# Modeling Language for LDAP and Automatic Production System

Kenji Saotome[*], Seiichi Kondo[**], Akihisa Onuma[***], Takashi Komiya[***], Masahiko Ishino[****], Tsukasa Kudo[**], Sanshiro Sakai[*****] and Tadanori Mizuno[******]

[*]Hosei Business School of Innovation Management, Japan
[**]Mitsubishi Electric Information Systems Corporation, Japan
[***]Mitsubishi Electric Corporation, Japan
[****]Fukui University of Technology, Japan
[*****]Faculty of Informatics, Shizuoka University, Japan
[******]Graduate School of Science and Technology, Shizuoka University, Japan

***Abstract*** - LDAP Directory Service begins to be used as a tool for the development of Enterprise Information System. Nevertheless, there are not the standards of the model for the design of the Directory and generally incomplete diagrams of the Directory have been illustrated. The methods to design the Directory are expected. Then, we proposed the modeling language extending UML for the design of Directory Information Tree (DIT). We developed for trial the system that automatically generates the programs that manage the Directory, and evaluated it. We found that this system is enough applicable and efficient.
***Keywords***: Directory, LDAP, UML, MDA, Modeling.

## 1 INTRODUCTION

In late years the standardization of the directory service advances, and the directory service products which manage the data of the information system of the company come to be released, and it begins to be used. However, when the needs to build the original data structure by the system matter occurred, it becomes custom to show the incomplete figure of the degree to exemplify hierarchical structure and do it with a design document because there is not a standard design model peculiar to the directory. Therefore by the conventional construction technique of the directory, it becomes difficult to get the desired directory for the program developer and the user because we are not able to realize the smooth mutual understanding between the directory designer and the program developer and the user using it. There is the danger that a problem occurs just before the operation after the development.

We already proposed *Directory Modeling Language* specialized in the directory service based on the UML model, and produced the system(*Automatic Production System*) which generated *Directory Management Program* from this modeling language automatically and demonstrated the applicability of this system [1].

This research domain is the directory, programs automatic production and UML expansion, but the other researches corresponding to these all domains are not found. Therefore the theme of this research is an advanced one [2][3] [4].

In this time, we added the notation of the schema definition of the object class and the attribute type to the modeling language and realized the automatic generation function of the schema, and the system implementing the specifications of the practical use level was completed. By this report, we report about Directory Modeling Language and the specifications of Directory Management Program that Automatic Production System generates and the evaluation of this system.

The directory of this report has the structure of ITU-T (International Telecommunication Union-Telecommunication Standardization Sector) Recommendation X.500 series and the interface of LDAP(Lightweight Directory Access Protocol) defined by IETF(Internet Engineering Task Force)[5][6][7]. The directory modeling language to propose is based on UML(Unified Modeling Language) standardized in OMG(Object Management Group) [8][9].

## 2 MODELING LANGUAGE SPECS

This section describes the syntax specifications and the semantic specifications of Directory Modeling Language.

### 2.1 Syntax Specification

The model of Directory Modeling Language is expressed by the *class diagram* of UML. In the class, the *stereotype* and the *tagged value* is specified to show the role of the class. We show below the syntax specifications of this modeling language..

#### 2.1.1 <<LDAP>> Class

(1)For a *model*, there must be only one *class* specifying an <<LDAP>> stereotype.

(2)In this class, a {LDAPRoot} tagged value must be specified. In {LDAPRoot} tagged value, a *character string* is specified.
(3)In this class, the *association* must not be specified.

### 2.1.2 <<LDAPEntry>> Class

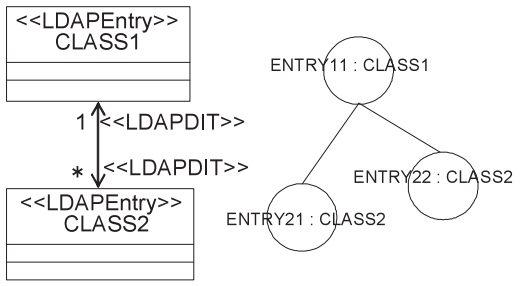
(1)In this model the class name must be unique among the classes specified <<LDAPEntry>> stereotype.
(2)In the class, one or more {LDAPObjectClass} tagged values must be specified. In {LDAPObjectClass} tagged value, a character string is specified.
(3)The class has one or more *attributes*.
(4)The attribute consists of the attribute name and the *type*. The attribute name must be unique in this class.
(5)The type must be **String**, **byte[]**, **Collection**, **Collection<String>** or **Collection<byte[]>**.
(6)<<LDAPRDN>> stereotype must be specified for one or more attributes in a class.
(7) The association between classes is able to be specified, and the *multiplicity*, the *navigability* and the *role* name at the association end are able to be specified. When there are one or more associations in a class, the role name at each association end must be unique each other.
(8)The multiplicity of the association is specified the following one. The notation (*) means more than 0.
- 1
- *

