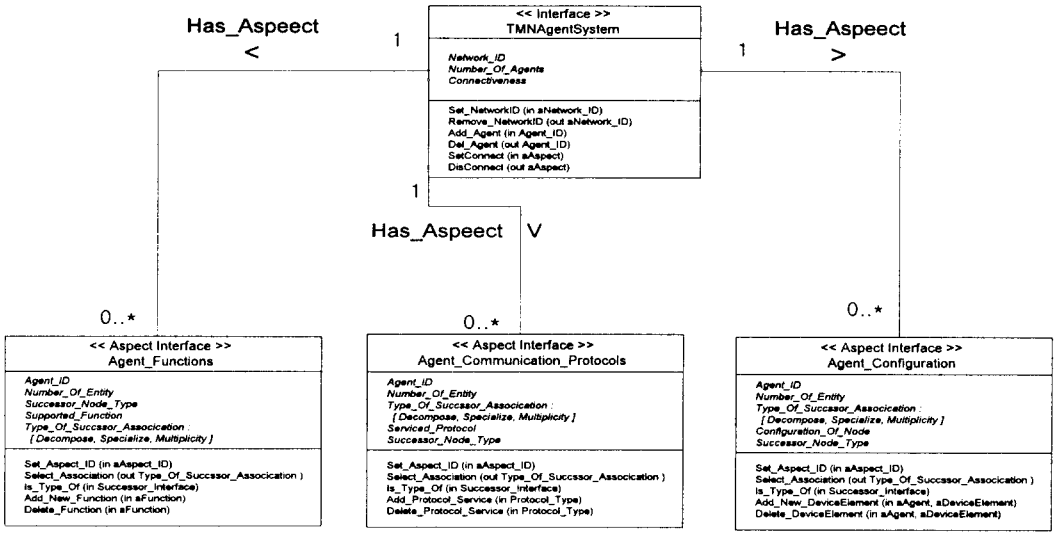

그림 2 Aspect Interface

the set of the aspect node name(id) (Refer to **Definition 3**)

LT : It is the loading type of the entity

LT ∈ { Dynamic, Static, none } □

【 Definition 2 】 Attribute Set of Entity Node Structure

The *attribute set* **A** of **E** can be defined as the following structure.

$$\forall a \in \mathbf{A},\ a = < Aid,\ AT >$$

where, Aid : Attribute name

AT : Attribute type set

AT ∈ { char, string, integer, real, boolean } □

측면개체 노드는 다음의 구조체(structure)로서 정의할 수 있다.

【 Definition 3 】 Aspect Node Structure

If S is ***the set of the aspect structure***, it can be defined as the following.

$$\forall s \in S,\ s = < ASPid,\ OWNER >$$

where, ASPid : Name of aspect s

OWNER : Set of the entity node name that has s as the aspect □

Farmer 모델에서 OLB와 ILB는 다음과 같이 정의하고 있다.

【 Definition 4 】 Structure of OM-Component Type Entity

The similarity type $\mathbf{E_o}$ of the entity structure set E is the set, such that,

{ e ∈ **E** | e.LT = Dynamic }.

This similarity type $\mathbf{E_o}$ is called ***the OM-Component type entity structure.*** □

【 Definition 5 】 Structure of IM-Component Type Entity

The similarity type $\mathbf{E_I}$ of the entity structure set E is the set, such that,

{ e ∈ **E** | e.LT = Static }.

This similarity type $\mathbf{E_I}$ is called ***the IM-Component type entity structure.*** □

위의 정의들을 바탕으로 Farmer 모델 구성요소 및 추상화 개념들을 ISM의 인터페이스 및 관계(relationship)으로 변형하는 FTI(Farmer model To Interface specification model) 알고리즘은 다음과 같다..

