**1. Workshop on Databases and Software Engineering**

# 2. An XML-based Software Requirement Document for System Development

*This paper proposes an approach to change the procedure of the system development in the past by introducing the technique of XML. Under the framework, the developers write the results from requirements analysis down as a XML-based software requirement document in the initial period of the system development, and expect to turn software requirement documents into one of the system operation components through the machine readable XML documents. In other words, the developers only need to embed software requirement document into program in the implementation phase and then systems will operate according to the definition of software requirement documents.*

*This approach not only makes user' requirements come true completely, but also raises the convenience of software maintenance in the future. In view of it, if the systems need to be altered because of changes in business rules, we will just need to modify XML-based software requirement documents and then the systems will update the interior functions immediately.*

4. Author
Yu-Liang Chi,
Assistant Professor
Dept. of MIS, Chung Yuan Christian University
No.22 Pu-Jen,Pu-chung Li, Chung-Li (32023), Taiwan, R.O.C
Email: maxchi@cycu.edu.tw
TEL : 886-3-4563171 ext.5408
FAX : 886-3-466-0094

Kuo- Ming Lo
Dept. of MIS, Chung Yuan Christian University
Email: callmes@mail2000.com.tw
TEL : 886-3-4563171 ext.5408
FAX : 886-3-466-0094

5. Contact Author: Kuo- Ming Lo,

6. Keywords: Software Requirement, XML-based Software Requirement, and Application System Development

# An XML-based Software Requirement Document for System Development

**Yu-Liang Chi and Kuo- Ming Lo**
Dept. of MIS, Chung Yuan Christian University, Taiwan, R.O.C.
maxchi@mis.cycu.edu.tw, callmes@mail2000.com.tw

## Abstract

*This paper proposes an approach to change the procedure of the system development in the past by introducing the technique of XML. Under the framework, the developers write the results from requirements analysis down as a XML-based software requirement document in the initial period of the system development, and expect to turn software requirement documents into one of the system operation components through the machine readable XML documents. In other words, the developers only need to embed software requirement document into program in the implementation phase and then systems will operate according to the definition of software requirement documents.*

*This approach not only makes user' requirements come true completely, but also raises the convenience of software maintenance in the future. In view of it, if the systems need to be altered because of changes in business rules, we will just need to modify XML-based software requirement documents and then the systems will update the interior functions immediately.*

## 1. Introduction:

As business activities change with each passing day and business operating environments become more complex, the business model must be changed dynamically and flexibly. The enterprises can change their operating model to follow the consumers' demand easily, but the information system can't be change easily. Consequently, the enterprises hope the information system can possess the abilities to suit for the changeful information requirement and the shortest developmental time flexibly.

In the period of the information system development and maintenance, business rules in the information system are often modified. "Business Rules", it means the real rules and procedure used to operate the business comprehensively. Business rules define what needs to do in the program on software level. It conducts the program how to solve the problem that it aimed at [1]. But most system developmental ways write the business rule in the codes [2]. When the rules need to be changed by following the business policies and benefits in the future, the information system won't satisfy with what the users need. As a result, this study brings up a new idea. It is to draw out the business rules that the system operation needs from the system and save them as an independent file. In other words, we can make the system read the business rules to replace writing the business rules in the system internal part (codes). When the business must change the business rules because of external factors, users just need to alter the file they need. In fact, software requirement documents record the business rules that the system operation needs. The business rules are compiled via conversing with customers in the initial period of system development. Then, in the implementation phase, the programmers write them down as the system (codes) repeatedly. Therefore, if we can make the system read software requirement documents directly, it will reach the idea of causing the business rules and the system (codes) apart.

How do we let the system read the software requirement documents? XML can resolve the problem depending on the present techniques. For the reason, this study plans to write the software requirement documents by using XML made by W3C, and expects to let the system possess the ability of reading the software requirement documents through the XML characteristics of human-machine readable.

In view of the above-mentioned mentioned motivation, this study will improve the software requirement documents in the requirement engineering. The major work is to use XML as the language to describe the software requirement. The method not only enables the system to read and implement requirements that it defines but also to replace the restriction that the software requirement documents just can regard as human reading in the past, and then achieves the major four purposes of this study.

(1) Make business rules and programming logic divided.

(2) Software requirement documents can be read and implemented by machines.

(3) Software requirements specification pattern can be reused repeatedly.

(4) Software requirement documents can change the system functions.

## 2. Background

The purpose of the software requirements specification is to record the user's requirement for the system software. It is the bridge between developers and customers, the basis of the software development and maintenance, and the criterion of gauging the system quality during the system check. Hence, the software requirements specification is the most important one of all the system documents. [3]

Software requirement documents (are also named as software requirement specifications, SRS, sometimes) record the requirement that the system need to possess, including requirement definitions, requirement specifications and detailed software specifications [4]. Davis (1990) summarizes, compares with the different methods of describing requirement specifications and classifies them for five sections:[5]

(1) Structural natural language: such as software requirement document outlines announced by Institute for Information Industry.

(2) Design descriptive language: such as PDL

(3) Requirement specification languages: such as PSL, PSA and RSL.

(4) Graphical Notations: such as SADT.

(5) Mathematical Specifications.

## 3. XML-based Software Requirement Document

XML-based software requirement documents brought up by this study are the composite of the three specifications. XML-based software requirement documents are the executable document using XML to describe the above-mentioned specifications. It changes the role of software requirement documents in the former software evolutional process and turns the software requirement documents into the data resources of the system operation. The data resources are such as the functions for the system operation, business rules and the procedure. The XML-based software requirement documents can not only provides for the readers from the different levels but also make the system read and implement the content of requirement documents.

### 3.1 XML-based Software Requirement Document for System Development

The evolutional process of XML-based software requirement specifications for system development can be divided into four stages: requirement analysis, requirement document design, programming design and system maintenance.

**(1) Requirement Analysis**

It includes requirement collection, requirement definition, requirement specification and software specification in requirement analysis phase. It is different from the present way. In this phase, XML is used as the language for

analysis, and all the documents from every step are XML document. Consequently when system analysts set up the requirement or software, they can enter into the repository first to seek for the finished requirement specifications and then treat them as a part of new specification so as to obtain the most benefits by reusing software requirement specification levels.

**(2) Requirement Document Design**

In requirement document design phase, the main is to compile XML documents (requirement definition, requirement specification and software specification) from the first phase, and portray as the human-machine readable XML-based software requirement documents.

**(3) Programming design**

In this phase, the main is to parse in accordance with the Software requirement document. Programmers make program can read software requirement documents and implement dynamically to replace keying the requirement document definition in the codes. In other words, the system will do everything of whatever system architecture or business rules that requirement documents define.

**(4) System Maintenance**

The main is to update the system by altering software requirement documents to substitute for modifying the internal codes. Hence, people who maintain the system don't need to be familiar with the codes.