(9) The arrow of both directions or the arrow of single direction is specified for the navigability of the association.
(10) At the association end, one stereotype of the following kinds must be specified.
- <<LDAPDIT>>
- <<LDAPDN>>
- <<LDAPAttr>>

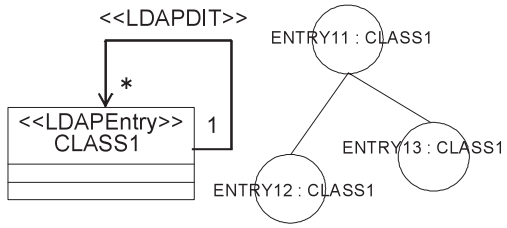
(11)When <<LDAPDIT>> stereotype at the association end is specified, <<LDAPDIT>> stereotype must be specified at another association end if nessesary. The multiplicity at this association must be * vs. 1.
(12) At the association end of <<LDAPAttr>> stereotype, {LDAPKey} tagged value must be specified.

### 2.1.3 <<LDAPDefAttributeTypeS>> Class

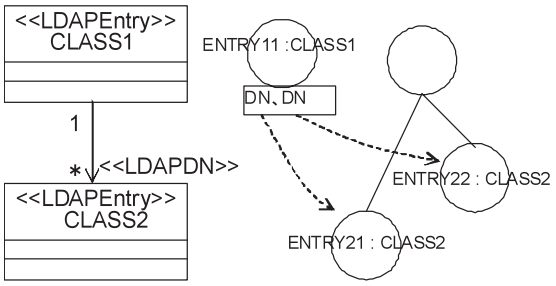
(1)In this class, a {LDAPSyntax} tagged value must be specified. A character string is specified in {LDAPSyntax} tagged value.
(2)The class has one or more attributes.
(3)The attribute consists of the name and the type.
(4)The attribute name must be unique among the attribute names described in the classes of <<LDAPDefAttributeTypeS>> stereotype.
(5)In the type, **void** or **Collection** is specified .



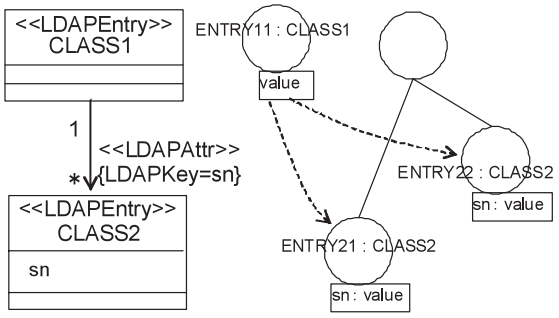

(a) <<LDAPDIT>> and DIT.




(b) <<LDAPDIT>> and DIT(Self Assoc).




(c) <<LDAPDN>> and DIT.




(d) <<LDAPAttr>> and DIT.

Figure 1 : Modeling Language and DIT.

(6)In this attribute, if necessary, {LDAPEquality}, {LDAPOrdering} or {LDAPSubstr} tagged value is specified.
(7)In this class, the association must not be specified.

#### 2.1.4 <<LDAPDefObjectClass>> Class

(1)In this class, a {LDAPSuperior} tagged value must be specified. In {LDAPSuperior} tagged value, a character string is specified.
(2)A class has one or more attributes.
(3)The attribute consists of the attribute name and the type.
(4)The attribute name must be unique in a class.
(5)**void** must be specified if specified.
(6)For each attribute, <<LDAPMust>> stereotype is able to be specified.
(7)In this class, the association must not be specified.

## 2.2 Semantic Specifications

### 2.2.1 Class

#### 2.2.1.1 <<LDAP>> Class

In the class specified <<LDAP>> stereotype, the information of the whole model is specified.

In {LDAPRoot} tagged value specified in this class, *DN(Distingushed Name)* which is top entry of all directory entries belonging to the classes of this model is specified.

#### 2.2.1.2 <<LDAPEntry>> Class

The class specified <<LDAPEntry>> stereotype defines the directory entry. In the class, an *object class* of the directory constituting an entry belonging to the class is specified in the {LDAPObjectClass} tagged value. One or more {LDAPObjectClass} tagged values are able to be specified, but **top** object class is omitted.

The each attribute name defined in the class must be the *attribute type* of the directory to belong to one of the object classes specified in {LDAPObjectClass} tagged value.

The type of the attribute defined in the class is the type of JAVA used in APIs which Automatic Generation System generates.

