

2012

# Ofbiz Implementation of an ERP

Apache Open For Business



## Acknowledgement

First and foremost, we want to thank VENUS DATA PRODUCTS, Surat for giving us the source of power, knowledge and strength to furnish the preliminary project work. We would like to express our gratitude to our guide Prof. Ami Choksi for her wisdom, endurance and encouragement. She motivates us about the importance of constant and gives us her outstanding professional conduct. We also sincerely thanks for the time spent proof reading and correcting our many mistakes. We acknowledge our sincere in datedness and gratitude to our parents for their love, dream and, sacrifice throughout our life. Our sincere thanks go to all our classmates and members of the staff of the Computer Engineering Department, CKPCET, who helped us in many ways and made our stay at CKPCET pleasant and unforgettable. And also thanks to Head of Computer Engineering, Prof. Ami Choksi for her co-operation.

Vivek Champaneria

## ***Abstract***

*For achieving the better solution for SME is found in the offering in the offering by apache business project OfBiz. It is Apache's major project and it gives similar functionality just as the SAP which is for large enterprise company. Here, we will do the implementation of the OfBiz at VENUS DATA PRODUCTS, they uses of the sales accounting system and manage its service calls manually and on a spread sheet. The implementation of the SAP system here is not feasible because of its cost. The need of their business will be fulfilled by the development of the OfBiz products. It will help to use the available resources more efficiently and provide the good customer support. It will help to improve its current market share in the business by helping the employ to work in efficient way .It will save the money that is wasted in internal process because of its integrated and collaborative approach.*

*Brief Introduction of the project and OfBiz and the obvious and optimal solutions are described in detail in **Chapter 1**.*

*Whole development of our project along with OfBiz Component, architecture and Features is discussed in detail in **Chapter 2**.*

*All OfBiz modules like Catalog, HR, Order and CRM is in **Chapter 3**.*

***Chapter 4** gives the us the knowledge about usage of the system practically.*

***Chapter 5** contains Impacts Of OfBiz.*

***Chapter 6** Contains the project summery.*

**OFBIZ IMPLEMENTATION OF AN ERP**  
**A PROJECT REPORT**

Submitted by

**VIVEK CHAMPANERIA**  
(Enrollment No – 080090107003)

In fulfillment for the award of the degree of  
**BACHELOR OF ENGINEERING**

In

Department of Computer Engineering



**C. K. Pithawalla College of Engineering and Technology**

**Surat**

**Gujarat Technological University, Ahmadabad**

**May, 2012**



C. K. Pithawalla College of Engineering and Technology

Surat

Department of Computer Engineering

Year – 2012

CERTIFICATE

Date: 25/05/2012

*This is to certify that the dissertation entitled “OFBIZ- IMPLEMENTATION OF AN ERP” has been carried out by VIVEK CHAMPANERIA under my guidance in fulfillment of the degree of Bachelor of Engineering in Department of Computer Engineering (8<sup>th</sup> Semester) of Gujarat Technological University, Ahmadabad during the academic year 2011-12. The work done by them is found satisfactory.*

Guide  
(Prof. Ami T. Choksi)

Jury

Head of the Department  
(Prof. Ami T. Choksi)

# TABLE OF CONTENTS

Chapter No	Title	Page No
<b>Chapter 1</b>	<b>Introduction to Project</b>	<b>1</b>
1.1	Problem Summery	1
1.2	Obvious Solution	2
1.3	Optimal Solution	2
<b>Chapter 2</b>	<b>Introduction to Apache Open for Business</b>	<b>3</b>
2.1	OFBiz: Component-Oriented Architecture	3
2.2	OFBiz: MVC Architecture	4
2.3	OFBiz: Service-Oriented Architecture (SOA)	5
2.4	OfBiz Features	6
2.5	Ofbiz Folder Structure	7
	2.5.1 Applications Folder	7
	2.5.2 Framework Folder	7
	2.5.3 Hot-deploy Folder	7
	2.5.4 Special purpose Folder	8
	2.5.5 Lib Folder	8
	2.5.6 Runtime Folder	8
	2.5.7 Theme Folder	8
2.6	Ofbiz Application Structure	8
	2.6.1 Config Directory	9
	2.6.2 Data Directory	9
	2.6.3 Service Directory	9
	2.6.4 Web App Directory	9
	2.6.5 Widget Directory	9
2.7	Ofbiz Descriptors	10
	2.7.1 Component Load.xml	10
	2.7.2 Controller Descriptor	11
	2.7.3 Entity Model Descriptor	12

	2.7.4	Ofbiz-component Descriptor	12
	2.7.5	Service Model Descriptor	14
	2.7.6	Web.xml Descriptor	14
2.8		Ofbiz Engine	16
	2.8.1	Java Services	16
	2.8.2	OfBiz Services	17
	2.8.3	User Interface	19
	2.8.4	OfBiz Database Entity Engine	22
<b>Chapter 3</b>		<b>Ofbiz Modules</b>	<b>24</b>
3.1		Catalog Manger	24
	3.1.1	Introduction	24
	3.1.2	Database	25
3.2		HR manager	31
	3.2.1	Introduction	31
	3.2.2	Database Schema Human Resource	31
3.3		Order Manager	38
	3.3.1	Introduction	38
	3.3.2	Database Schema	38
3.4		CRM(Trouble Ticketing) Manager	42
	3.4.1	Introduction	42
	3.4.2	Database	42
	3.4.3	Functionalities	43
<b>Chapter 4</b>		<b>Using Ofbiz</b>	<b>44</b>
4.1		Startup Ofbiz	44
4.2		Create Product	44
4.3		Create Employee	47
4.4		Create Request complaints	48
<b>Chapter 5</b>		<b>Impacts of Ofbiz</b>	<b>49</b>
<b>Chapter 6</b>		<b>Conclusion</b>	<b>50</b>

# LIST OF FIGURES

<b>Figure No.</b>	<b>Figure Description</b>	<b>Page No.</b>
2(a)	OfBiz Component Oriented Architecture	4
2(b)	OfBiz MVC Architecture	5
2(c)	OfBiz Service Oriented Architecture	6
2(d)	Ofbiz package folders	8
2(e)	web application Directory	10
2(f)	Component-Load.xml	10
2(g)	Controller Descriptor	11
2(h)	Entity Model.xml	12
2(i)	Ofbiz-component.xml	13
2(j)	Service Model Descriptor	14
2(k)	web.xml Descriptor	15
2(l)	Java Services in OfBiz	16
2(m)	OfBiz Service Engine	19
2(n)	User Interface Designing in Ofbiz	20
2(o)	OfBiz Screen Widget Architecture	21
2(p)	OfBiz Entity Engine	23
3(a)	Product Definition	25
3(b)	Product Category	27
3(c)	Product Pricing	28
3(d)	Product Features	29
3(e)	Overall Product model	30
3(f)	Employment	32
3(g)	Position Definition	33
3(h)	Position Fulfillment	34
3(i)	Position Reporting Structure Schema	35
3(j)	Salary Determination	36
3(k)	Human Resource Schema	37
3(l)	Standard order model.	40
3(m)	Overall order model.	41
3(n)	CRM Schema	42
4(a)	Create Catalog	44
4(b)	Edit Product Catalog	44
4(c)	New Category	45
4(d)	Update Category	45
4(e)	Create Product	46
4(f)	Update Product	46
4(g)	Create New Employee	47
4(h)	Add new Employee	47
4(i)	Customer Relation main page	48
4(j)	Create Request page	48



# LIST OF TABLES

Table No	Table Name	Page No
1	Markup Language	20

## **1.1 Problem Summery**

The SME Companies are generally making the DTP Software. The SME has limited number of customers as the company is developing. As the growth of business increase the number of customer increase and the requirement of increase in software quality and services and standards of the company. To maintain the service standards and provides the tools that are required by the customer is much more difficult for the SME.

