

Translation from GDMO/ASN.1 to tML/Schema

Wenli Dong^{*1}

**1 Corresponding author* Institute of Software, Chinese Academy of Science, Beijing, China
wenli@iscas.ac.cn

Abstract

For interoperation of heterogeneous management systems and integration of existing legacy applications in telecommunication network management systems, a translation from GDMO/ASN.1 to a universal language is necessary. Inheriting the extensibility and flexibility of XML, as the application of XML in telecommunication, tML can be used in definition and publication of management interface information model in telecommunication network management regardless of its implementation platform and language. This paper resolves one of the key problems of integrating tML in telecommunication network management: the translation from GDMO/ASN.1 to tML/Schema. This paper studies the translation method from GDMO/ASN.1 to tML/Schema. The corresponding relationship between GDMO/ASN.1 and tML/Schema is analyzed in this paper. The translation process consisting of two steps, that is, the translation from ASN.1 to tML/Schema and the translation from GDMO template to tML/Schema, are discussed in this paper.

Keyword

Guidelines for Definition of Managed Objects, Abstract Syntax Notation One, telecommunication Mark-up Language

1. Introduction

Telecommunication network management systems are used to control and monitor the components of distributed systems, with the increasing complexity of network management systems, many different subsystems need to collaborate to complete a management task. Telecommunication network is distributed, dynamic, which consists of different vendors and developed over time. Thus telecommunication network management is a challenging task that requires the interoperation of heterogeneous management systems and integration of existing legacy applications.

Network management interface information modeling is an important research direction of telecommunication network management. International standardization organizations have proposed various standards/recommendations for defining network management interface. Among them, the most advanced standards/recommendations are the Common Management Information Service (CMIS) of Open Systems Interconnection (OSI) [1,2]. To define managed objects of interest to the Telecommunications Management Network for use in CMIP (Common Management Information Protocol) [3,4], Guidelines for Definition of Managed Objects (GDMO) is specified [5,6]. GDMO represents a hierarchy of managed objects and use ASN.1 (Abstract Syntax Notation One) [7,8] for syntax. GDMO is a standard for defining objects in a network in a consistent way. In order to enable the telecommunications industry to utilize emerging platform independent framework, a translation from GDMO/ASN.1 to a universal language is necessary.

As a universal language, XML (eXtensible Markup Language) messages follow a series of W3C standards and have been popular with various applications. XML includes XSL (XML Stylesheet Language), Xlink, XML Namespace, and XML Schema etc. XML expands the function and conception of database system. XML owns some obvious advantages: (1) Pure text. Almost every editing tool can be used to create and edit XML. (2) Opening. XML is specified by W3C and XML doesn't belong to any accompany. (3) Extensibility. As a markup language, the tags in XML aren't pre-defined and users can define tags according to requirements by themselves. (4) Self-description. XML consists of characters and tags, and computer can distinguish the content in document automatically independent of specified data explaining program by analyzing document information through tag. (5) Powerfully linking ability. XLink specification will let us create links between more than two points. Links can be constructed. And greater control over increasingly complex links is allowed. XML can support the ability to create links to databases (6) Separation of content and form. XSLT (XML Stylesheet Language Transformation) is the mechanism

to transform XML. By XSLT, XML data can be transformed from one style into another style, and the personalized display can be implemented through CSS or XSL

Recently, the application of XML technology in network management starts to get attention [9,10,11,12,13]. M.3030 [9] proposes tML (telecommunication Mark-up Language) that may be applied initially at the TMN (Telecommunication Management Network) [14,15,16] X Reference Point. Inheriting the advantages from XML, tML can be used to describe data formats on the X interface. M.3030 thinks that using tML in TMN is important for integrating applications with their Web-based systems. However, M.3030 doesn't provide way about how to describe the data. Application of XML Technology in electric power telecommunication equipment warning system is proposed in [10]. This paper thinks the translation from GDMO/ASN.1 to XML/Schema is an important step in the application of XML in electric power telecommunication equipment warning system. But it just gives a simple description about the translation, and the translation isn't detailed in this paper. Application XML (named tML in TMN with some restriction specified by M.3030) in TMN to provide a mechanism for integrating existing legacy applications regardless of their platforms or languages and integrate applications with their Web-based systems and interoperation of heterogeneous management systems is becoming more and more important. And one of the key problems of application of tML in TMN is the translation from GDML/ASN.1 to tML/Schema. Based on the research of GDMO based telecommunication network management interface information interface construction and tML technology, the translation from GDMO/ASN.1 to tML/Schema is proposed in this paper.