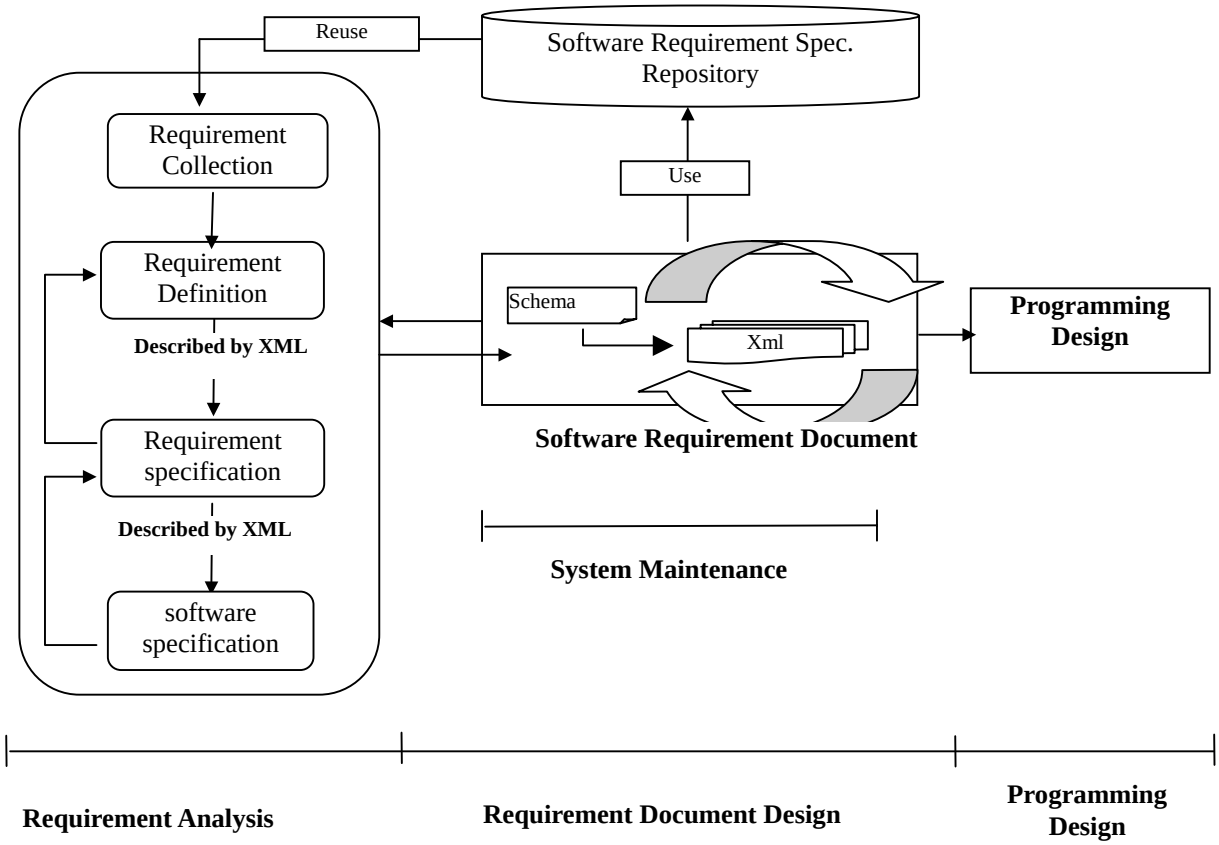



**Figure 1 : XML-based Software Requirement Document for System Development**

### 3.2 Constructing XML-based software requirement documents

The processes for writing the XML-based software requirement documents are as follows:

| | |
|---|---|
| **Step1 :** using XML to describe the flowchart of each function. | **Step6 :** using XML to describe each process. |
| **Step2 :** using XML to describe the data glossary. | **Step7 :** using XML to describe each sub-process. |
| **Step3 :** using XML to describe the system architecture. | **Step8 :** using XML to describe the procedure of each process. |
| **Step4 :** using XML to describe each model. | **Step9 :** using XML to describe the rules of each procedure. |
| **Step5 :** using XML to describe database. | **Step10 :** integrating XML-based software requirement |

Each step could be subdivided into two steps：

**(1) Definition of schema**

Schemas are used as stipulating the arrangement of tags and the format of content. We can save for the defined XML schemas in a particular repository. When we want to develop a new system, we can consult the pervious schemas to write XML documents and then shorten making time. Besides we can write schemas by ourselves, we can use others' schemas to define the structure of XML documents. Hence, in order to write XML software requirement documents in a uniform format, this research will define schemas in the above-mentioned nine steps so as to establish a XML-based software requirement.

**(2) Writing the real data**

After defining XML schemas, we can fill the real data in one by one according to the defined document structure.

### 3.3 Using XML-parser to read XML-based software requirement documents.

After establishing XML software requirement documents, it comes to the program design because XML documents have the machine-readable character, the information system can read XML-based software requirement documents and operate in terms of the content. How do we design a system that can read XML documents? Due to we design the system by using Java, we will combine with JAXP provided by Sum and take it for the parse tool of XML documents.

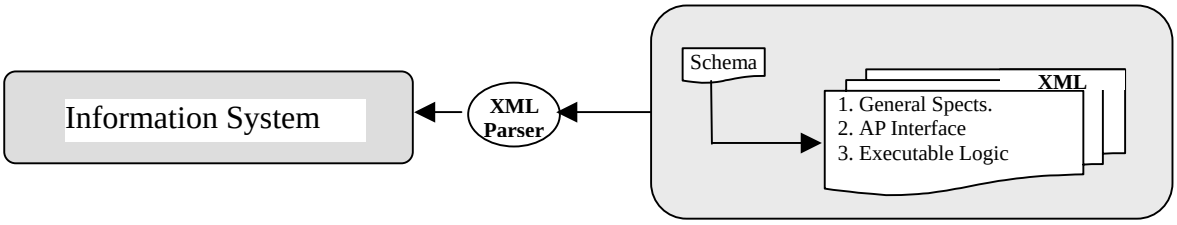



**Figure2 : Using XML-parser to read XML-based software requirement documents.**

### 3.4 Present XML-based software requirement documents by using XSL

XSL is the abbreviation of "extensible stylesheet language". Its main function is to set the appearing way of XML documents. Besides the documental appearance, it can select the appearing content from XML documental data. Hence, it can change the appearing form depending on users' requirement. For example, it can change the appearing order of data. Through the different XSL, it can show the whole content in the same documents. Some content or turning it into the form of HTML. Consequently, we can make the software requirement documents suit for executive officer or programmers in accordance with the different XSL not to make two distinct documents.