The SME needs the ERP System which includes the FA (Finance and Accounting), Inventory Management and Manufacturing Management. This will give the better understanding of the activities of the various departments in the company and manage the dynamic nature of business, giving accurate and timely Reports to the key stakeholders.

To solve this problem the implementation of an ERP System is required. The ERP solution with the modules of Financial Accounting, Inventory Management, Sales Force Management, Customer Relationship Management and Human Resource with various POS (Point of Sale). This is solved by the German company SAP, Inc. The program developed by the SAP is much effective and gives the desired results in the Large Enterprise Companies. The implementation of SAP in SME is unaffordable due to its high cost and the high probability of failure due to changing requirements of business. Hence the Creation of ERP System with OfBiz tool is much affordable and also it gives promising results with low cost. Hence it is best suitable by the SME. The implementation of OfBiz against the SAP requires these modules at the VENUS DATA PRODUCTS.

- Finance and Accounting
- Inventory Management
- Manufacturing Management
- Customer Relation Management

- Human Resource Management
- Sales Force Automation
- Flexible Reporting

The OfBiz promises the results desired by the VENUS DATA PRODUCTS. The Company used to handle the customer relation center using the manual Phone Calls which is very costly because it will take the 1-2 employee of the only one operation. The Accounting Section also handles manually so the chances of ambiguity is more and there is no central control all over the business so the company doesn't able to use its resources as efficiently as it can. The OfBiz will give the ERP system which will overcome this inability.

### **1.2 Obvious Solution**

The solution to this problem is solved in the Large Enterprise Company is by deploying the ERP (Enterprise Resource Planning System). It helps to understand the company market needs and also become a bridge between the various departments of the enterprise company. IBM, DELL and ESSAR companies have deployed the SAP (System, Application and Products) ERP System to maintain the business and their business standards. Though the deployments of SAP system in the SME (Small and Medium Enterprise) Companies are not affordable because of its high cost and its deployment will need much time. It will give the central control over the business and also the gives the flexible end user interface. The SME Companies doesn't afford the solution given by the SAP due to its high cost and high probability of failures.

### **1.3 Optimal Solution**

The solution to this lies in the Apache the business Project OfBiz (Open for Business).It is the Apache's major project. It is giving the same functionality as the SAP will introduce in the Large Enterprise Company. As apache is a non-benefit foundation the OfBiz is an Open Source License. It will come with very handy features.

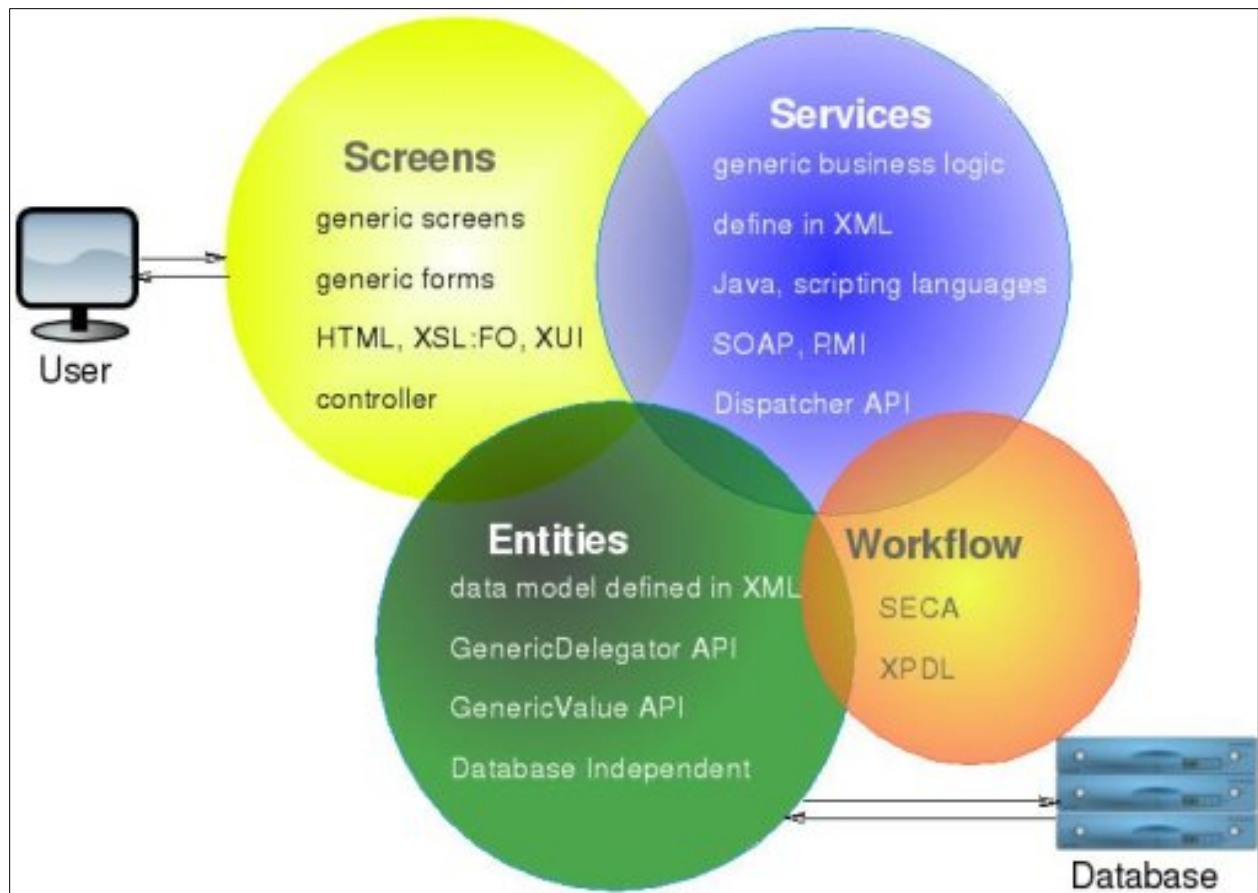
Apache Open For Business (Apache OFBiz) is an open source enterprise resource planning (ERP) system. It provides a suite of enterprise applications that integrate and automate many of the business processes of an enterprise

OFBiz is one such noteworthy free open source framework that is based on Sun JAVA, J2EE, W3C XML, HTML, and SOA. It has a loosely coupled multilayered architecture that implements design patterns such as Model-View-Controller (MVC), Service Oriented Architecture (SOA), and database-independent persistence layer that allows existing or custom-developed applications to be added by dropping into a components-based architecture.