The rest of this paper is organized as follows. Section 2 briefly introduces GDMO and tML/Schema. The translation of GDMO to tML/Schema includes two steps. The first step is the translation between languages. The second step is the translation from template to element. Section 3 gives the translation method of ASN.1 to tML/Schema. Then the translation of GDMO template to tML/Schema is described in section 4. In section 5 an example is illustrated and the conclusion is drawn in section 6.

2. GDMO and tML/Schema

Managed resource is abstracted to managed object class (MOC) in telecommunication network management interface information model. The following nine GDMO template types are defined to describe object class in GDMO: managed object class

template, package template, parameter template, name binding template, attribute template, attribute group template, behavior template, action template, and notification template. Using these templates, GDMO can describe the attributes and behavior of various objects in detail.

The core of GDMO is managed objected class template. Managed objected class template defines the basic framework of managed object. In GDMO, besides managed objected class template, if necessary, attribute template, behavior template, action template, and notification template can be used to define object complementarily. The elements in template define the location of the class in inheritance tree, the behavior, and attribute of the class. At the highest level, the template for the Managed Object Class is:

```

<class-label> MANAGED OBJECT CLASS
[DERIVED FROM <class-label>;
]
[CHARACTERIZED BY <package-label>;
    [BEHEVIOUR <behavior-
definition-label>;
    ]
    [ATTRIBUTES <attribute-label>
propertylist [<parameter-label>;
    ]
    [ATTRIBUTE GROUP <group-
label>;
    ]
    [ACTIONS <action-label>
[<parameter-label>;
    ]
    [NOTIFICATIONS <notification-
label> [<parameter-label>;
    ]
    ]
]
[CONDITONAL PACKAGES <package-
label>
    PRESENT IF condition-
definition;
]

```

REGISTERED AS object-identifier;

GDMO is not derived from Abstract Syntax Notation One (ASN.1). However, by using ASN.1, definitions of data values in GDMO are detailed

tML is the application of XML in telecommunication network management. tML Schema is used as a template for defining tML-based interface information.

tML/Schemas express shared vocabularies and allow computers to carry out commands invoked by user. tML/Schema can use a reference to an element or an attribute similar to cloning an object. They provide

means for defining the structures, contents and semantics of tML/Schema documents, and define the cardinality of an element (e.g., the number of its possible occurrences). This element may or may not have attributes, text, children, and sub-elements. A sequence of sub-elements shall be considered if the element contains sub-elements. The definition of the target namespace and several default options can also be pre-defined.