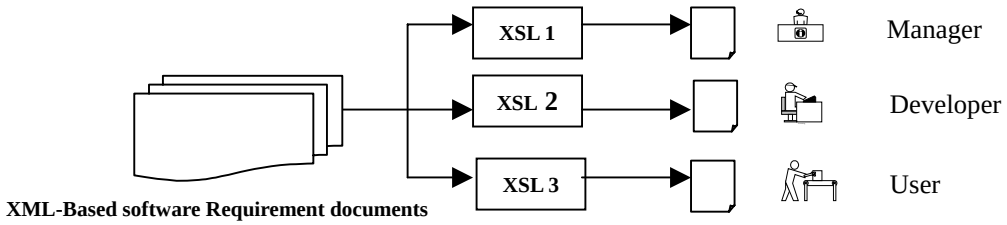



**Figure3 : Present XML-based software requirement documents by using XSL**

## 4. XML-based software requirement specification repository

### 4.1 XML-based software requirement spec. Repository

XML-based Software requirement specification repository is to store the relative specifications of the drafted software system such as the system architecture specifications, the descriptive specifications of flowcharts and the business logic specifications. The purpose is to help the developers get data they need by way of the feature and advantage of XML in requirement analysis phase. We expect to define a complete software requirement documents via a standard requirement specifications. "XML-based Software requirement specification repository" includes the following seven components of XML schema now.

(1) Schema of flowcharts
(2) Schema of data glossary
(3) Schema of the system architecture
(4) Schema of database description
(5) Schema of the mutual relation between system models and database
(6) Schema of process's procedure.
(7) Schema of business rules.

The system developers can use the service provided by XML-based software requirement specification repository through browser and downloading XML schemas for writing requirement specifications of software requirement documents.

### 4.2 . Introduction of the internal components in repository

**(1) Schema of flow charts**

It describes the flow of the system operation like the order process, purchase process…etc.

**(2) Schema of data glossary**

It is to describe the blueprint for the system operation such as the blueprints of order, sending bills, bill of asking for outlay.

**(3) Schema of the system architecture**

It describes the whole functional structure in the system such as models, process functions and operations included in the system.

**(4) Schema of database description**

It describes the data repository so-called database, such as the basic data of customers order data product data…etc.

**(5) Schema of the mutual relation between system models and database**

It describes the interactive relation between models and database.

**(6) Schema of business rules**

It describes the business rules included in the system.

## 5.Example

We describe the developmental methodology according to XML software requirement documents by using a business process system named “EZIT” for short in the following sections.

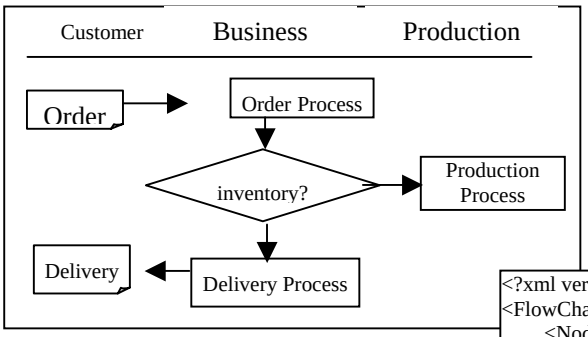
### 5.1. Writing XML-based software requirement documents

EZIT is a trade company that manages the commerce of the cars and motorcycles component. It has two plants and manufactures some parts of components by itself. The scope covered in the system includes three parts: “Sales Management Model”, “Operations Management Model” and “Purchase Management Model”. In the analytical process, because many steps and principles are similar, we select “Sales Management Model” for an illustration.

Sales Management Model can be divided into ten steps for writing XML-based software requirement documents. The application for nine steps is as follows.

**Step1: using XML describes the operative flowchart of each function**

In accordance with the outcome from requirement analysis, “sales management” can be divided into order process, sending process, and returned process. We completely imitate “Schemas of flowcharts description” designed to describe its flowchart. The flowcharts (order process, sending process) are shown as follows.




**Described By XML**

```
<?xml version="1.0" encoding="big5"?>
<FlowChart  Name="Order-Delivery Process">
    <Node  Name="Order"  Type="Document">
        <To  Nodename="Order Process"> </To>
        <Actioner  Name="Customer"></Actioner>
    </Node>
    <Node  Name="Order Process"  Type="Process">
        <From  Nodename="Customer"> </From>
        <To  Nodename="Inventory Check"> </To>
        <Actioner  Name="Business"></Actioner>
    </Node>
    <Node  Name="Inventory"  Type="check">
        <From  Nodename="Order Process"> </From>
        <To  Flag="Y"  Nodename="Delivery Process"> </To>
        <To  Flag="Y"  Nodename="Production Process"> </To>
        <Actioner  Name="Business"></Actioner>
    </Node>
    <Node  Name="Delivery Process"  Type="Process">
        <From  Nodename="Invertory Check"> </From>
        <To  Nodename="Delivery"> </To>
        <Actioner  Name="Business"></Actioner>
    </Node>
    <Node  Name="Delivery"  Type="Document">
        <From  Nodename="Delivery Process"> </From>
        <Actioner  Name="Customer"></Actioner>
    </Node>
</FlowChart >
```

**Figure4 : using XML describes the operative flowchart of each function**

**Step2: using XML to describe the data glossary**

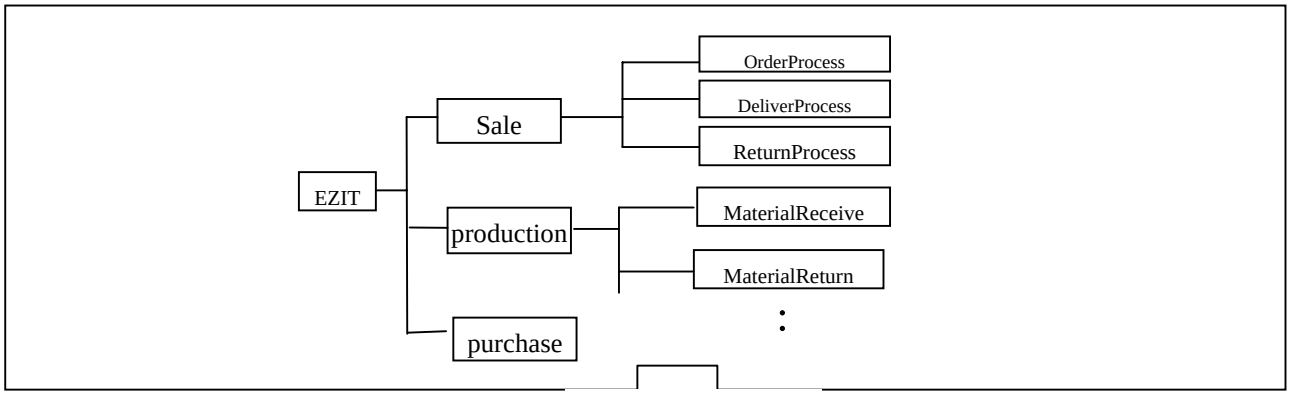
The orders in the sending process should have a fixed form of the blueprint. The order blueprint can view the present order forms as the basis. Hence, we can completely imitate “schemas of the data glossary description” designed to portray the order blueprint as XML It is shown as follows:

```
<DataVocabulary>
<Vocabulary  name="Order Data Glossary">
<Attribute  Name="Name"  NO="A"  Length="20"  Style="C"  Example="xxx">
<Attribute  Name="Address"  NO="B"  Length="40"  Style="C"  Example="Taipei">
<Attribute  Name="TEL"  NO="C"  Length="10"  Style="C"  Example="2123635" >
<Attribute  Name="OrderID"  NO="D"  Length="8"  Style="N"  Rule="YYMMDD99"  Example="02050301" >
<Attribute  Name="Delivery_Date"  NO="E"  Length="8"  Style="D" Rule="YYYYMMDD" Example="20020503" >
<Attribute  Name="ProductID"  NO="F"  Length="8"  Style="C" Rule="999999" Example="000001" >
<Attribute  Name="ProductName"  NO="G" Length="10"  Style="C"  Example="screw" >
<Attribute  Name="Quan"  NO="H"  Length="10"  Style="N" >
<Attribute  Name="Unite"  NO="I"  Length="4  Style="C" >
<Attribute  Name="Price"  NO="J"  Length="10  Style="N" >
<Attribute  Name="Sum"  NO="K"  Length="10"  Style="N" >
<Attribute  Name="Total"  NO="L"  Length="10  Style="N" >
</Vocabulary>
<DataVocabulary>
```