1. OfBiz Component Based Architecture
2. OfBiz MVC Architecture
3. OfBiz Service Oriented Architecture

### **2.1 OFBiz: Component-Oriented Architecture**

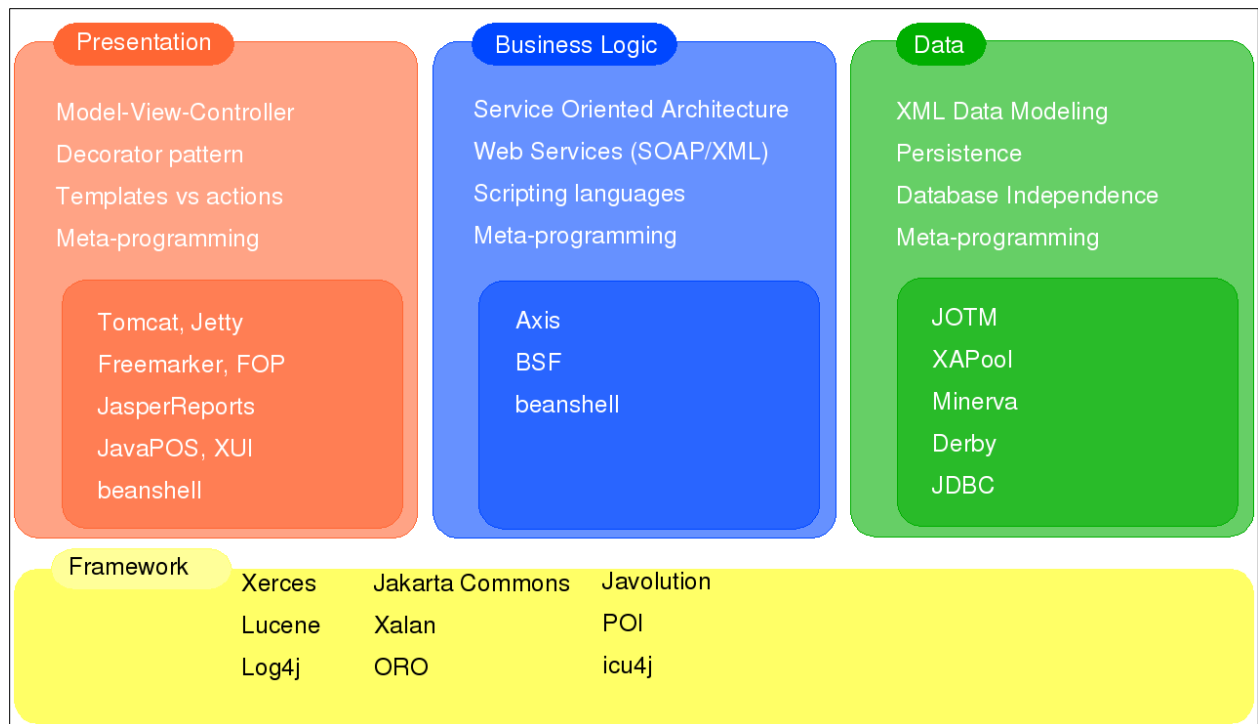
OFBiz is totally a component-oriented architecture in which each module or functionality can be defined as a separate component and can be integrated into the same application or reused across many applications deployed in the same server. For this, OFBiz comes up with its own in-built server OFBiz that in itself contains Tomcat 6.0 and Tomcat 5.5, out of which we can choose one.



**Fig. 2(a) OfBiz Component Oriented Architecture**

## **2.2 OFBiz: MVC Architecture**

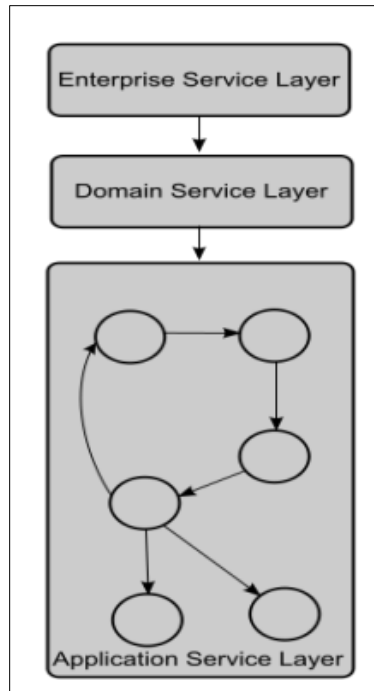
The MVC mainly consists of reusable decorator patterns in the form of special purpose template language called FTL (FreeMarker Template Language) instead of the traditional JSP pages. The templates together work with actions in the form XML meta-programming that are coded as Screen Widgets. The widgets and FTL together become reusable across many different applications, for example, the same Login Widget can be used in both Shopping as well as Accounting applications. Thus, MVC achieves the true means of reusability in the form FTL templates and Screen Widgets. The actions can be invoked using special scripting language called BeanShell Script or Groovy. To add look and feel for the application, OFBiz greatly supports integrating predefined visual themes or customizing the existing themes or a newly created theme.



**Fig. 2(b) OfBiz MVC Architecture**

### **2.3 OFBiz: Service-Oriented Architecture (SOA)**

Next comes Service-Oriented Architecture (SOA) which sits in the middle part of the tiered framework and contains the core business logic to be processed. Services in OFBiz operate in a Service-Oriented Architecture (SOA). These services not only have the ability to invoke other services internally, but can also be ‘opened up’ and invoked by remote applications using, amongst other methods, the widely adopted messaging protocol SOAP.



**Fig. 2(c) OfBiz Service Oriented Architecture**

## **2.4 OfBiz Features**

1. Flexible Open Source License
2. Large Community Support
3. Modules of Financial & Accounting, Customer Relationship Management (CRM), Human Resource Planning and Manufacturing Resource Planning
4. Thousands of downloads and many implementation, even on large scale
5. Very Flexible Framework designed using latest technologies in mind where customization and enhancing its features can be achieved without complexities, following standardized best practices
6. Complements and Integrates various other Open Source Solutions
7. Written in JAVA it is OS independent web based Solution

8. Provides the functionality – SOA (Service Oriented Architecture), SOAP (Simple Object Access Protocol), OAGIS (Open Applications Group Integration Specification) and REST (Representational State Transfer).
9. Database Supported – MySQL, Oracle, PostgreSQL
10. Integrates with famous Open Source projects for e-commerce and Portal like Magento, & LifeRay
11. Easy to add plugins due to SOA
12. Integrates with BI Platforms like JasperSoft and Pentaho
13. BPM (Business process management) has been implemented by many ISV (Independent software vendor).

## **2.5 Ofbiz Folder Structure**

Ofbiz package contains mainly four folders namely Application, Framework, Hot-Deploy, Special purpose.

### **2.5.1 Applications Folder**

This folder contains the applications that are necessary for an enterprise application such as human resource management, Product management, etc.

### **2.5.2 Framework Folder**

This folder contains the necessary files and classes that require the Ofbiz to run. This is the heart of an Ofbiz package. This folder also contains the mandatory database files that are require to make organizational application run.

### **2.5.3 Hot-deploy Folder**

This folder contains the user-defined custom applications. This folder serves the purpose of extending the existing application and adds new features to the application without doing any changes in the currently existing and working applications.



#### 2.5.4 Special purpose Folder

This folder contains the application that every organization may require or may not require. For special purpose the organization require some specific application such as ecommerce, online payment gateways, portal, etc.

#### 2.5.5 Lib Folder

This folder contains the third party library files and Drivers that require during the running the application.

#### 2.5.6 Runtime Folder

This folder contains the temporary files that are created during the runtime of server. It also contains the history and log and backup files of server.

#### 2.5.7 Theme Folder

This folder contains the image files and css files and webpages that gives the creative look and feel to the system.