When the *attribute syntax* shows binary data such as **Binary**, **Octet String** or **Certificate**, **byte[]** is specified when a single value, and **Collection<byte[]>** is specified when multi values. When the attribute syntax shows string data such as **Directory String**, **Boolean** or **Integer**, string is specified when a single value, and **Collection** or **Collection<String>** is specified when multi values.

<<LDAPRDN>> stereotype shows that it is an attribute type to become *RDN (Relative Distinguished Name)* of the entry.

#### 2.2.1.3 <<LDAPDefAttributeTypeS>> Class

The class specified <<LDAPDefAttributeTypeS>> stereotype defines the user-defined attribute type with the *attribute syntax*. The attribute syntax is specified in {LDAPSyntax} tagged value of the class. The attribute syntax (the **SYNTAX** keyword in the attribute type definition of the directory) is specified in {LDAPSyntax} tagged value.

The attribute of the class shows the attribute type of the directory, and the attribute name shows the name of attribute type. In the type, **void** must be specified when attribute type of single value, and **Collection** must be specified when attribute type of multi values.

If necessary, {LDAPEquality}, {LDAPOrdering} and {LDAPESubstr} tagged value are able to be specified in each attribute of the class. {LDAPEquality}, {LDAPOrdering} and {LDAPESubstr} tagged value show the *matching rule* of **EQUALITY** , **ORDERRING** and **SUBSTR** each.

For the character string to specify in these tagged values, the keyword of the matching rule of the attribute type definition of the directory must be specified.

#### 2.2.1.4 << LDAPDefObjectClass >> Class

The class specified <<LDAPDefObjectClass>> stereotype defines user-defined object class. In {LDAPSuperior} tagged value specified in the class, the superior object class (the keyword **SUP** of the object class definition) is specified.

The attribute of the class shows the attribute type to belong to this object class. The attribute name shows the attribute type name. <<LDAPMust>> stereotype in each attribute shows that this attribute type is the required attribute type of this object class.

### 2.2.2 Association

The association shows the association between the entries. Either of <<LDAPDIT>>, <<LDAPDN>> or <<LDAPAttr>> stereotype is specified at the association end to show the implementation of the association of the directory.

#### 2.2.2.1 Association of <<LDAPDIT>>

<<LDAPDIT>> stereotype shows that the association is implemented by *DIT (Directory Information Tree)*. The multiplicity of this association must be * vs. 1. This association is implemented by the method that an entry belonging to the class of the association end specified "1" is posted as the direct upper entry of the entry belonging to the class at the other association end.

When there are more than 2 associations by <<LDAPDIT>> stereotype in a class, this class cannot have more than 2 association that have the "*" multiplicity among these association ends at this class side. Figure 1(a) shows an example of the class to use <<LDAPDIT>> stereotype and DIT implemented. The implementation of the association by DIT is the most natural structure for the directory. As to express the organization of the company, the self association to

associate with the own class is the typical example associated by DIT of the directory. Figure 1(b) shows the class diagram expressing the self association and DIT implemented.

#### 2.2.2.2 Association of <<LDAPDN>>

The <<LDAPDN>> stereotype shows that the association is implemented by DN. An entry belonging to a class has DN of an entry belonging to the class at the other association end. This implementation resembles the method that is used for the relations of the group and the member to be called the static group in the directory. Figure 1(c) shows the class diagram to use the <<LDAPDN>> stereotype and DIT implemented. The dotted lines in this figure are the notation to show the associated entries, and are not to constitute the real DIT.

#### 2.2.2.3 Association of <<LDAPAttr>>

The <<LDAPAttr>> stereotype shows that the association is implemented using an attribute value of the attribute type. {LDAPKey} tagged value is specified on the association end. In {LDAPKey} tagged value, the attribute name to use for the association among the attributes of the class at the other association end is specified. When the value for this association in the entry is equal to the attribute value in the attribute type specified by the {LDAPKey} tagged value belonging to the class of the other association end, it is considered that these entries are associated. This implementation resembles the relations of the primary key and the foreign key of the relational data base. Figure 1(d) is the figure of the class diagram specified <<LDAPAttr>> stereotype and DIT implemented.

## 3 DIRECTORY MANAGEMENT PROG

### 3.1 Outline

We developed for trial the system(Automatic Production System) to generate automatically the program(Directory Management Program) to manage the directory from the model of Directory Modeling Language.

Directory Management Program has the following specifications so that assuming trial.

- •Only import by batch of all data
- •Only access by the application program

Directory Management Program consists of *Directory Loading Compiler*, *Directory Access API* and *User-defined Schema Information*.