**Figure 5 : using XML describes the Order’s data glossary**

**Step3: using XML to describes the system architecture**

We can use XML-based schema system architecture designed to describe the system architecture diagram as XML. It is shown as follows:



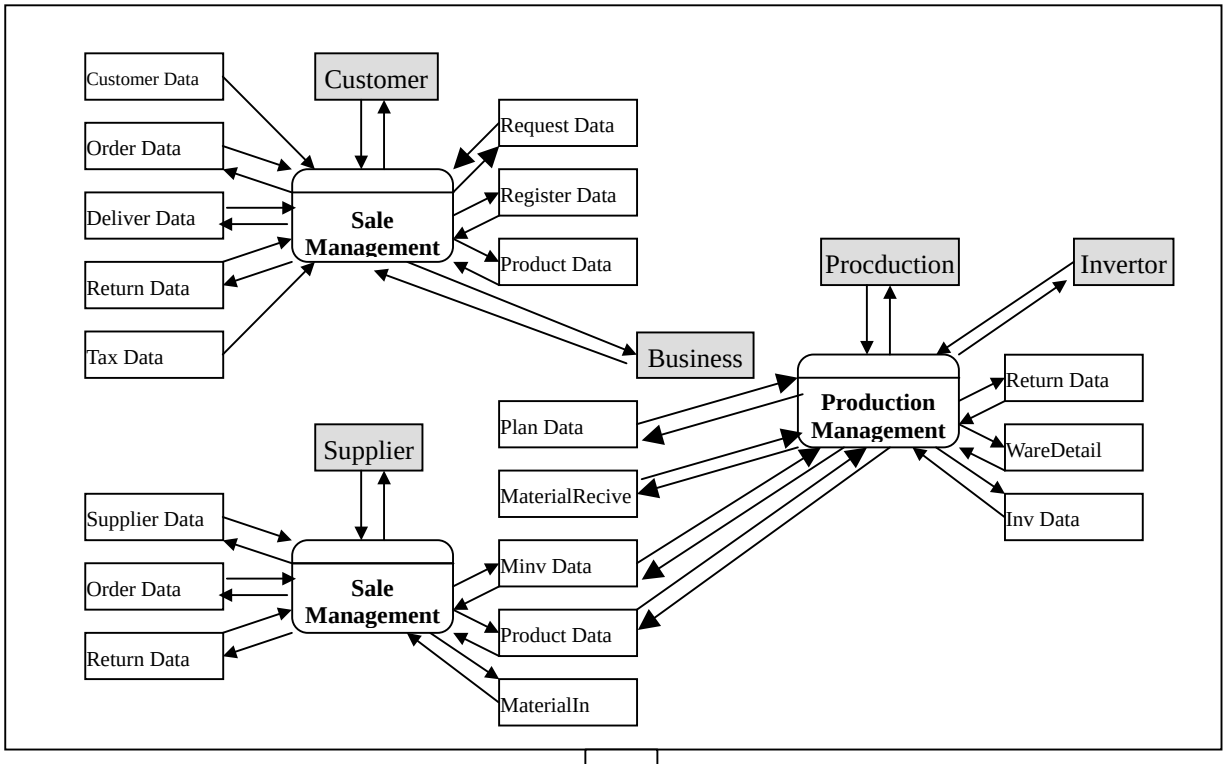

**Described By XML**

```
<?xml version="1.0" encoding="big5"?>
<System_framework  Name="EZIT">
<Model  Name="Sale"  ShowText="SaleMang">
  <Fun  Name ="OrderProcess"  ShowText=" OrderProcess "  OBJName="OrderProcess"></ Fun >
  < Fun  Name =" DeliverProcess "  ShowText =" DeliverProcess "  OBJName ="DeliverProcess"></ Fun >
  < Fun  Name=" ReturnProcess "  ShowText =" ReturnProcess"  OBJName ="ReturnProcess"></ Fun >
</ Model >
< Model  Name =" productionMang"  ShowText=" productionMang ">
  <Fun Name =" MaterialReceive "  ShowText=" MaterialReceive "  OBJName=" MaterialReceive " ></ Fun >
  < Fun Name =" MaterialReturn "  ShowText=" MaterialReturn "  OBJName=" MaterialReturn " ></ Fun >
  < Fun Name =" WareHouseProcess "  ShowText=" WareHouseProcess "  OBJName="WareHouseProcess">< Fun >
    < Fun Name =" CheckProcess "  ShowText=" CheckProcess "  OBJName="CheckProcess" ></ Fun >
</Model>
<Model Name ="SaleMang"  ShowText=" SaleMang ">
    <Fun Name ="Order"  ShowText="Order"  OBJName="Order" ></Fun>
    <Fun Name ="Stock"  ShowText="Stock"  OBJName="Stock" ></Fun>
    <Fun Name ="Back"  ShowText="Back"  OBJName="Back" ></Fun>
</Model>
</ System_framework>
```

**Figure 6 ：using XML describes the system architecture**

**Step 4: using XML to describe each model**

According to the interactive components between XML-based system model and database, the interactive diagram relationship can be shown in XML as follows.