Name	Date modified	Type
.svn	4/4/2012 9:08 AM	File folder
applications	2/23/2012 3:24 AM	File folder
bin	4/5/2012 2:07 PM	File folder
debian	2/23/2012 3:24 AM	File folder
framework	2/23/2012 3:25 AM	File folder
hot-deploy	4/4/2012 9:09 AM	File folder
lib	2/23/2012 3:25 AM	File folder
runtime	4/4/2012 10:04 AM	File folder
specialpurpose	2/23/2012 3:25 AM	File folder
themes	2/23/2012 3:24 AM	File folder
tools	2/23/2012 3:25 AM	File folder

**Fig.2(d) Ofbiz package folders**

### **2.6 Ofbiz Application Structure**

Ofbiz application serves the purpose of organization by implementing the business logic into the applications. Ofbiz application contains standard java application directories and some

special directories. Each Ofbiz application contains build, config, data, document, entitydef, script, servicedef, src, webapp, widget directories along with Ofbiz-component.xml file.

#### **2.6.1 Config Directory**

This directory contains the configuration files such as java properties files and xml files. This directory is dedicated to configuration files only each web application configuration is found in config directory

#### **2.6.2 Data Directory**

This directory contains the database schema files. This folder contains the database tables that being created or used by application .This folder serves the purpose of data layer in MVC architecture.

#### **2.6.3 Service Directory**

This directory contains services that are called by the web application and created by the application. This folder has all the required information about the services that are used in the application. This folder serves the purpose of control layer in MVC architecture.

#### **2.6.4 Web App Directory**

This directory has the actual web application contains and the programing logic that is used in the application. This is the standard application directory for any java web application.

#### **2.6.5 Widget Directory**

This directory contains the widgets that are the files of widget engine for decorating the web application. The files in this folders act as master page in web application. This folder contains the screen definition files, forms definition file, Menu Files and Tree Files.

Name	Date modified	Type
.svn	4/4/2012 9:08 AM	File folder
build	2/23/2012 3:25 AM	File folder
config	2/23/2012 3:24 AM	File folder
data	2/23/2012 3:24 AM	File folder
documents	2/23/2012 3:24 AM	File folder
entitydef	2/23/2012 3:24 AM	File folder
script	2/23/2012 3:24 AM	File folder
servicedef	2/23/2012 3:24 AM	File folder
src	2/23/2012 3:24 AM	File folder
webapp	2/23/2012 3:24 AM	File folder
widget	2/23/2012 3:24 AM	File folder
build	2/23/2012 3:24 AM	Safari Document
ofbiz-component	2/23/2012 3:24 AM	Safari Document

**Fig. 2(e) web application Direcatory**

## **2.7 Ofbiz Descriptors**

### **2.7.1 Component Load.xml**

This is an important descriptor of an Ofbiz package. this folder contains the information about the application componanat that are loaded in the system. As shown in figure each load componenet element cotaints the name and the location of the compnanat files.

```
<?xml version="1.0" encoding="UTF-8"?>
<component-loader xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/component-loader.xsd">
  <load-component component-location="party"/>
  <load-component component-location="workeffort"/>
  <load-component component-location="product"/>
  <load-component component-location="humanres"/>
  <load-component component-location="order"/>
  <load-component component-location="marketing"/>
</component-loader>
```

**Fig. 2(f) Component-Load.xml**

### 2.7.2 Controller Descriptor

This Descriptor is the main key descriptor in Ofbiz application. It transfer the request from the server to specific location and according to events. The order of tags in controller descriptor is as follows.

1. Site-conf
2. Include
3. Pre-processor
4. Post-processor
5. Request-map
6. Response-map
7. View-map

```
<?xml version="1.0" encoding="UTF-8"?>
<site-conf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/site-conf.xsd">

  <include location="component://common/webcommon/WEB-INF/common-controller.xml"/>
  <include location="component://party/webapp/partymgr/WEB-INF/controller.xml"/>

  <description>    </description>

  <!-- view handlers -->
  <handler name="service-multi" type="request" class="org.ofbiz.webapp.event.ServiceMultiEventHandler"/>

  <!-- Events to run on every request before security (chains exempt) -->
  <preprocessor>
  </preprocessor>

  <!-- Events to run on every request after all other processing (chains exempt) -->
  <postprocessor>
    <event name="test" type="java" path="org.ofbiz.webapp.event.TestEvent" invoke="test"/>
  </postprocessor>

  <!-- Request Mappings -->
  <request-map uri="view">
    <security https="true" auth="false"/>
    <response name="success" type="request" value="main"/>
  </request-map>

  <!-- View Mappings -->
  <view-map name="LookupPartyName" type="screen" page="component://party/widget/partymgr/LookupScreens.xml#1

</site-conf>
```

**Fig. 2(g) Controller Descriptor**

### 2.7.3 Entity Model Descriptor

This Descriptor file is the database blue print file. Tables that are being accessed are mentioned in this descriptor file. Application can only have access to schemas that are mention in the entity model descriptor file.

This file contains entity model as a root tag and entity tag in which attribute name as table name is written. The child tag is name as field tag which contains the information about the column of the table and its identity as primary key as an attribute of field tag.

The relation with other table such as primary key foreign key relation are described in the relation field as shown in figure.

```
<?xml version="1.0" encoding="UTF-8"?>
<entitymodel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/entitymodel.xsd">
  <entity entity-name="PartyQual"
    package-name="org.ofbiz.humanres.ability"
    title="Party Qualification Entity">
    <field name="partyId" type="id-ne"></field>
    <field name="partyQualTypeId" type="id-ne"></field>
    <field name="qualificationDesc" type="id-long"></field>
    <field name="title" type="id-long"><description>Title of degree or job</description>
    <field name="statusId" type="id"><description>Status e.g. completed, part-time etc.<
    <field name="verifStatusId" type="id"><description>Verification done for this entry
    <field name="fromDate" type="date-time"></field>
    <field name="thruDate" type="date-time"></field>

    <prim-key field="partyId"/>
    <prim-key field="partyQualTypeId"/>
    <prim-key field="fromDate"/>

    <relation type="one" fk-name="PARTY_QUAL_PRTY" rel-entity-name="Party">
      <key-map field-name="partyId"/>
    </relation>
  </entity>
</entitymodel>
```

**Fig. 2(h) Entity Model.xml**

### 2.7.4 Ofbiz-component Descriptor

This Descriptor contains the security information about the web application. Each Ofbiz web application contains this file in the application folder. Whenever the Ofbiz load the component from the component load descriptor it will search for this file in application

folder. This file gives the information about the location of application in the relative path and owner of the application and permission of application.

Each Ofbiz component Descriptor has root tag as Ofbiz-component tag and its successor as web-app tag. This is portion is highlighted in the figure.