The example of figure 2 assumes the directory which manages the information of the organization of the company, the region where it is located at and the employee who belonged to it. We select the standard attribute type as much as possible, but define the user-defined attribute type when we cannot use it. In figure 2, **addExtTelNumber**, **addExtFaxTelNumber**, **addCn** and **addSn** of **Employee** are the user-defined attribute type. **addExtTelNumber** and **addExtFaxTelNumber** show the extension telephone or FAX number. **addCn** and **addSn** are additional **Cn** and **Sn**. For example these are defined for the purpose of keeping the full name that includes GAIJI with the character code set (Shift-JIS etc) except utf-8.

Figure 3 describes the definition information of these user-defined attribute types. **addExtTelNumber** and **addExtFaxTelNumber** are defined as the attribute type of the <<LDAPDefAttributeTypeS>> class which name is **DirectoryString**. **addCn** and **addSn** are defined as **Binary** because they are expressed with the character code set except utf-8. The **addOrgPerson** object class that these user-defined attribute types belongs to is defined in the <<LDAPDefObjectClass>> class.

### 3.2 Directory Loading Compiler

#### 3.2.1 Characteristics

Directory Loading Compiler has the following characteristics.

(1)Incomplete information is not left in the directory because the loading script can detect the structural error of DIT at the compilation time.

(2)The loading script can check the relation of definition and reference about the association.

(3)To use the LDIF form it is necessary to be specified the DN of the each entry itself and the related entry, but to use the loading script the loading data can be specified without being conscious of DN

#### 3.2.2 Specifications

Directory Loading Compiler interprets the loading script for the loading directory data by batch, and generates the loading data of the LDIF form. Figure 4 is the script described in the specifications of Directory Loading Compiler generated by the model of Figure 2. **Region**, **Section** and **Employee** of Figure 4 is the class name, and the entry corresponding to this each class is specified in **#entry()**. In **#entry()**, the attribute type and the attribute value is specified in the form of **parameter=value**. For example, **st="Tokyo"** of the entry of **Region** shows that **st** is the attribute type and **Tokyo** is the attribute value.

At the following {} of **#entry()**, the entry being the lower entry in DIT is specified. The entry of **ou="sales department"** and **ou="Development Division"** of **Section** class becomes a direct lower entry which RDN is **st="Tokyo"**. In this way, the DN of each entry is decided by expressing DIT structure with the nest and RDN of each entry.

The role name in the class diagram is used to express the association except DIT. The **section** parameter

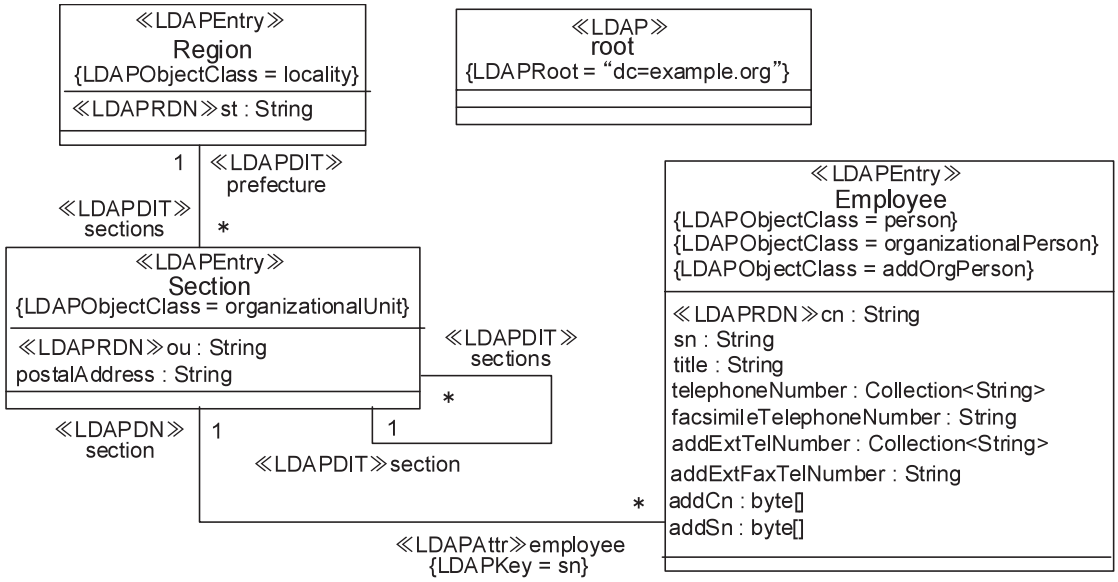



Figure 2 : DIT Model by Directory Modeling Language.