According to the Schema syntax specifications, the containment relationship among these constructs can be illustrated as in [17]. The typical structure can be shown as following:

```
<?xml version="1.0" encoding="GB2312"?>
<xsd:schema
targetNamespace="http://latest/series/nmcpnmc.doc#
CircuitSheet"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://latest/series/nmcpnmc.doc#CircuitShee
t" elementFormDefault="qualified"
attributeFormDefault="unqualified">d
    <xsd:element name="electrical
information">
        <xsd:complexType
name="management information">
            <xsd:element
name="state" type="xsd:string"/>
            <xsd:element
name="stamp" type="xsd:string"/>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

MOC template is the abstract of a group of managed object with same characteristic, and describes the characteristic of the managed object. tML/Schema can be used to describe the function of the object in distributed system. For the translation from GDMO/ASN.1 to tML/Schema, the translation from template to tML/Schema element is the most important step in translation from GDMO/ASN.1 to tML/Schema .

The technology of translation from GDMO-MOC to tML/Schema element is analyzed as follows.

Following three regulations should be in consideration of tML/Schema based TMN managed object (described by tML/Schema):

Firstly, the managed object described in GDMO with information related with CMIS (Common Management Information Service). Being the service provided by CMISE (Common Management Information Service Element), the Common management information service (CMIS) is a service that may be employed by network elements for

network management. Thus, all managed object should support the function of CMIS request and response. Secondly, the object described by tML/Schmea should conform to the grammar rules specified by W3C. The meaning, usage and relationship of the components e.g. data type, element, element content, attribute and attribute value are restricted and documented by tML/Schema components. Thirdly, in TMN, the managed object should inherit from M3100::top, from which the basic attributes such as object identifier is defined.

Then, the generation process of tML/Schema based TMN managed object is formed:

Firstly, the tML::rootElement is defined as the root element of Schema describing managed object, and implement the basic function; secondly, the ManagedObject inheriting form tML::rootElement is defined, and implement the basic function of CMIS; at last, the M3100::top inheriting from ManagedObject is defined, and implement basic attribute of managed object; other managed object class can inherit from ManagedObject, and implement specific function. Figure 1 give the inheritance system described above.

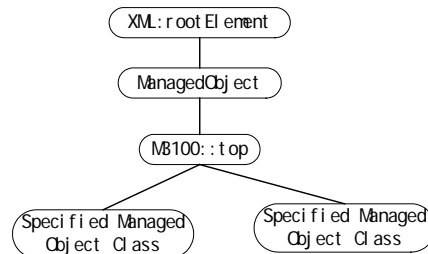


Figure 1. Inheritance system describing managed object class

What's more, the error mechanism is provided in TMN. In TMN, a series of basic error identifiers are provided through ROSE (Remote Operations Service Element) [18,19], among which every error parameter is defined through parameter template, actually, which is implemented by "ANY DEFINED BY" clause. tML/Schema is related with tML Namespace tightly, and this relation simplifies the creation of the Schema documentations using more than one namespaces. So, the errors such as visit rejection, class instance conflict, and the failure to obtain list defined in X.711[20] can be translated into tML/Schema element, which can be defined in a Schema document, and be classified by the relationship between element and element content, that can be shown as following:

```
<xsd:element name=" CMIP_ACTION_ERROS">
<xsd:complexType>
    <xsd:sequence>
```

```

        <xsd:element
name="OSIMgmt_AccessDenied " type="OSIMgmt
_AccessDenied_EORROTYPE "/>
        <xsd:element
name="OSIMgmt_ClassInstanceConflict"
type="OSIMgmt_ClassInstanceConflict_ EORRO-
TYPE "/>
        .....
        <xsd:sequence/>
        <xsd:complexType/>
        <xsd:element/>
        <xsd:element name="
CMIP_ATTIBUTE_ERROS">
        <xsd:complexType/>
        <xsd:sequence>
        <xsd:element
name="OSIMgmt_GetListErrors"
type="OSIMgmt_GetListErrors_EORROTYPE "/>
        <xsd:element
name="NoSuchObjectInstances"
type="NoSuchObjectInstances_EORROTYPE"/>
        .....
        <xsd:sequence/>
        <xsd:complexType/>
        <xsd:element/>

```

GDMO template is described by ASN.1, and tML/Schema is described by tML. So, the translation of GDMO/ASN.1 to tML/Schema should include two steps. The first step is the translation from ASN.1 to tML/Schema. The second step is the translation from GDMO template to tML/Schema. The second step is based on the first step.

3. Translation from ASN.1 to tML/Schema

Basic type mapping of ASN.1 and tML/Schema is shown in table 1.

Table 1. Basic type mapping of ASN.1 and tML/Schema

Schema	ASN.1
string	BITSTRING/OCTEDSTRING
normalizedString	
token byte	
unsignedByte	
base64Binary	
hexBinary time	
dateTime duration	
date gMonth gYear	
gYearMonth gDay	
gMonthDay Name	
QName NCName	
anyURI language	

Integer	INTEGER
PositiveInteger	
nonNegativeInteger	
NegativeInteger	
nonPositiveInteger	
Int	
UnsignedInt	
Long	
UnsignedLong	
Short	
UnsignedShort	
0	NULL
Boolean	Boolean

For more complex type, we have the following Handling of complex type:

A. CHOICE

The purpose of the CHOICE construct in ASN.1 is to specify a list of possible types that a data element may belong to. The ASN.1 notation does not strictly define any semantics for determining an appropriate type; this type can be described by choice type in Schema and vice versa.

B. ANY

The ANY type is not restricted to a specific type just like the any type in Schema.

C. SEQUENCE SEQUENCE-OF SET SET-OF

In ASN.1, SEQUENCE SEQUENCE-OF SET SET-OF is used to specify an array of elements. The type of the elements can be either simple or complex (constructed). A list of elements may be of fixed, variable, or unconstrained size. Sequence in tML/Schema is kind of type specifying an array of elements too. So the two types can be mapped correspondingly.

Based on the relationship of ASN.1 and tML/Schema, the following conversion rules are generated:

Rule 1: ASN.1 type is corresponding with Schema mark, and the scope is defined in the facet of that type.

Rule 2: The Schema mark comes from the generator, but, if there are context, and the type is assigned full name, the full name will replace the identifier in the generator.

Rule 3: For set, choice, and sequence in ASN.1, when there are unique name, the Schema mark for their instance is the identifier in the generator followed by number corresponding to its occurrence order in the Schema, and the number is started from '1'.

Based on the above rules, a translation example can be illustrated as follows.

A Record whose type is SEQUENCE in ASN.1 is described as following:

```
Record ::= SEQUENCE {
    number Number,
    name Name }
Number ::= INTERGER
Name ::= OCTET STRING
```

The corresponding tML/Schema can be shown as following:

```
<?xml version="1.0" encoding="UTF-16"?>
<xsd:schema
targetNamespace="urn:int.itu/tML/SICSExample"
xmlns:tML="urn:int.itu/tML/SICSExample"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
version="1.0">
<xsd:element name="record">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element
name="number" type="xsd:interger"/>
            <xsd:element
name="Name:" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
</xsd:schema>
```

The translation system consists of two parts mainly: ASN.1-parser and tML/Schema-parser. After the ASN.1 document is input, the ASN.1 performs lexical analysis, and syntactic analysis to verify the grammar correctness of this ASN.1 document. Then the tML/Schema-parser will extract information from ASN.1 parsing result, then tML/Schema is generated.

tML/Schema is made up of markup language including element declaration, attribute declaration etc. There are two primary ways that applications can pull information from a tML/Schema-parser parser: Simple API for tML/Schema (SAX): as an event stream or Document Object Model (DOM): as a tree. SAX is the standard for event-stream support. It provides a callback API so that an application can be notified of tML/Schema elements as they are parsed. The SAX parser reports events such as the start and end of elements to the application as it walks over the document. Because the SAX parser reports events as it

visits different parts of the document, it does not have to build any internal structure. DOM provides an API for representing the logical structure of documents and follows a tree construct stored in memory. The SAX parser is chosen in the translation from ASN.1 to tML/Schema. Because this translation is accomplished by parsing once, and the SAX parser can meet the parsing requirements, and the SAX parser can implement the translation from ASN.1 to XML/Schema on line, not occupying large memory.

4. Translation from GDMO Template to tML/Schema

In TMN, various managed objects templates are defined in GDMO, and the Managed object class template in GDMO can be illustrated as figure 2

In tML/Schema, the validating structure, content and restriction are described by Schema. The sharable vocabulary, the tML/Schema document structure based on the vocabulary and the relationship between the tML/Schema documents are defined. tML/Schema consists of data type definition and element declaration to validate the well-form elements and attributes. tML/Schema is the set of Schema component, which is classified into three groups:

1. basic component
 - Simple type definitions
 - Complex type definitions
 - Attribute declarations
 - Element declarations
2. component
 - Attribute group definitions
 - Identity constraint definitions
 - Model group definitions
 - Notation declaration
3. help component
 - Annotations
 - Model groups
 - Particles
 - Wildcards
 - AttributeUses

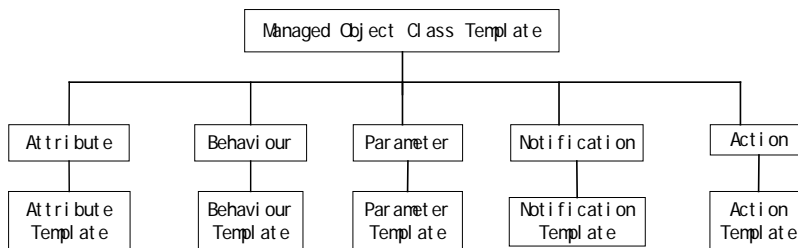


Figure 2. Managed object class template in GDMO

Based on relationship between GDMO templates and Schema structure, the mapping relationship between GDMO template and the tML/Schema component in Schema are described as following:

Based on inheritance relationship, managed object class is translated into a root element or a non-root element;

Package template is the container of behavior template, attribute template, attribute group template, notification template, and action template. So the translation of package template can be transformed into the templates contained in the package templates, actually, in translation, we just replace package template with its content.

Attribute group template consists of a group of attributes, which is translated into an element whose type is complex type, and each attribute in attribute group template can be translated into a sub-element of the element translating from that attribute group;

An attribute of the managed object class is translated into a sub-element of the element translating from that managed object class;

Behavior template is translated into a sub-element of the element translating from that managed object class, and if there are more than one responses, it can be translated into a element whose type is complex type;

Each notification in managed object class is translated into one or more elements;

The action template in managed object describes the function of that template, and it can be translated into the annotation in tML/Schema

Parameter template defines parameter syntaxes that can be included by Package, Attribute, Action, and Notification templates. Parameter template be mapped to the sub-element of the element with the data type describing the parameter of the behavior. Parameter template can describe the dependence between managed objects, and this dependence relationship can be described by name space.