Entity resource and resource loader are the optional. They give detailed information about the application.

```
<?xml version="1.0" encoding="UTF-8"?>
<ofbiz-component name="humanres"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://ofbiz.apache.org/dtds/ofbiz-component.x

  <resource-loader name="main" type="component"/>

  <classpath type="jar" location="build/lib/*"/>
  <classpath type="dir" location="config"/>

  <entity-resource type="model" reader-name="main" loader="main"
location="entitydef/entitymodel.xml"/>

  <entity-resource type="data" reader-name="seed" loader="main"
location="data/HumanResTypeData.xml"/>
  <entity-resource type="data" reader-name="demo" loader="main"
location="data/HumanResDemoData.xml"/>

  <service-resource type="model" loader="main" location="servicedef/services.xml"/>

  <webapp name="humanres"
    title="HR"
    description="HumanResourceApplication"
    server="default-server"
    location="webapp/humanres"
    base-permission="OFBTOOLS,HUMANRES"
    mount-point="/humanres"
    app-bar-display="true"/>
</ofbiz-component>
```

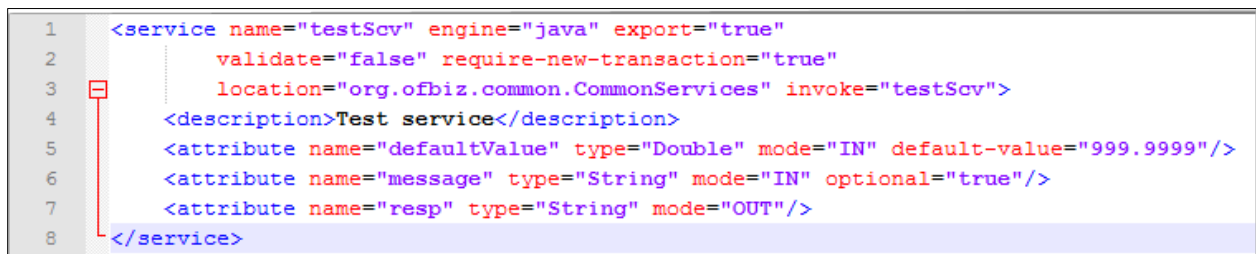
**Fig. 2(i) Ofbiz-component.xml**

### 2.7.5 Service Model Descriptor

This Descriptor contains the information about the services that are used by the application. Root tag of service model descriptor is Services and its successor as service.

Service tag have attributes as Name, engine, location, invoke. Name attributes gives the logical name to the service while engine gives the type of service such it may be java service that is called by the application. Location attribute gives the information to the Ofbiz where the service is located and how to invoke it with help of invoke attribute.

Here location parameter will give the location of java class in the Ofbiz package and invoke parameter will give the information to which method it invoke.

The image shows a screenshot of an XML code editor with a light blue background. On the left side, there is a vertical line of numbers from 1 to 8. A red bracket on the left side of the code spans from line 3 to line 7. The XML code is as follows:

```
1 <service name="testScv" engine="java" export="true"
2     validate="false" require-new-transaction="true"
3     location="org.ofbiz.common.CommonServices" invoke="testScv">
4     <description>Test service</description>
5     <attribute name="defaultValue" type="Double" mode="IN" default-value="999.9999"/>
6     <attribute name="message" type="String" mode="IN" optional="true"/>
7     <attribute name="resp" type="String" mode="OUT"/>
8 </service>
```

**Fig. 2(j) Service Model Descriptor**

### 2.7.6 Web.xml Descriptor

This Descriptor is the standard java web application descriptor. This files contains the name of the application. The order of tag in descriptor is as follows.

1. Web app
2. contex-pram
3. filter
4. filter-mapping
5. listener
6. servlet
7. servlet-mapping
8. session-config
9. welcome-list

```

<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <display-name>Open For Business - HumanRes Manager</display-name>
  <description>HumanRes Manager Module of the Open For Business Project</description>
  <context-param>
    <param-name>entityDelegatorName</param-name>
    <param-value>default</param-value>
    <description>The Name of the Entity Delegator to use, defined in entityengine.xml</description>
  </context-param>
  <filter>
    <filter-name>ContextFilter</filter-name>
    <display-name>ContextFilter</display-name>
    <filter-class>org.ofbiz.webapp.control.ContextFilter</filter-class>
    <init-param>
      <param-name>disableContextSecurity</param-name>
      <param-value>N</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>ContextFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <listener><listener-class>org.ofbiz.webapp.control.ControlEventListener</listener-class></listener>
  <servlet>
    <servlet-name>ControlServlet</servlet-name>
    <display-name>ControlServlet</display-name>
    <description>Main Control Servlet</description>
    <servlet-class>org.ofbiz.webapp.control.ControlServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>ControlServlet</servlet-name>
    <url-pattern>/control/*</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>60</session-timeout> <!-- in minutes -->
  </session-config>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>

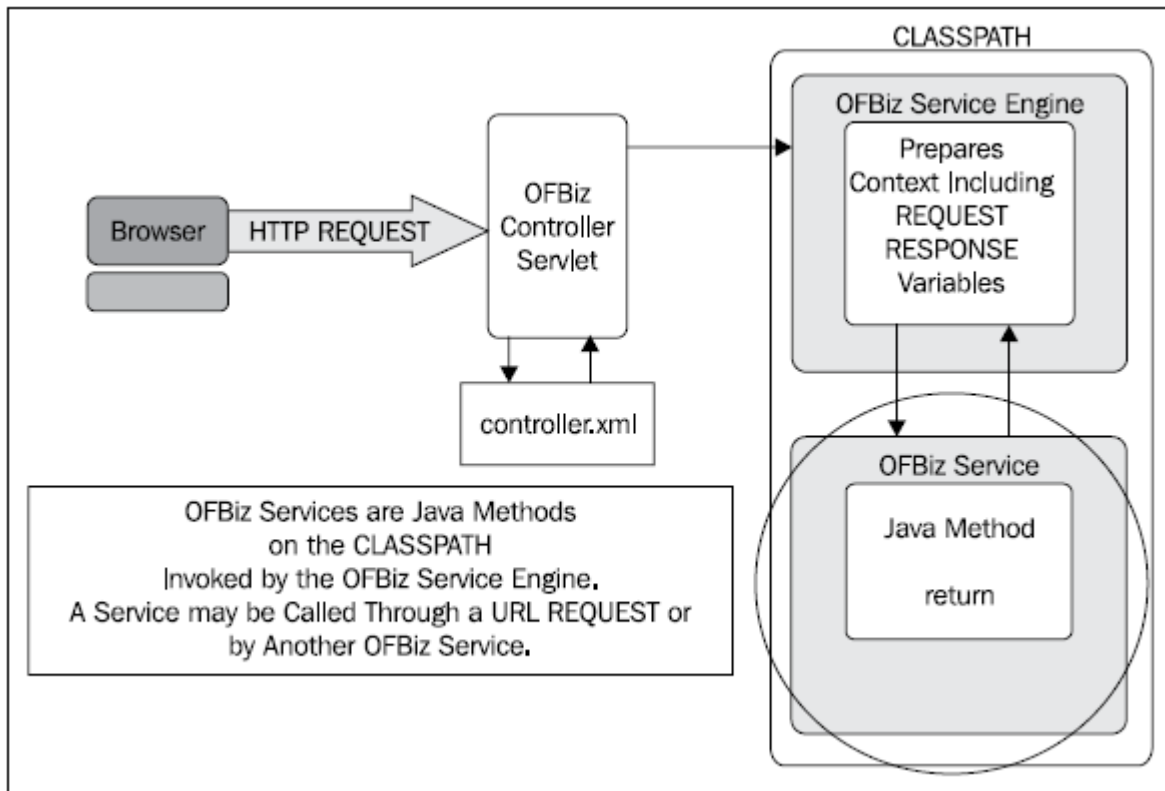
```

**Fig. 2(k) web.xml Descriptor**



## 2.8 Ofbiz Engine

### 2.8.1 Java Services



**Fig. 2(l) Java Services in OfBiz**

Services are Java methods that are invoked by the OFBiz Service Engine as shown. Consumers of OFBiz Services may be web browsers, other OFBiz Events or Services, and even remote systems (for example SOAP-based messaging requests).