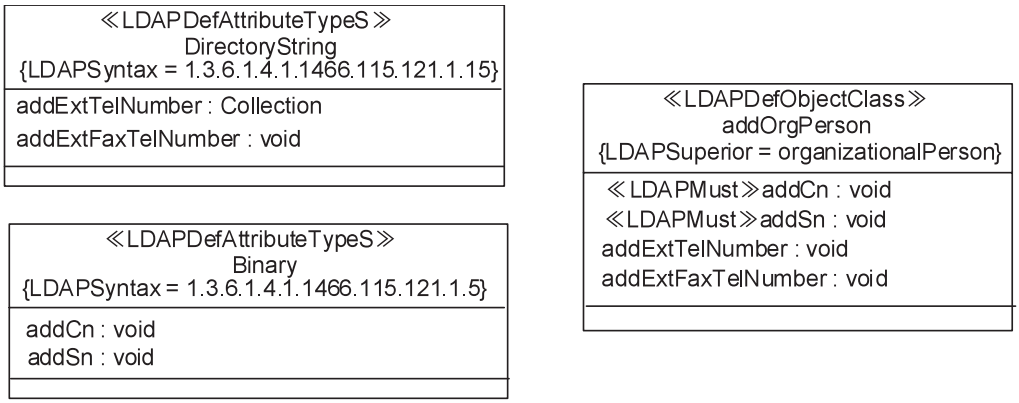



Figure 3 : Scheme Definition Model by Directory Modeling Language

specified in the entry of **Employee** class is the role name specified at the <<LDAPDN>> stereotype of the class diagram, and the associated entry at this parameter is specified.

## 3.3 Directory Access API

### 3.3.1 Characteristics

Directory Access API has the following the characteristics.

(1)The API uses JNDI (JAVA Naming and Directory Interface) which is JAVA standard interface of LDAP directory access and is implemented as the API which does not depend on the directory server.

(2)Directory Access API can be accessed with the interface of the design pattern adopted in EJB (Enterprise JavaBeans).

### 3.3.2 Specifications

Directory Access API has the functions to get the entry, the attribute value of the entry and the associated entry to the entry and so on.

Figure 5 shows the function of the API to access the entry belonging to **Section** class of Figure 2. **findByPrimaryKey()** method of **SectionHome** class gets the instance of **Section** class by the attribute type of RDN. **findAll()** method get all the entries belonging to **Section** class. **getDn()**, **getOu()** and **getPostalAddress()** of **Section** class get the attribute value of the entry. **getRegion()**, **getSection()**, **getSections()** and **getEmployee()** get the associated entry. **SectionDTO** class is the JavaBeans implemented **Java.io.Serializable** and has all of the attribute values.

```
Region {
   #entry (st="Tokyo") {
      Section {
         #entry (ou="Sales", postalAddress="Shinjuku-ku, Tokyo", employees=<sn="A"> <sn="B">) ;
         #entry (ou="development", postalAddress="1, Chiyoda-ku, Tokyo", employees=<sn="C">) {
            Section {
               #entry (ou="development-1", postalAddress="2, Chiyoda-ku, Tokyo", employees=<sn="D"> <sn="E">) ;
               #entry (ou="development-2", postalAddress="3, Chiyoda-ku, Tokyo", employees=<sn="F"> <sn="G">) ;
 ……
}
Employee {
   #entry (cn="A", sn="A", title="chief", telephoneNumber="03-1111-0001", facsimileTelephoneNumber="03-1111-1001",
      addCn="gmA=", addSn="gmA=", addExtTelNumber="11-0001", addExtFaxTelNumber="11-1001", section=<ou="Sales">) ;
   #entry (cn="B", sn="B", title="staff", telephoneNumber="03-1111-0002", facsimileTelephoneNumber="03-1111-1002",
      addCn="gmE=", addSn="gmE=", addExtTelNumber="11-0002", addExtFaxTelNumber="11-1002", section=<ou="Sales"> ) ;
 ……
   #entry (cn="I", sn="I", title="staff", telephoneNumber="06-2222-0002", facsimileTelephoneNumber="03-2222-1002",
      addCn="gmg=", addSn="gmg=", addExtTelNumber="22-0002", addExtFaxTelNumber="22-1002", section=<ou="Personnel"> ) ;
}
```

Figure 4 : Input of Directory Loading Compiler.

## 3.4 User-defined Schema Information

### 3.4.1 Characteristics

User-defined Schema Information has the following characteristics.

(1)The User-definition Schema Information does not depend on the directory server because it is generated by the standard LDIF form.