**Described By XML**

```
<Relationship  Name="EZIT"  Level="0">
  <Node  Name="SaleManage"  Type="Modle">
    <DataBase  Name="">
        <Table  Name="Cust_data"  ToSystem="True"></Table>
        <Table  Name="Order_data"  ToSystem="True"  ToTable="True"></Table>
        <Table  Name="Deliver_data"  ToSystem="True"  ToTable="True"></Table>
        <Table  Name="Return_data"  ToSystem="True"  ToTable="True"></Table>
        <Table  Name="Tax_data"  ToSystem="True"></Table>
        <Table  Name="Request_data"  ToSystem="True"  ToTable="True"></Table>
        <Table  Name="Register_data"  ToSystem="True"  ToTable="True"></Table>
        <Table  Name="Product_data"  ToSystem="True"  ToTable="True"></Table>
    </DataBase>
    <Actioner Name="Customer"  ToSystem="True"  ToActioner="True">
    <Actioner Name="Business"  ToSystem ="True"  ToActioner ="True">
  </Node>

  <Node  Name="SaleManage"  Type="Modle">
    <DataBase  Name="">
        <Table  Name="Supplier_data"  ToSystem="True"  ToTable="True"></Table>
        <Table  Name="Order_data"  ToSystem="True"  ToTable="True"></Table>
        <Table  Name="Return_data"  ToSystem="True"  ToTable="True"></Table>
                          : :

  </Node>

                          : :
</Relationship>
```

**Figure7 : using XML describes each model**

**Step 5: using XML to describe the database**

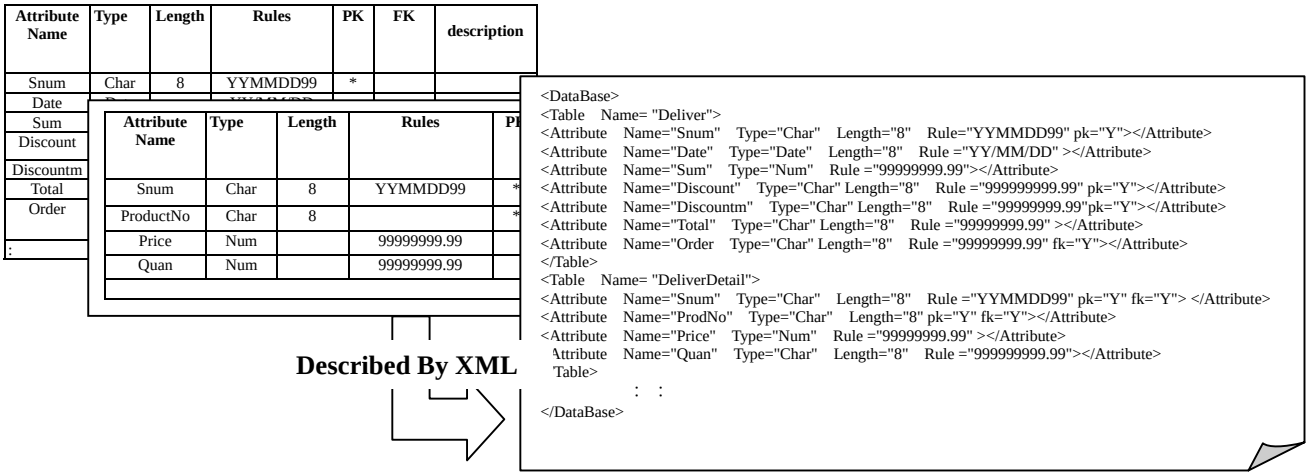



**Figure 8 : using XML describes the database**

**Step 6: using XML to describe each process**

In the model of sales management, sales management includes four subsidiaries: orders, sending returned, asking for outlay and entering in the accounts. According to the components of XML-based system architecture in chapter4, the diagram of the interactive relationship among the subsidiaries can be shown in XML as follows.

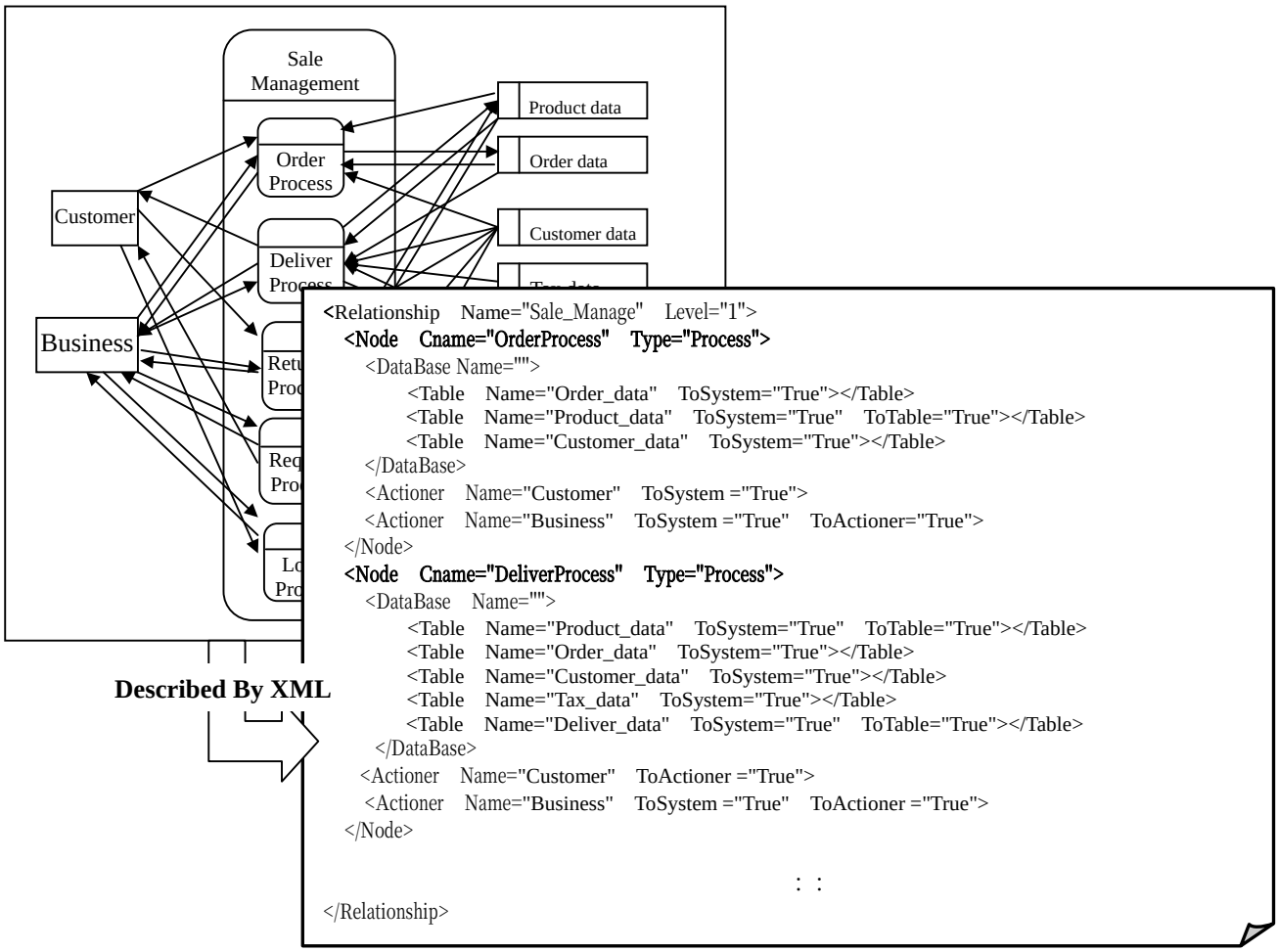



**Figure 9 : using XML describes each process**

**Step 7: using XML to describe each sub-process**

Taking the sending process in sales management model as an example, the sending process can be subdivided into five operations including addition, modification, deletion, inquiry, and print.

