Transformation Methodology : From the Farmer Model To Component Interface Meta Model

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

Faculty of Computer Application Engineering, Dongeui University,
San 4, Kaya-dong, PusanJin-ku, Pusan, 614-714, KOREA
Tel:+82-51-890-1725,1727,1707 Fax:+82-51-890-1704
E-mail:{shp, sgmin, tskim}@hyomin.dongeui.ac.kr

Abstract *A fundamental tenet of CBD is that a component has a specification which describes what that component does, and how it behaves when its services are used. In general, the implementation may be written in a different programming language and execute on a different technology platform, from the language and platform used by the client program. In order to implement practically the system that is designed by the Farmer model, there is need to have the ISM (Interface Specification Model) which explains specification about the functions of entities of the Farmer model, such as, entity node, aspect node and ILB/OLB. This paper suggests the transformation methodology from the concepts of the Farmer model to the mapping notions of the ISM. Also in reality, TMN agents system which is designed by the Farmer model is transformed to the ISM system design.*

1. Introduction

In most of the models including the object oriented model[1][2] which has been emerged to escape from the software crisis, the entity of the real world cannot be observed in various aspects since it can be described in the fixed aspect according to the view point of the model constitutor. The Farmer model[3] that is proposed in order to overcome this disadvantage analyzes the entity of the real world in the aspect of the various points that is not the fixed aspect. After that, the aspect elements that have completed the analysis are defined to be the aspect object. The Farmer model is theoretical model, by which we can analyze the real world system-entities into several multiple aspects and design them by using the concept of entity node, aspect entity node, uniformity entity node, uniformity aspect entity node and multiplicity abstraction. In order to implement practically the real world system-entities designed by the Farmer model to allow components built using different languages (not so Javabeans[4], CORBA beans[5][6] unless combined with one of the others), different tools and compiled with different compilers, to interoperate through clearly defined interfaces, there are a great need of the Interface Specification Model[7 - 13] to show specifications about functions which the entity objects of the Farmer model, such as, entity node, aspect entity node and ILB/OLB[3], carry out. Nowaday, CORBA is the most important middleware project over undertaken by our industry. It has been designed to allow intelligent components to discover each other and interoperate in an object bus. CORBA also specifies an extensive set of bus-related services for creating and deleting objects, accessing them by name, storing them in persistent stores, externalizing their states, and defining ad hoc relationships between them. Most common uses of the word 'component' mean an independently deliverable unit of software that encapsulates its design and implementation and offers standard interfaces to the outside, by which it may be composed with other components to form a larger whole. This paper shows the transformation methodology from the concepts defined in the Farmer model to the construction elements of the Interface Specification Model which is proposed in the Component Based Development(CBD), and also explains transformation algorithm. Furthermore, we can see the real example of transformation of Telecommunication Management Network(TMN) agent[14][15] design that is executed by from the Farmer model to Interface Specification Meta Model.

2. Component Interface Meta Model

A component is a collection of operations, intended to be used as a building block when constructing applications or larger-grained components, that is made available as an independently delivered software package. A software component is an independently deliverable package of reusable software services.

In general, a component has three facets. The first facet is a specification, which describes the semantics. This explains what the component does, and how a client should use it. The second facet is an implementation design, which describes how an implementer has chosen to design and construct software and data stores to meet the intended specification. The last is an executable which delivers the component's capability on a designated platform. It is worth emphasising that not all facets of a component have to be present to make a component usable. For example, the builder of a component may be happy to publish the details of the specification to maximise its

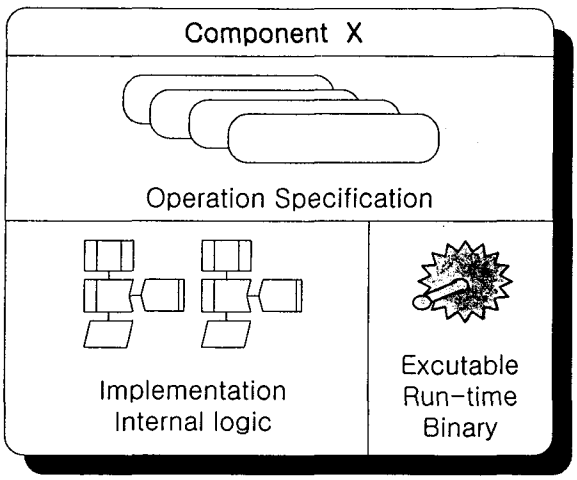

Figure 1 Three Facets of a Component

reuse, but unwilling to publish his implementation design which he may regard as his own intellectual property.