The translation described above can be shown in figure 3.

The mapping relationship illustrated above can be analyzed as following:

(1) Relationship between element and managed object

In tML/Schema, an element is the description of its attribute and the relationship between this element and its sub-elements. When a managed object contains some behaviors, a customer can describe the behaviors through element type definition. Because an element may not describe characteristics of that object completely, a managed object can be translated into one or more elements.

(2) Relationship between element and behavior

The relationship between managed object and element is based on behavior. The translation is mainly to create the mapping between element and behavior. In tML/Schema, element is the smallest independent unit provided for a customer. In GDMO, behavior is a kind of management operation invoked by object. So, the element can be mapped to the behavior correspondingly and so does the behavior.

(3) Relationship between element attribute in tML/Schema and attribute in managed object

In TMN, attribute is the description of behavior characteristics of managed object, and it is a necessary component in managed object definition. By sending request to managed object, the attribute value can be read, written, and modified. But in tML/Schema, the attribute is not the necessary component of element definition. Although it is called attribute too, attribute defined in tML/Schema is different from the attribute of managed object. They can not be mapped between each other simply. Attribute in managed object is equal to a series of function reading or writing managed object, and describing the data transporting between client and server. When attribute in managed object is transported between client and server, this attribute can be translated into the attribute of the element, otherwise, it needn't be translated. The attribute in managed object needn't exist in the corresponding element type definition of managed object.

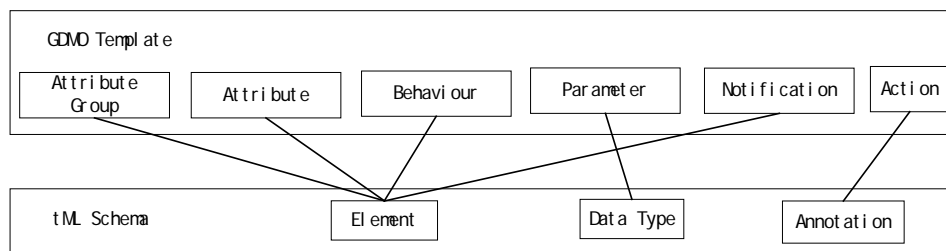


Figure 3. Mapping between main components in GDMO templates and Schema

(4) Relationship between data type of the element and the parameter in managed object

Not only the behavior in managed object but also the element in tML/Schema corresponding with that behavior needs the support of the parameter definition. If a behavior in managed object is mapped to an element defined in tML/Schema, the parameter of the behavior in the managed object should be mapped to the sub-element of the element with the data type describing the parameter of the behavior.

(5) Relationship between action and annotation

In the definition of GDMO, action description is necessary, and it is related with the actual implementation process and implementation method of GDMO object service. For element definition in tML/Schema, annotation is a complementary illustration, and it can provide corresponding description for implementation.

(6) Relationship between notification in GDMO and element of tML/Schema

Notification in GDMO is the report from managed object to client or other object through interface. In fact, this report is a kind of operation, and it can be seen as the response to a request of a manager. The notification operation differs from non-notification operation. In notification operation, notification and request can be asynchronous, and owns multi-response characteristic. For example, request can be sent in initial stage of non notification operation invocation, but notification may exist in the whole life cycle of object existence. So, notification can be translated into an element in tML/Schema, and also can be translated into more than one responding elements in tML/Schema corresponding to its various activities in the whole life cycle of the object. When translating from a behavior or notification in GDMO to an element in tML/Schema, if the "WITH REPLY SYNTAX" sub clause is contained in this behavior, necessary comment in tML/Schema is generated to describe the multi-response treatment.

From the discussion above, we can say, a GDMO template can be translated into one or more tML/Schema elements, including one root element, and the element describes multi-response mechanism notification. If there is inheritance relationship between managed objects, there is inheritance relationship between elements in corresponding Schema document. And the inheritance relationship is corresponding with the inheritance relationship of managed object class. When translating GDMO template to element in tML/Schema, the translation method of attribute, behavior, notification in conditional package is the same with that of necessary package, and they can be distinguished by comment.

5. Case Study

A behavior template of creating file and corresponding tML/Schema can be shown as follows.

The behavior template of creating file is described as following:

```
createFile ACTION
BEHEVIOUR
createFileBhv BEHAVIOUR
DEFINE AS "Receipt of this action causes the creation
of a file.....";
MODE CONFIRMED;
WITH INFORMATION SYNTAX
ASN1 DefineTypesModule.CreateFile Argument;
WITH REPLY SYNTAX
ASN1 DefineTypesModule.CreateFile Response;
REGISTERED AS {eDRAction 1};
Corresponding tML/Schema is defined as following:
<xsd:element
name="createFile_MR:CMIPMultipleReply">
<xsd:complexType>
<xsd:element name="next">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="request_id"
type="long"/>
<xsd:complexType>
<xsd:attribute
name="type" type="xsd:string" use="required"
fixed="in"/>
<xsd:complexType/>
<xsd:element/>
<xsd:element
name="next_element"
type="ASN1DefineTypesModule::CreatFileResponseT
ype"/>
<xsd:complexType>
<xsd:attribute
name="type" type="xsd:string" use="required"
fixed="in"/>
<xsd:complexType/>
<xsd:element/>
<xsd:sequence/>
<xsd:complexType/>
<xsd:element/>
<xsd:complexType/>
<xsd:element/>
<xsd:element
name="createFile_MRPull:CMIPMultipleReply">
<xsd:complexType>
<xsd:element name="CMIPMultipleReply">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="pull_next">
```