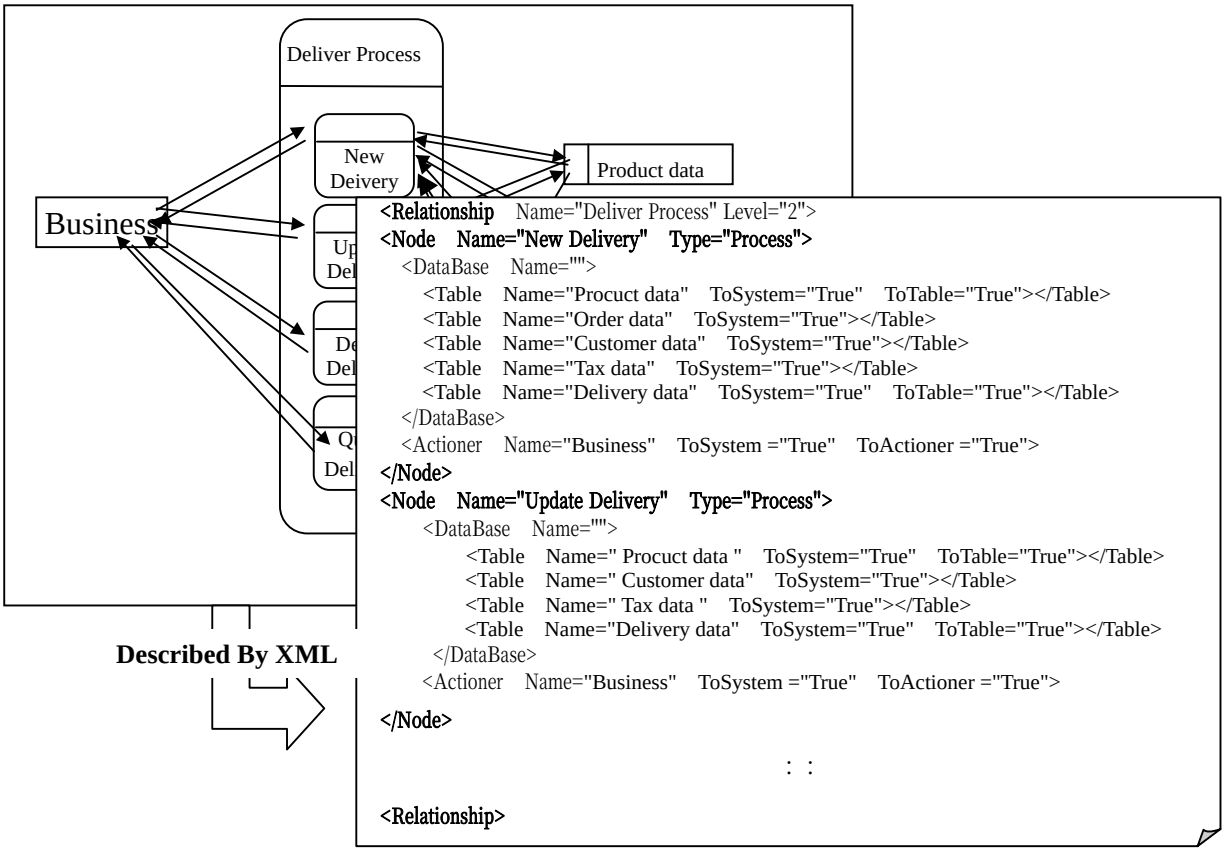



**Figure 10：using XML describes each sub-process**

**Step 8: using XML to describe the procedure of each process**

In “New Delivery”, it can be subdivided into six subsidiary procedures**:** the basic data procedure of the sending bill, the procedure of the tax rate, the procedure of sending amount, the data detective procedure of the sending bill and the data storing procedure of the sending bill. The process steps for each procedure are as follows.

According to “schema of the procedure” in repository, the new delivery procedure can be shown in XML as follows:

| Procedure | Rule |
|---|---|
| Procedure Of Deliver Data | 1. Input OrderNumber<br>2. Automatic make Deliver Number<br>3. Load Customer Data<br>4. Sending bill Process<br>5. Update Deliver data of Database |
| Procedure Of Tax rate | 1. Load Date<br>2. Search Tax from Tax data |
| Procedure Of sending amount | 1. ComputeSummary<br>2. ComputeTotal<br>3. SetTaxMoney<br>4. ComputeMoney |
| Errorr Process | Error process |
| Procedure Of Deliver Save | 1. Call Error Process<br>2. Make Sure Deliever data<br>3. Make Sure Product Inventory data<br>4. Make Sure Material Inventory data |

**Described By XML**

```
<Process name="New Delivery" >
    <Procedure  Cname="Delivery data  process"  Ename="NewDeliver">
        <Rule  Name="Load order Number">
        <Rule  Name=" make Deliver Number ">
        <Rule  Name=" Load Customer Data ">
        <Call_Procedure  Name=" sending bill Process ">
        <Rule  Name=" Update Deliver data of Database ">
    </Procedure>
    <Procedure  Cname=" Tax rate "  Ename="TaxProcess">
        <Rule  Name="Load Date">
        <Rule  Name="Search Tax">
    </Procedure>
    <Procedure  Cname=" Sending amount "  Ename="DeliverMoneyProcess">
        <Rule  Name="ComputeSummary">
        <Rule  Name="ComputeTotal">
        <Rule  Name="SetTaxMoney">
        <Rule  Name="ComputeMoney">
    </Procedure>
    <Procedure  Cname="Error Process"  Ename="ErrorProcess">
        <Rule  Name=" Error Process ">
    </Procedure>
                                    : :
<Process>
```

**Figure 11：using XML describes the procedure of each process**

### Step 9: using XML to describe the rules of each procedure

After subdividing the “procedures of the New Delivery”, we need to describe the rules of the bottom layer. We continue the example-“the procedure of the sending amount” on previous step to describe the process rules as XML.

| Rule Name | Content |
|---|---|
| 1. ComputeSummary | DeliveryDetailSummary=DeliveryDetail ( amount*price) |
| 2. ComputeTotal | Total=( DeliveryDetailSummary *Discount)-Discountvalue |
| 3. SetTaxMoney | Tax= Total *TaxRate |
| 4 ComputeMoney | Money= Total +Tax |

**Described By XML**

```
<BusinessRulesWareHouse>
 <BusinessRule>
      <BusinessRule_header>
            <Rule_Name> ComputeSummary </Rule_Name>
     </BusinessRule_header>
     <BusinessRule_body>
          <Formula>
          DeliveryDetailSummary=DeliveryDetail ( amount*price)
        </ Formula>
     </BusinessRule_body>
  </BusinessRule>
  <BusinessRule>
      <BusinessRule_header>
            <Rule_Name> ComputeTotal </Rule_Name>
     </BusinessRule_header>
     <BusinessRule_body>
          <Formula>
          Total=( DeliveryDetailSummary *Discount)-Discountvalue </ Formula>
     </BusinessRule_body>
  </BusinessRule>
                                    : :
</BusinessRulesWareHouse>
```

**Figure 12 : using XML describes the rules of each procedure**

**Step10:** integrating XML-based software requirement

It will be a simple XML-based software requirement document, if we integrate the above-mentioned XML documents of each step and increase the other relative information.