CBD is defined as "the industrialization of the software development process based on assembly of prefabricated software components" and CBD is based on two basic ideas. Firstly, that application development can be significantly improved if applications can be quickly assembled from pre-fabricated software components. Secondly, that an increasingly large collection of interoperable software components will be made available to developers in both general and specialist catalogues.

A fundamental tenet of CBD is that a component has a specification which describes what that component does, and how it behaves when its services are used. Given knowledge solely of the specification, any potential user, or client, of those services can focus on his part of the overall solution without concern as to how those services are actually rendered. The services will be rendered by some programmer or designer who provides an implementation for the component, expressed in terms of code and data which will be guaranteed to meet the specification. In general, the implementation may be written in a different programming language and execute on a different technology platform, from the language and platform used by the client program. One component may be replaced by another in an application, as long as both implement the same specification. The split between specification and implementation is the essence of the term encapsulation. The specification acts as a firewall between the component provider and its consumer - restricting the effects of changes on either side, provided the firewall is left intact.[7][8]

3. Transformation Methodology for the Definition of COM IDL Interface

In order to implement practically the system that is designed by the Farmer model, there is need to have the Interface Specification Model which explains specification about the functions of entities of the Farmer model, such as, entity node, aspect node and ILB/OLB.

This paragraph suggests the transformation methodology from the concepts of the Farmer model to the mapping notions of the ISM. Also in reality, TMN agents system which is designed by the Farmer model is transformed to the ISM system design.

In order to support aspect nodes of the Farmer model, the ISM has aspect interface. Especially, aspect interface catalog keeps specific information of aspect interfaces. Uniformity entity node of the Farmer model is transformed into the interface having the same identity in the interface catalog. In the case of uniformity aspect entity node of the Farmer model, it is mapped to the aspect interface having the same identity in the aspect interface catalog.

Representative entity nodes generated by the multiplicity abstraction of the Farmer model are transformed to representative interfaces of the ISM. ILBs in the Farmer model are mapped to interfaces of the ISM which support static interface invocation (SII) and these interfaces set their attribute value, Type_Of_BasicComponent, into "ILB". In the case of OLBs of the Farmer model, they are transformed into interfaces of the ISM which support dynamic interface invocation(DII) and these interfaces set their attribute value, Type_Of_BasicComponent, into "OLB".

For mapping from the concepts of the Farmer model to the notions of the ISM, we take into account several kinds of considerations as follows in view of the Farmer model construction elements.

1) Entity node
- Mapping into the interface of the ISM

2) Aspect entity node
- Mapping into the aspect interface of the ISM
- Keep aspect interface catalog
- Maintain AIFR(Aspect IFR) apart from IFR(Interface Repository)
- Basic attributes of aspect interface
  Agent_ID
  Number_Of_Entity
  Successor_Node_Type
  Supported_Function
  Type_Of_Successor_Association :
  [Decompose, Specialize, Multiplicity]
  Serviced_Protocol
  Configuration_Of_Node