```

        <xsd:complexType>
        <xsd:sequence>
        <xsd:element          name="request_id"
type="long"/>
                <xsd:complexType>
                    <xsd:attribute
name="type"    type="xsd:string"    use="required"
fixed="out"/>
                <xsd:complexType/>
        <xsd:element/>
        <xsd:element
name="next_element"
type="ASN1DefineTypesModule::CreatFileResponseT
ype"/>
                <xsd:complexType>
                    <xsd:attribute
name="type"    type="xsd:string"    use="required"
fixed="out"/>
                <xsd:complexType/>
        <xsd:element/>
        <xsd:complexType/>
        <xsd:sequence/>
        <xsd:element/>
        <xsd:element name="try_next">
        <xsd:complexType>
        <xsd:sequence>
        <xsd:element          name="request_id"
type="long"/>
                <xsd:complexType>
                    <xsd:attribute
name="type"    type="xsd:string"    use="required"
fixed="out"/>
                <xsd:complexType/>
        <xsd:element/>
        <xsd:element
name="next_element"
type="ASN1DefineTypesModule::CreatFileResponseT
ype"/>
                <xsd:complexType>
                    <xsd:attribute
name="type"    type="xsd:string"    use="required"
fixed="out"/>
                <xsd:complexType/>
        <xsd:element/>
        <xsd:complexType/>
        <xsd:sequence/>
        <xsd:element/>
        <xsd:sequence/>
<xsd:complexType/>
        <xsd:element/>

```

6. Conclusion

In integrating tML into TMN, the translation of GDMO template to tML/Schema is one of the most

important steps. In TMN, managed object is described by GDMO template, and ASN.1 is adopted to describe GDMO template. When adopting tML/Schema modeling, the relationship of object described is implemented through element, and the element is described by tML/Schema. By creating element of TMN managed object in tML/Schema, a distributed soft implementation technology, tML, is adopted to implement distributed management of TMN. Because the translation is related with descriptive language, the translation includes two steps, the first step is the translation from ASN.1 to tML/Schema, and the second step is the translation from GDMO template to tML/Schema. This paper emphasizes the mapping between data type of ASN.1 and tML/Schema, the translation rules of ASN.1 and Schema, and the translation relationship from GDMO template to tML/Schema, at last, a translating example is given to illustrate the translation method. The translation from GDMO/ASN.1 to tML/Schema is based on the analysis of the component relationship and corresponding relationship of GDMO/ASN.1 and tML/Schema, and the translation is similar to depth traversal, the translation method proposed in this paper is practical. The tool based on the translation method proposed by this paper has been designed and developed and is used in Web-based telecommunication network management system, and the application has proved the translation method is practical.

7. Acknowledgement

This work is supported by the National High-Tech Research and Development Plan of China under Grant 2007AA001Z190).

8. References

- [1] ITU X.710, Common Management Information Service definition for ITU applications, 1991
- [2] ISO/IEC 9595, Information Technology - Open Systems Interconnection - Common Management Information Service definition, 1991.
- [3] ITU X.710 ISO/IEC 9595, Common Management Information Service Definition.
- [4] ITU X.711 ISO/IEC 9596-1, Common Management Information Protocol 1 Specification
- [5] ITU X.721 ISO/IEC 10165-2, Information Technology - Open Systems Interconnection - Structure of Management Information - Part 2: Definition of Management Information

- [6] ITU X.722 ISO/IEC 10165-4, Information Technology - Open Systems Interconnection - Structure of Management Information - Part 4: Guidelines for the Definition of Managed Objects (GDMO)
- [7] ITU X.208 ISO/IEC 8824, Specification of Abstract Syntax Notation One (ASN.1)
- [8] ITU X.209 ISO/IEC 8825, Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)
- [9] ITU M.3030, Telecommunications Markup Language (tML) framework, August 2002.
- [10] Yang Xu, Huisheng Gao, Jie Ding. Application of XML Technology in Electric Power Telecommunication Equipment Warning System. <http://www.dt365.com/Article/ShowArticle.asp?ArticleID=3318>. 2006.
- [11] Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, Nikita Schmidt, Franco Zambonelli. A survey of autonomic communications. *ACM Transactions on Autonomous and Adaptive Systems*. 2006, 1(2), pp:223-259.
- [12] In-Gyu Kim, Doo-Hwan Bae, Jang-Eui Hong. A component composition model providing dynamic, flexible, and hierarchical composition of components for supporting software evolution. *Journal of Systems and Software*. 2007, 80(11), pp:1797-1816.
- [13] Xun Zhang, Jiahai Yang, Jilong Wang, Hui Zhang. XML-based Network Configuration Management System. *Computer Engineering*. 2008, 34(03), pp: 127-129.
- [14] ISO/ITU M.3010, Maintenance: Telecommunications Network. Principles for a Telecommunications Management Network, October 1992
- [15] ISO/ITU M.3020, Maintenance: Telecommunications Network. TMN Interface Specification Methodology, October 1992.
- [16] ISO/ITU M.3180, Maintenance: Telecommunications Network. Catalogue of TMN Management Information, October 1992.
- [17] DONG Wenli, MENG Luoming. tML Schema Based ICS Proforma and Generation Method. *Chinese Journal of Electronics*, 2005, 14(4), pp. 681-685.
- [18] ITU X.219, Remote Operations: Model, Notation, and Service Definitions, 1988
- [19] ITU X.229, Remote Operations: Protocol Specification, 1988
- [20] ITU-T Rec.X.711. Revision to include ASN.1:1997, 2000-02.