For a quick test of an OFBiz Service, call the Service directly from a web page. To make a test invocation of an OFBiz Service from a web page:

1. Create a controller.xml request-map entry for the Service. For example, to call a Service called myService, the controller.xml file may have an entry such as:

```
<request-map uri="myService">
  <security https="false" auth="false"/>
  <!-- invoke points to the Service name as defined in the Service Definition -->
```

```
<event type="service" invoke="myService"/>
<response name="success" type="view" value="main"/>
<response name="error" type="view" value="error"/>
</request-map>
```

2. Create an HTML form or href request for the Service's URI as configured in the controller.xml, as follows:

```
<form action ="<@ofbizUrl>myService</@ofbizUrl>" method="post">
<input type="hidden" name="param1" value="I_am_hidden">
</form>
<a href="<@ofbizUrl>myService&param1=I_am_hidden</@ofbizUrl>
```

3. Invoke the HTML form or href link from the web page and observe the results. By putting debug statements in your Service, you should see them log to the console window if your Service logic is being executed.

### 2.8.2 OfBiz Services

OfBiz has been characterized as having an "event-driven, service-oriented architecture". OfBiz implemented a number of architectural features enabling a service-oriented design.

These features include:

A context-aware Service Engine available for use across the entire framework or, if configured, externally through supported network interfaces.

OFBiz Service consumers need not concern themselves with the location of the called Service or with a Service's implementation details. OFBiz handles all that transparently to both the service provider and consumer.

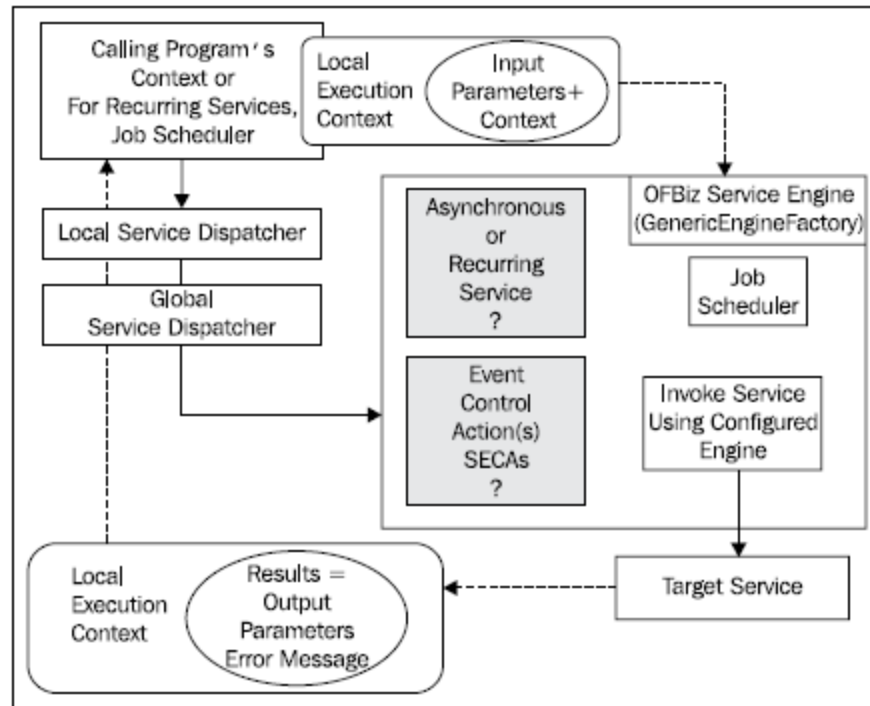
OfBiz Services are:

- Multiple invocation methods including: inline or synchronous with the calling program; out-of-band or asynchronous from the caller's processing logic and/or as a scheduled job for recurring execution. OFBiz handles all input, output, and context parameter transfers seamlessly to both Service provider and Service caller.

- Chaining of Services for a true, event-driven Service platform and implementation of complex workflows. Services may be configured to be invoked based on external events or triggers. Once triggered, an OFBiz Service may call other Service(s) based on additional triggering Events and/or conditions. Any combination of Services may be chained together to form Service Event Condition Action(s) or SECAs.
- A fully integrated job scheduler for recurring and single use asynchronous job scheduling. The OFBiz job scheduler handles all the mundane coordination tasks associated with job scheduling, including calendar lookups, frequency, and interval timing.
- Service creation and implementation tools, including selectable input and output validations based on configured parameter types; authentication and authorization checks integrated with OFBiz user login processing, and even localization preservation across Service calls.

The heart of the OFBiz service-oriented implementation is the Service Engine factory. Using a factory pattern, OFBiz provides an easily extendable Service management and invocation tool supporting any number of concurrent Services and any number of third-party execution engines including, but not limited to: Java, Groovy, Javascript, JPython, and the OFBiz "simple" Service (based on the OFBiz Mini-Language.) By offloading Service implementation to programming language-specific engines, OFBiz Services may be written and implemented in any language that suits the developer's fancy.

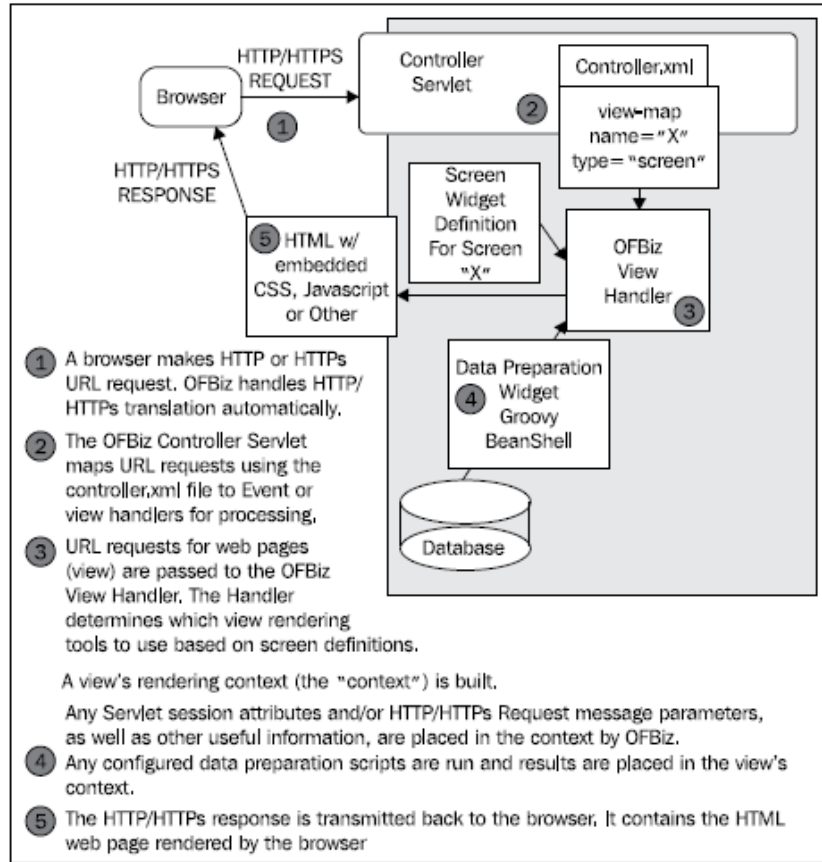
The Service Engine factory may be called from anywhere in the frameworks to handle the details of Service invocation, as shown in the following diagram:



**Fig. 2(m) OFBiz Service Engine**

### 2.8.3 User Interface

The OFBiz screen handler takes Screen widget definitions (as pointed to using controller.xml view-map entries) and creates HTML documents or OFBiz "screens". Screen widget definitions are instructions to the screen handler on how to build screens by merging the specified actions with HTML. Screen widget definitions are contained in XML documents that, by convention, have their own directory called the widget directory within a containing Component. Each Screen widget "definition" document has one or more Screen widget "definitions". A definition is nothing more than an XML element with child elements and attributes that describe how a screen is to be constructed. Each definition starts with a screen element that has a name attribute. Following the screen element are one or more section elements. Within section elements, there may be one or more widgets and/or actions elements as shown. Any widget element may contain any number of nested elements within it, including file location pointers to other Screen, Form, Menu, and Tree widget definitions as shown

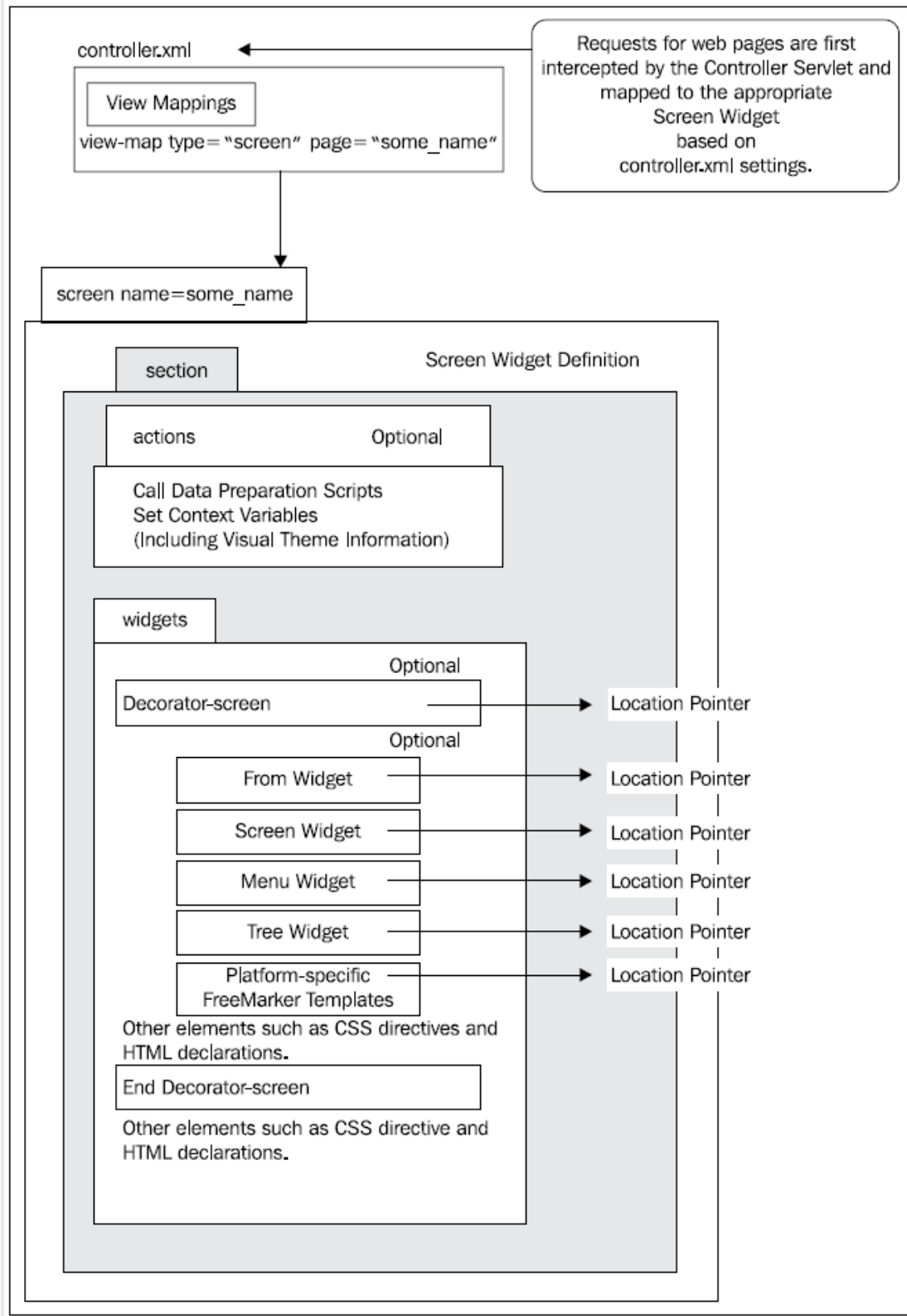


**Fig. 2(n) User Interface Designing in Ofbiz**

The following table lists some commonly used HTML and the equivalent XML statements:

**Table 1. Markup Language**

TYPES	HTML	XML
Plain text	Here IS Some Text	<label text="Here IS SomeText"/>
Markup tags	<h1></h1>	<label style="h1"></label>
	<p></p>	<label style="p"> </label>
DIVs	<div> </div>	<container></container>
HTML hyperlink	<a href= "a URL">Click</a>	<link target="a URL"text="Click"/>
Tree	<ul class="basic-tree"><li> <a class="leafnode"> </a> </li></ul>	<tree name="mytree" root-node-name="first-node" ><node name="first-node"></node></tree>



**Fig. 2(o) OfBiz Screen Widget Architecture**

#### **2.8.4 OfBiz Database Entity Engine**

Secure and reliable data storage is the key business driver behind any data management strategy. That OFBiz takes data management seriously and does not leave all the tedious and error-prone data management tasks to the application developer or the integrator is evident from the visionary design and implementation of the Entity Engine.

The Entity Engine is a database agnostic application development and deployment framework seamlessly integrated into the OFBiz project code. It handles all the day-to-day data management tasks necessary to securely and reliably operate an enterprise. These tasks include, but are not limited to support for:

- Simultaneously connecting to an unlimited number of databases
- Managing an unlimited number of database connection pools
- Overseeing database transactions
- Handling database error conditions

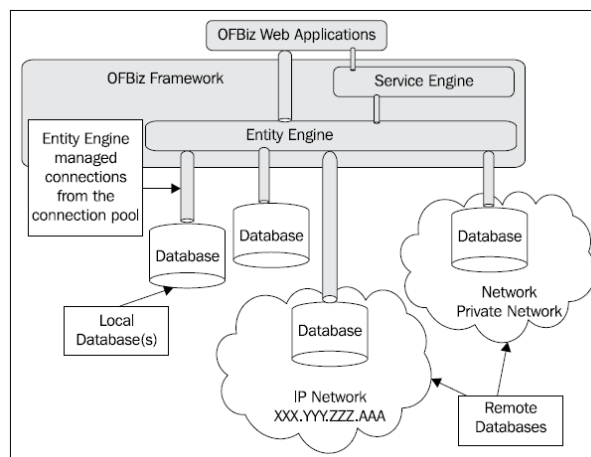
The true power of the Entity Engine is that it provides OFBiz Applications with all the tools, utilities, and an Application Programming Interface (API) necessary to easily read and write data to all configured data sources in a consistent and predictable manner without concern for database connections, the physical location of the data, or the underlying data type.

To best understand how to effectively use the Entity Engine to meet all your data storage needs, a quick review of Relational Database Management Systems (RDBMS) is in order :

- RDBMS tables are the basic organizational structure of a relational database. An OFBiz entity is a model of a database table. As a model, entities describe a table's structure, content format, and any applicable associations a table may have with other tables.
- Database tables are further broken down into one or more columns. Table columns have data type and format characteristics constrained by the underlying RDBMS and assigned to them as part of a table's definition. The entity model describes a mapping of table columns to entity fields.