### 3.4.2 Specifications

The User-defined Schema Information consists of the data of the LDIF form to register the user-defined attribute type generated from the information of the class of <<LDAPDefAttributeTypeS>> stereotype and the user-defined object class generated from the information of the class of <<LDAPDefObjectClass>> stereotype. Figure 6 is the LDIF formed data of the attribute type definition generated from the <<LDAPDefAttributeTypeS>> class of Figure 3. Figure 7 is the LDIF formed data of the object class definition generated from the <<LDAPDefObjectClass>> class of Figure 3.

## 4 EVALUATION

In comparison with the existing application program using the directory, we constructed the directory, tested the function and measured the performance.

The object of the evaluation is Information Leakage Prevention Solution of Mitsubishi Electric Corporation (Mitsubishi Solution System as follows)[10]. We describe the functionally equal directory with Directory Modeling Language for the organization and the employee information of the directory of Mitsubishi Solution System, compare the performance of the each program that we make with Directory Access API and the API of Mitsubishi Solution System.

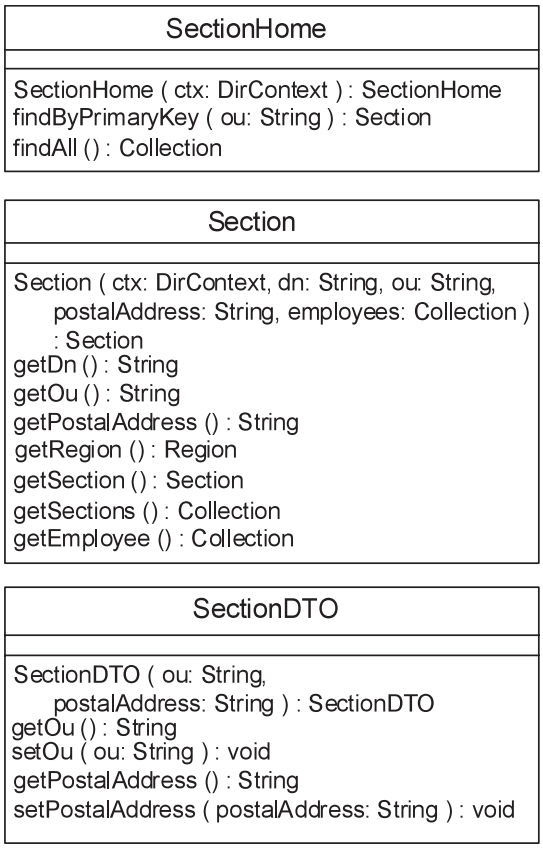



Figure 5 : Methods of Class "Section".

## 4.1 User-defined Schema definition

We defined 35 user-defined object classes and 136 attribute types of Mitsubishi Solution System and were able to confirm the same schema definition.

By this definition, the classes in the model that we described by Directory Modeling Language were 7 <<LDAPDefAttributeTypeS>> classes and 35 <<LDAPDefObjectClass>> classes.

What 136 attribute types are able to be expressed by 7 <<LDAPDefAttributeTypeS>> classes shows that the attribute syntaxes of these attribute types are 7 kinds.

## 4.2 Building DIT

We were able to confirm that the DIT approximately same as Mitsubishi Solution System was build.

There were 9 <<LDAPEntry>> classes, 30 attributes and 13 associations in the model that we described.

## 4.3 Performance Measurement

### 4.3.1 Measurement Method

#### 4.3.1.1 Data Structure

We assume the company of 5,000 employees and compare the performance using data for the measurement such as Table.1.

#### 4.3.1.2 Evaluation Programs

We assume the implementation of the address book by the directory and use the following two programs for the performance comparison.

(1) Program1

This is the program that is assumed to find the employees from the organization hierarchy. While referring in sequence from the main office to the lower organization, this program displays the data of the organizational unit and all the employees belonging to there. The entries to access are 781 organizational units and 5156 employees.

(2) Program2

This is the program that is assumed to directly find an employee by the attribute. This program generates an employee name at random and acquires the data of the employee with the employee name as a key. The entries to access are 5000 employees, 5000 organizational units and 4,999 upper heads except the president.