### 5.2.Reading software requirement documents though XML Parser

The operative principle of reading software requirement documents through XML parser is to turn XML documents into individual programming objects. When the program stores or accesses these objects again, it does the same thing for XML documents such as the system functional structure in requirement documents. After being analyzed by the parser, its branch structure is as follows:

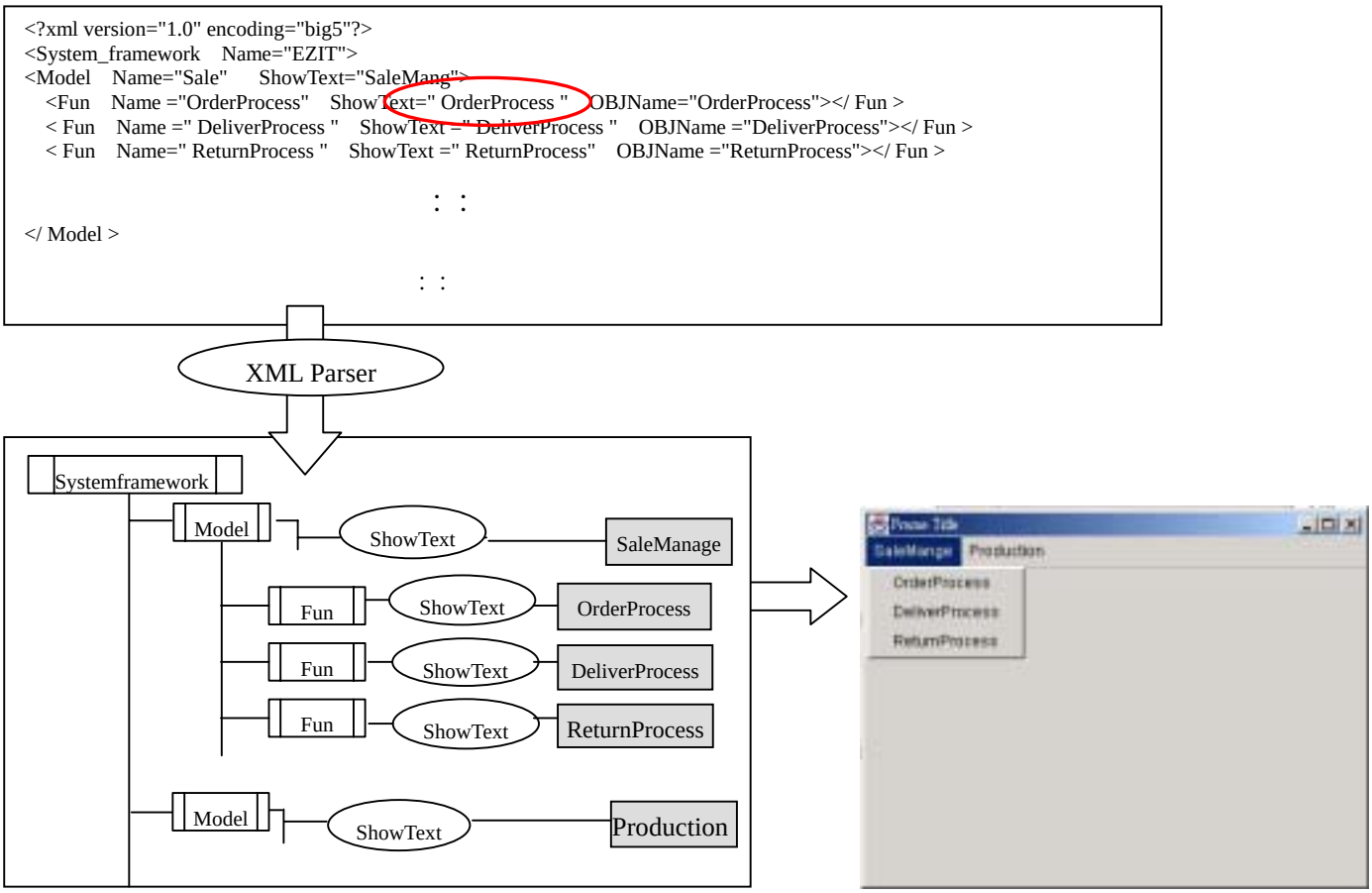



**Figure 13 ：Reading software requirement documents though XML Parser**

Through XML parser, the system functional structure defined by XML software requirement documents will be read by program and the functional table is shown to be one of functional items included in the system.

### 5.3.Implementing the content in XML software requirement documents dynamitic

As above, after the program can read software requirement documents by using XML Parser, the problem now is how to use the data and make the system operate according to them. We continue using the above instance of “the system functional structure” to interpret how to the content defined by XML software requirement documents is implemented dynamically.

The system functional structure defined in XML software requirement documents consist of several models and tags. Each tag includes various function tags. It records the attributes of *Name*, *OBName*, and *ShowText* in the function. *Name* records the Chinese name to differentiate the character of the function, *ShowText* records the language that the

function wants to show and its purpose is to exhibit the frame in the system functional table after the program is read. *ObName* records the functional object's name. When a chain of events is enabled in the function, the object must be implemented. According to the chart below, we can realize that the whole system will operate step by step in terms of the data that it reads and links up the related object.