[Algorithm] FTI(E : Entity Type Node)

// Trans-mapping Farmer Model to Interface Specification Model

```
// TMN_Agent_Aspects.idl
Module TMN_Agent_Aspects {
        Aspect Interface Agent_Functions {
                void Set_Aspect_ID (in string aAspect_ID)
                void Select_Association (out any Type_Of_Succssor_Assocication )
                void Is_Type_Of (in any Successor_Interface)
                void Add_New_Function (in any aFunction)
                void Delete_Function (in any aFunction) }
        Aspect Interface Agent_Communiation_Protocols {
                void Set_Aspect_ID (in string aAspect_ID)
                void Select_Association (out any Type_Of_Succssor_Assocication )
                void Is_Type_Of (in any Successor_Interface)
                void Add_Protocol_Service (in any Protocol_Type)
                void Delete_Protocol_Service (in any Protocol_Type) }
        Aspect Interface Agent_Configuration {
                void Set_Aspect_ID (in string aAspect_ID)
                void Select_Association (out any Type_Of_Succssor_Assocication )
                void Is_Type_Of (in any Successor_Interface)
                void Add_New_DeviceElement (in any aAgent, in string aDeviceElement)
                void Delete_DeviceElement (in any aAgent, in string aDeviceElement) }
}
```

표2. TMN_Agent_Aspects.IDL

```
1. WHILE End of FMD DO
// FMD : Farmer Model Diagram

1.1 Traverse Farmer Model Diagram from node E to
    leaf node by the BFS(Breadth First Search)

2.1 Read Current Node

2.2 CASE type_of(current_node) OF

// Farmer Model Diagram의 current node가 entity
   node type인 경우
2.2.1 Entity Node Type :
// In case former relationship of current Entity Node
   is multiplicity relationship, skip.
IF Former relationship of current Entity Node =
   multiplicity
THEN
        Exit Case
END IF

// In case the next relationship of current Entity Node
   is multiplicity relationship, generate representative
   Entity Type Node.
IF the next relationship of current Entity Node =
multiplicity
THEN // Generate Representative Entity Type Node
        Get e.Eid // get name of entity node e
        Get e.A  // get attribute set of entity node e
        Set interface id by using  "Representative_"
        + e.Eid
        Set e.A to interface attributes
        Add new attribute to interface attributes.
        Add methods to list of operations of interface

ELSE // Generate interface
        Get e.Eid // get name of entity node e
        Get e.A  // get attribute set of entity node e
        Set e.Eid to interface id
        Set e.A to interface attributes
        Add methods to list of operations of interface
```

END IF

// Farmer Model Diagram의 current node가 aspect entity node type인 경우
2.2.2 Aspect Entity Node Type :
// Generate aspect interface
Get a.Aid // get name of aspect node
Set a.Aid to Agent_ID
Set any values to attribute set.
// attribute set의 내용
// Number_Of_Entity, Successor_Node_Type,
// Supported_Function,
// Type_Of_Succssor_Association:
// [Decompose, Specialize, Multiplicity]
// Serviced_Protocol, Configuration_Of_Node

Add methods to list of operations of aspect interface.

// Farmer Model Diagram의 current node가 Uniformity Entity Node Type 인 경우
2.2.3 Uniformity Entity Node Type :
Find any interface which has the same name of current Entity Node in the interface catalog

IF check_if_current_node = exist THEN
    Entity_Node ← Current Uniformity Entity Node Type
    Call FTI(Entity_Node)
ELSE
    Exception(No_Exist)
ENDIF

// Farmer Model Diagram의 current node가 Uniformity Aspect Entity Node Type 인 경우
2.2.4 Uniformity Aspect Entity Node Type :
Find any aspect interface which has the same name of current Aspect Entity Node in the aspect interface catalog

IF check_if_current_node = exist THEN
    Aspect_Entity_Node ← Current Uniformity Entity Node Type
    Call FTI(Aspect_Entity_Node)
ELSE
    Exception(No_Exist)
ENDIF

// Farmer Model Diagram의 current node가 ILB Multiplicity Component Type Node 인 경우
2.2.5 ILB Multiplicity Component Type Node:
// Generate interface
Get e.Eid // get name of entity node e
Get e.A // get attribute set of entity node e
Set e.Eid to interface id
Set e.A to interface attributes
Add new attribute Type_Of_BasicComponent & set it to [ILB].