#### 4.3.1.3 Measurement environment

We connect two following computers in 100BaseT and access from PC that the evaluation programs execute to the directory server in LDAP interface.

```
dn: cn=schema
changetype: modify
add: attributetypes
attributeTypes: ( addExtTelNumber-oid
 NAME 'addExtTelNumber'
 DESC 'User Defined Attribute'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 X-ORIGIN 'user defined' )
attributeTypes: ( addExtFaxTelNumber-oid
 NAME 'addExtFaxTelNumber'
 DESC 'User Defined Attribute'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
 SINGLE-VALUE
 X-ORIGIN 'user defined' )
attributeTypes: ( addCn-oid
 NAME 'addCn'
 DESC 'User Defined Attribute'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
 SINGLE-VALUE
 X-ORIGIN 'user defined' )
attributeTypes: ( addSn-oid
 NAME 'addSn'
 DESC 'User Defined Attribute'
 SYNTAX 1.3.6.1.4.1.1466.115.121.1.5
 SINGLE-VALUE
 X-ORIGIN 'user defined' )
```

Figure 6 : LDIF for Definition of Attribute Types.

```
dn: cn=schema
changetype: modify
add: objectclasses
objectClasses: ( addOrgPerson-oid
 NAME 'addOrgPerson'
 SUP organizationalPerson STRUCTURAL
 MUST ( addCn $ addSn )
 MAY ( addExtTelNumber
 $ addExtFaxTelNumber )
 X-ORIGIN 'user defined' )
```

Figure 7 : LDIF for Definition of Object classes.

(1)The PC that the evaluation programs execute
H/W CPU: Intel Pentium4 2.8GHz
Memory: 760MB, HDD: 35GB
S/W Java 1.4.2 06
Windows XP Professional

(2)The directory server
H/W CPU: Intel Xeon 3.2GHz
Memory: 2GB, HDD: 292GB
S/W SunONE Directory Server 5.2
Windows Server 2003

### 4.3.2 Result

Table 2 shows the measurement result. In Table 2, MMS is Mitsubishi Solution System, and DMP is Directory Management Program.

### 4.3.3 Discussion

We found that Directory Modeling Language has enough description ability of the application system, because with Directory Modeling Language, we was able to build DIT approximately same as Mitsubishi Solution System.

We found that Directory Access API is implemented by approximately same method as the API of Mitsubishi Solution System in the access of the directory, because about the number of the entries sent to PC, the difference for program1 is 781 and the difference for program2 is only 81.

We found that Directory Access API can work with practical performance, because the total of the execution times of Program1 and Program2 is same as Mitsubishi Solution System.

We show the evaluation for the execution times of both. The differences of Directory Management Program and Mitsubishi Solution System about the processing to influence performance are the follows.

- The APIs of Directory Management Program gets all attribute's values of the entry, but some of the APIs of Mitsubishi Solution System can get a part of the attribute's values. Therefore Directory Management Program becomes disadvantageous in the quantity of data sent to PC.
- When Mitsubishi Solution System acquires the entry, it performs the access of the directory without the transfer to check the contradiction of the directory structure. The check is unnecessary for Directory Management Program because this system creates the correct data with Directory Loading Compiler. Therefore Mitsubishi Solution System becomes disadvantageous in the number of the access of the directory.

In Program1, Mitsubishi Solution System uses the many APIs to get a part of the attribute's values of the entry for the implementation of this system. The influence surfaced in Program1.

Program2 sends many entries to PC. Mitsubishi Solution System performs many accesses of the directory to check the contradiction of the directory structure. The influence surfaced in Program2.

## 5 CONCLUSION

We found that this system is enough applicable and efficient. By using the result of this work, the improvement of the quality, the productivity and the maintenancibility can be expected.

We will implement OCL(Object Constraint Language) for the expansion of Directory Modeling Language in the future[11]. OCL is the function taken for the standard in UML2.0 and the language to describe the limitation and the query of the UML model.

Table 1 : Data Structure of Measurement

| organization | org units | members | Title |
|---|---|---|---|
| | 1 | 1 | president |
| head office | 5 | 5 | vice president |
| division | 25 | 25 | general manager |
| deaprtment | 125 | 125 | director |
| section | 625 | 5,000 | manager, staff |
| TOTAL | 781 | 5,156 | |

Table 2 : Result of Measurement

| Program | system | time (minutes) | num of entries sent to PC |
|---|---|---|---|
| Program1 | MMS | 20 | 19,842 |
| | DMP | 24 | 19,061 |
| Program2 | MMS | 54 | 40,081 |
| | DMP | 50 | 40,000 |

Now we have the most basic function of the acquisition of the entry such as the follows.

- **findByPrimaryKey()**
  find the entry by the primary key(RDN) in the class.
- **findAll()**
  find the all entry in the class.

When the search by the free search condition is necessary, it assumes that user oneself learn the implementation of the Directory Access API and realize it. In the application system, it is necessary to acquire the entry by the complex search condition of the attribute value etc. We will implement the search with the free search condition by OCL.

## REFERENCES

[1] K.Saotome, S.Kondo, A.Oonuma, T.Komiya, S.Sakai, T.Mizuno : "Modeling Language for LDAP and Automatic Production System of Directory Management Program", Information Processing Society of Japan, Database, Vol47, No.SIG13(TOD31), pp.28-39 (2006).