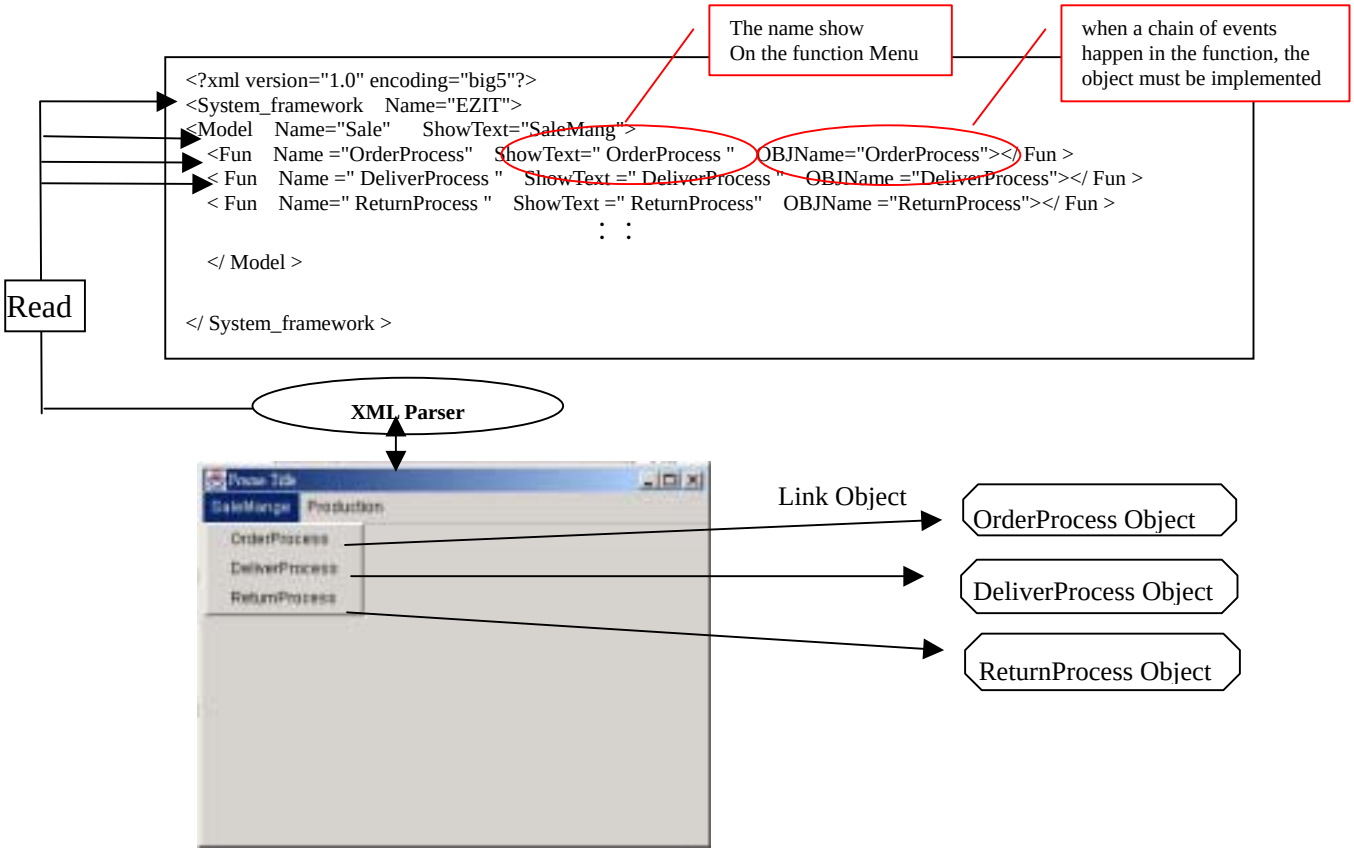



**Figure14 : Implementing the content in XML software requirement documents**

The above chart shows that the system operates in accordance with the content defined in XML software requirement documents so that if the enterprises need to change its functional structures because of the external environments, they just need to alter software requirement documents. We will introduce two changeable situations for readers.

**(1)Add/delete some function**

If the enterprises want to add a function of entering in accounts in the above system, they need to add a description in software requirement documents only. It is:

**<function Name =”AccountProcess” OBName =”Account Process” ShowText =” AccountProcess”></function>.**

And then the system will link up the object process named as “account process” to add a new system function according to the description. On the contrary, if the enterprises want to delete some function, they just need to delete the description of the function.

```
<?xml version="1.0" encoding="big5"?>
<System_framework Name="EZIT">
<Model Name="Sale" ShowText="SaleMang">
  <Fun Name ="OrderProcess" ShowText=" OrderProcess " OBJName="OrderProcess"></ Fun >
  < Fun Name =" DeliverProcess " ShowText =" DeliverProcess " OBJName ="DeliverProcess"></ Fun >
  < Fun Name=" ReturnProcess " ShowText =" ReturnProcess" OBJName ="ReturnProcess"></ Fun >
  < Fun Name =" LoginProcess " ShowText =" LoginProcess " OBJName="LoginProcess" ></ Fun >
                                    : :
  </ Model >
```

**(2)Update the system function**

If the enterprises want to modify the order process that they deal with, they just need to amend the description of *OBJ_Name* turning into one new *object-Order Process2*. When we restart the system, it will link up the new order object.

## 6.Conclusion

This research brings up system development methodology based on XML software requirement documents to achieve the purpose of making the business logic and the system logic divided. Because of this purpose, we probe for writing way of traditional software requirement documents first in order to understand the meaning and inadequacy for the whole system developmental process. Furthermore, we research in the relative XML techniques from W3C deeply. In the research process we discover the advantages and feasibility using XML as the descriptive language of software requirement documents, and bring up the system developmental way of software requirement documents based on XML to accomplish the research purpose.

The development methodology is divided into four process phases: the requirement analysis, the requirement documents design, programming design, and system maintenance.

What to do in each phase and the difference from the traditional developmental methodology is discussed as well. In order to prove its feasibility, we also use an example of a business process system to illustrate what should be implemented in each phase.

## Reference


[1] Hsien-kai.Chiu, "Applying XML to Document Business Logic through TBCG", Master's paper, National Central University, 2000, P5。

[2] Chang-kuo.Huang "A UML and Business Rule-based Business-tier Software Component Development Environment", Master's paper , National Taiwan University of Science and Technology, 2000, P2。

[3] Hung-chih.Kuo、Yen-ping.Chi, System analysis and Design, Hua-tsai, 1995, P107

[4] Sommerville Ian , Software Engineering 5thed , Wokingham, England ; Reading, Mass. : Addison-Wesley Pub. Co., c1996

[5] Davis,A.M.,Software Requirements:Analysis and Specification , Englewood Cliffs NJ:Prentice-Hall [82,123],1990.

[6]Bray,Tim,et. al., " Extensible Markup Languag 1.0 " ,W3C Recommendation, 1998.

[7] Avraham Leff, James T Rayfield , "Web-Application Development Using the Model/View/Controller Design Pattern" , Enterprise Distributed Object Computing Conference, 2001. Proceedings. Fifth IEEE International , 2001.

[8]Carla Sadtler et al., "Patterns for e-business: User-to-Business Patterns for Topology 1 and 2 using WebSphere Advanced Edition," IBM Redbooks publication, 2000.

[9]M.J Mahemoff, L.J Johnston,Handling multiple domain objects with Model-View-Controller," Technology of Object-Oriented Languages and Systems, 1999. TOOLS 32. Proceedings , 1999 ,Page(s): 28 –39

[10] Roger S. Pressman,"Software Engineering: A Practitioner's Approach",2001

[11]E. Guerrieri., "Software Document Reuse with XML," In P. Devandu and J. Poulin, editors, Pro. 5 th Int. Conf. on Software Reuse, pp:246-- 254, Victoria, Canada, June 1998.