// Farmer Model Diagram의 current node가 OLB Multiplicity Component Type Node 인 경우
2.2.6 OLB Multiplicity Component Type Node:
// Generate interface
Get e.Eid // get name of entity node e
Get e.A // get attribute set of entity node e
Set e.Eid to interface id
Set e.A to interface attributes
Add new attribute Type_Of_BasicComponent & set it to [OLB].

2.3 END CASE

2.4 Read Current Relationship between nodes

2.5 CASE type_of(Current_Relationship) OF

// Farmer Model Diagram의 current relationship 이 decomposition relationship 인 경우
2.5.1 Decomposition:
Generate Decomposition Relationship.

// Farmer Model Diagram의 current relationship 이 specialization relationship 인 경우
2.5.2 Specialization:
Generate Specialization Relationship.

// Farmer Model Diagram의 current relationship 이 Multiplicity relationship 인 경우
2.5.3 Multiplicity:
Generate Association Relationship.

// Farmer Model Diagram의 current relationship 이 Multiplicity relationship 인 경우

2.5.4 Multiplicity Instance Link

// E_C : It signifies the set of the elements that have dynamic or static loading type, that is, the set of OM-Component type entity node and IM-Component entity type node

Get E_C from the Farmer Model Diagram.
For $\forall e \in E_C = \{ e_i \mid 1 \le i \le n \}$,
FOR i = 1 to n
    // Generate interface
    Get e_i.Eid // get name of entity node e
    Get ei.A // get attribute set of entity node e
    Set e_i.Eid to interface id
    Set e_i .A to interface attributes

    IF e_i.a.LT = Dynamic THEN
        Add new attribute Type_Of_Basic Component & set it to [OLB].
    ELSE
        Add new attribute Type_Of_Basic Component & set it to [ILB].
    END IF

END FOR

2.6 END CASE

3. END WHILE

FTI 알고리즘의 correctness에 대한 증명은 trivial하다. FTI 알고리즘을 이용하여 그림 4에서 보여주는 Farmer 모델에 의하여 디자인된 TMN Agent 설계는 그림 5 ~ 그림 8에서 보여주는 바와 같이 인터페이스 명세모델에 의한 인터페이스 명세로 변형됨을 알 수 있다.

III. 결론

실세계의 사물을 다중측면에 의하여 분석한 후 개체노드, 측면개체노드, 균일성 개체 및 다중화 추상화라는 여러 개념을 이용하여 디자인할 수 이론적 모델에 머물던 기존의 Farmer 모델의 한계에서 벗어나 실제 구현 방안을 제시하였다는 점에서 본 논문의 의미를 찾을 수 있다. 즉, Farmer 모델에 의하여 표현되는 개체노드, 측면노드, 균일성 개체노드, 균일성 측면노드 등의 노드, 다중성 등의 추상화 개념들을 인터페이스로 변형시킴으로서 CORBA나 JAVA[18]의 패키지를 통하여 다양한 구현방법을 사용할 수 있을 뿐만 아니라 ILB 및 OLB 개념을 인터페이스로 매핑시킴으로서 ILB / OLB 타입의 개체노드의 재사용성을 극대화시킬 수 있는 장점이 있다. 본 논문에서는 Farmer 모델에 의하여 디자인된 망관리 시스템 에이전트의 구성모델을 컴포넌트 기반 개발방법론에서 제시하는 컴포넌트-인터페이스 메타모델로의 변형을 위한 FTI알고리즘을 제안하였다.

참고문헌

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

[2] James Rumbaugh and Michael Blaha, *Object-Oriented Modeling and Design*, OMG, 1991

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

[4] ComponentWare Consortium, "ComponentWare ArchI tecture : A technical product description", *I-Kinetics, Inc*, 1995.

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

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

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

[8] Soo-Hyun Park, Doo-Kwon Baik, "Platform 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

[9] Soo-Hyun Park, Doo-Kwon Baik, "The Farmer Model with the Component Farming Concept for Developing TMN Systems", *Journal of Circuits, Systems, and Computers,* World Scientific Publishing Co., Vol.9, Nos. 1 & 2, 1 - 22, Singapore, 1999

[10] Keith Short, "Component Based Development and Object Modeling", *White Paper,* Sterling Software, 1997