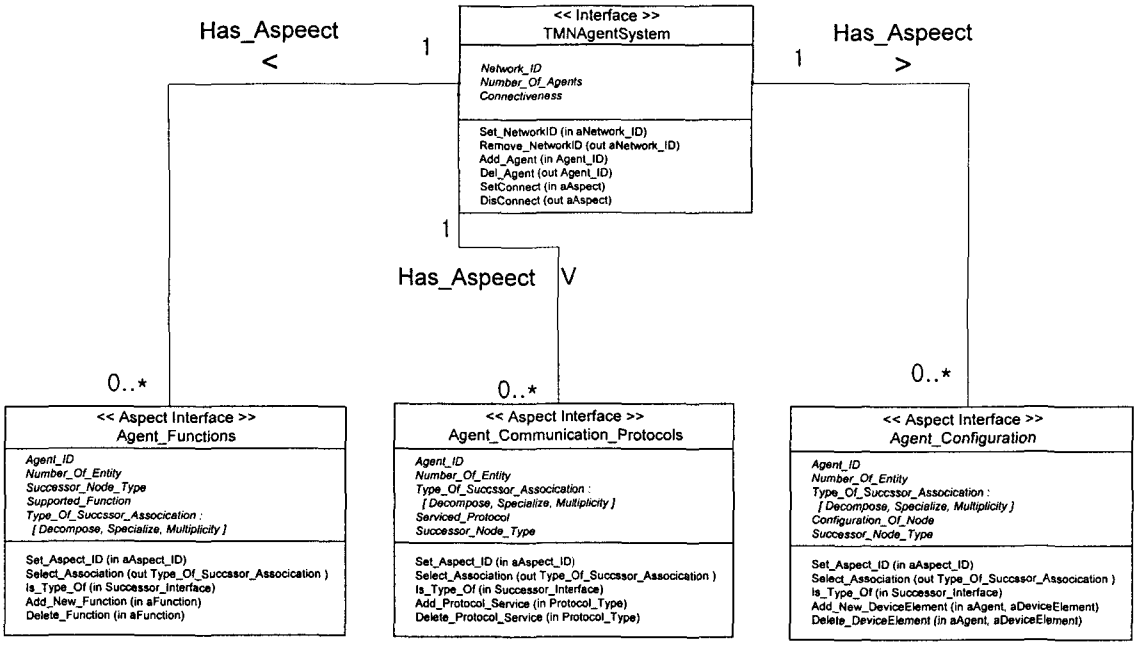

Figure 2 Mapping to the ISM from the Farmer Model in the case of TMN Agent Design

- Methods of Aspect interface
  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)

Figure 2 shows this kind of example of aspect interface and these aspect interfaces are preprocessed into CORBA IDL as figure 3.

3) Representative entity node generated by the multiplicity abstraction
- *Mapping into the interface of the ISM.* This interface has the name "representative_NODE_ID"
  NODE_ID : ID of Representative entity node
- Basic attributes of interface
  Attribute_Of_BasicComponents: [ILB, OLB]
  Number_Of_BasicComponents
  Type_Of_Succssor_Association :
  [Decompose, Specialize, Multiplicity]
- Basic operations of interface
  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_Assocication)

4) ILB
- Static Interface Invocation
- Mapping into the Interface of the ISM
- Attribute "Type_Of_BasicComponent" has "ILB" value.

5) OLB
- Dynamic Interface Invocation
- Mapping into the Interface of the ISM
- Attribute "Type_Of_BasicComponent" has "OLB" value.

4. Conclusion

In this paper, we have defined CBD as the industrialization of the software development process based on assembly of prefabricated software components. Given knowledge solely of the specification, any potential user, or client, of those services can focus on his part of the overall solution without concern as to how those services are actually rendered. This paper shows the transformation

```
// 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)
        }
}
```

Figure 3. IDL of TMN_Agent_Aspects

methodology from the concepts defined in the Farmer model to the construction elements of the Interface Specification Model which is proposed in the Component Based Development(CBD), and also explains transformation algorithm. Furthermore, we can see the real example of transformation of Telecommunication Management Network(TMN) agent design that is executed by from the Farmer model to Interface Specification Meta Model.

References

[1] James Martin and James J. Odell, "Object Oriented Software Analysis & Design", Prentice-Hall, 1992

[2] Ivar Jacobson, "Object-Oriented Software Engineering, A Use Case Driven Approach", Addison-Wesley,1992

[3] Soo-Hyun Park, Doo-Kwon Baik, "7. Evaluation of a Methodology for Construction of TMN Agents in the Aspects of Cost and Performance", *System Development Methods for Databases, Enterprise Modeling, and Workflow Management, Kluwer Academic / Plenum Publishers*, pp.109 - 138, Edited by W.Gregory Wojtkowski, Wita Wojtkowski, Stanislaw Wrycza, and Joze Zupancic, U.S.A., 2000

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

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

[6] Jon Siegel, "CORBA, Fundamentals and Programming", OMG, 1996

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

[8] The CBD*96* Standard Version 2.1, Sterling Software, 1998

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

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

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

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

[13] Desmond F.D'Souza, Alan C.Wills, *Objects, Components, And Frameworks With UML, The Catalysis Approach, Addison-Wesley*, 1998.

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

[15] Salah Aidarous and Thomas Plevyak, "TMN into the 21st Century, Techniques, Standards, Technologies and Applications", IEEE Press, 1994