[2] Dragan Milicev : "Automatic Model Transformations Using Extended UML Object Diagrams in Modeling Environments", IEEE TRANSACTIONS ON SOFTWEAR ENGINEERING VOL.28 NO4, pp.413-431, 2002.

[3] Ludovic Apvrille, Jean-Pierre Court : "TURTLE A Real-Time UML Profile Supported by a Formal Validation Toolkit", IEEE TRANSACTIONS ON SOFTWEAR ENGINEERING VOL.30 NO7, pp.473-487, 2004.

[4] MDA(Model Driven Architecture),
UML Profile for EDOC, UML Profile for EAI.
http://www.omg.org/mda/specs.htm#Profiles

[5] ITU-T(International Telecommunication Union-Telecommunication Standardization Sector) . http://www.itu.int/ITU-T/
[6] IETF(Internet Engineering Task Force) . http://www.ietf.org/
[7] Tim Howes : "Understanding and Deploying Ldap Directory Services", Macmillan Technical Pub, 2003.
[8] OMG(Object Management Group). http://www.omg.org/
[9] UML(Unified Modeling Language). http://www.uml.org/
[10] Information Leakage Prevention Solution of Mitsubishi Electric Corporation. http://global.mitsubishielectric.com/bu/security/rd/rd07_02.html
[11] OCL(Object Constraint Language). http://www.omg.org/technology/documents/formal/ocl.htm

(Received October 17, 2008)
(Revised July 17, 2009)

**Kenji Saotome** received a BE from the Osaka University, Japan in 1979, and a Dr.Eng in Information Engineering from the Shizuoka University, Japan in 2008. From 1979 to 2007, he was with Mitsubishi Electric, Japan. Since 2004, he has been a professor of Hosei business school of innovation management. His current research areas include LDAP directory applications and single sign-on system. He is a member of the Information Processing Society of Japan.

**Seiichi Kondo** received the master's degree in information science from Kyoto University, Japan, in 1984. Currently, he is a senior engineer in Mitsubishi Electric Information Systems Corporation. His research interests include information security systems, identity management, and access control.

**Akihisa Onuma** received a degree from the Waseda University in 1986, currently, he is a engineer in Mitsubishi Electric Corporation, Japan. His current research areas include data modeling and single sign-on system.

**Takashi Komiya** received a MA from the SAGA University, Japan in 2000, currently,he is a senior engineer in Mitsubishi Electric Corporation, Japan. His current research areas include overall information security systems, especially identity management and authentication systems .

**Masahiko Ishino** received the master's degree in science and technology from Keio University in 1979 and received the Ph.D. degree in industrial science and engineering from graduate school of Science and technology of Shizuoka University, Japan, in 2007. In 1979, he joined Mitsubishi Electric Corp. Since 2009, he is Professor of Fukui University of Technology. Now, His research interests include management information system, industrial engineering, application system of data-mining, and information security system. He is a member of Information Processing Society of Japan, Japan Industrial Management Association, Japan Society for Management Information.

**Tsukasa Kudou** received the master's degree in Engineering from Hokkaido University in 1980 and received the Dr. degree in industrial science and engineering from Shizuoka University, Japan, in 2007. In 1980, he joined Mitsubishi Electric Corp. He was the researcher of parallel computer architecture and the engineer of business software. Since 2005 he is the engineer of Mitsubishi Electric Information Systems Corp. Now, his research interests include database application and system quality assurance. He is a member of IEIEC and Information Processing Society of Japan.

**Sanshiro Sakai** received the B.E., M.E and Ph.D degrees from Shizuoka University, Japan, in 1979, 1981 and 1984, respectively. Currently he is a professor in the Faculty of Informatics, Shizuoka University. His current research interests are in computer supported collaborative learning, programming language education environments and understanding of computer programs. He is a

member of Information Processing Society of Japan (IPSJ), Institute of Electronics, Information and Communication Engineers (IEICE), and Japanese Society for Information and Systems in Education (JSiSE).

**Tadanori Mizuno** received the B.E. degree in industrial engineering from the Nagoya Institute of Technology in 1968 and received the Ph.D. degree in computer science from Kyushu University, Japan, in 1987. In 1968, he joined Mitsubishi Electric Corp. Since 1993, he is a Professor of Shizuoka University, Japan. Now, he is a Professor of graduate school of Science and technology of Shizuoka University. His research interests include mobile computing, distributed computing, computer networks, broadcast communication and computing, and protocol engineering. He is a member of Information Processing Society of Japan, the institute of electronics, information and Communication Engineers, the IEEE Computer Society, ACM and Informatics Society.