- Physically, data is stored in tables as one or more rows. A record is a unique instance of the content within a table's row. Users access table records by reading and writing one or more rows as mapped by an entity's model. In OFBiz, records are called entity values.
- Keys are a special type of field. Although there are several types of keys, OFBiz is primarily concerned with primary keys and foreign keys. A table's primary key is a column or group of columns that uniquely identifies a row within a table. The value of the primary key uniquely identifies a table's row throughout the entire database.
- A foreign key is a key used in one table to represent the value of a primary key in a related table. Foreign keys are used to establish unique and referentially correct relationships between one or more tables.
- Relationships are any associations that tables may have with one another.
- Views are "virtual" tables composed of columns from one or more tables in the database. OFBiz has a similar construct (although it differs from the traditional RDBMS definition of a "view") in the view-entity.

The Entity Engine provides all the tools and utilities necessary to effectively and securely access an unlimited number of databases regardless of the physical location of the data source, as shown in the following figure:



**Fig. 2(p) OfBiz Entity Engine**

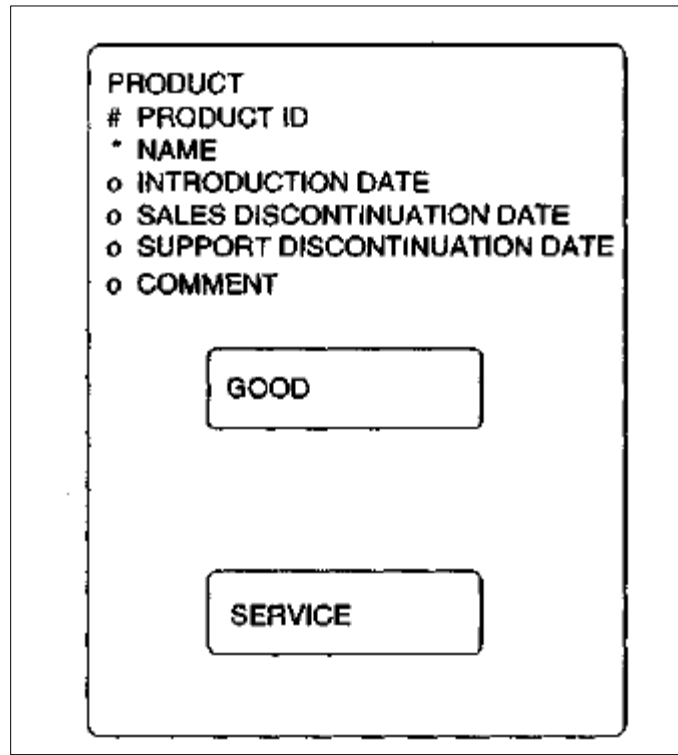


**3.1 Catalog Manger****3.1.1 Introduction**

This chapter focuses on the products that parties produce and use. Products are defined as goods or services that were, are, or will be sold by the enterprise. Goods are products that are more tangible in nature and generally created in advance for sale. Services are products that involve the use of parties' time and are less tangible in nature. Every organization needs to know a great deal of product information on a regular basis. The product model stores common product information regardless of whose products they are. This product model is therefore more flexible, stable, and understandable because product information is modeled only one time, regardless of whether it is the enterprise's products, competitors' products, or suppliers' products. The product model includes products that the enterprise provides, products from suppliers, and products that competitors provide. Some of the information may be independent of the supplier, such as the description, category, and features of the product. Some of the information about the products, such as the availability and pricing of products, may depend on the supplier of the product. The classification of products is a key aspect of maintaining product information. Products are often classified many ways—by product line, by model, by product grade, by industry segments, and by various other product categories. PRODUCT may be classified in one or more PRODUCT CATEGORYs to group products together that may be useful for several purposes, such as catalog organization, sales analysis, listings, or other types of product analysis. The PRODUCT CATEGORY can include more than one PRODUCT, and therefore the associative entity, PRODUCT CATEGORY CLASSIFICATION describes which products are in which category.

### 3.1.2 Database

#### 3.1.2.1 Product Definition



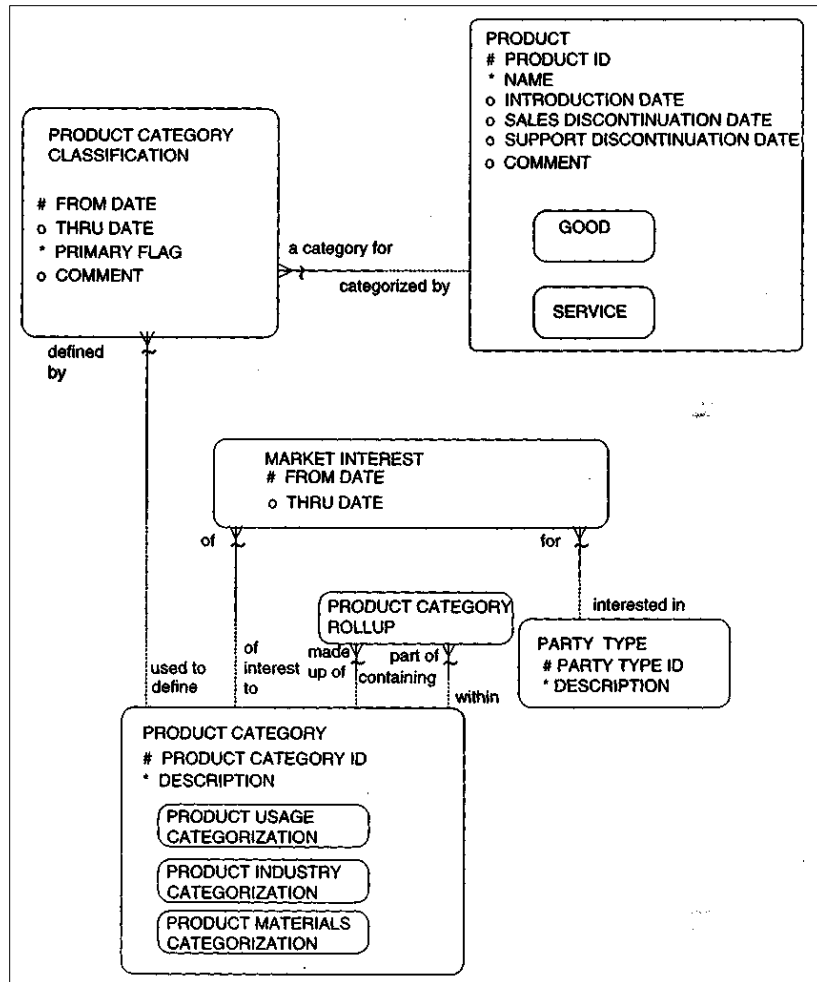
**Fig. 3(a) Product Definition**

Figure shows an entity called **PRODUCT** that models all products including the products the enterprise sells, products from suppliers, and competitors' products. The model shows that the key is **product id**, which is an identifier for the product. The attributes of **PRODUCT** are name, which uniquely describes a product; **introduction date**, stating when the product was first available to be sold; **sales discontinuation date**, which documents when the product will not be sold any more by the manufacturer; **support discontinuation date**, which states when the product will no longer be supported by the manufacturer; and a **comment**, which documents particular descriptions or notes relating to the product. Products include *both* tangible goods, which are called **GOODs**, and nontangible offerings, which are called **SERVICES**.

#### 3.1.2.2 Product Category

The classification of products is a key aspect of maintaining product information. Products are often classified many ways—by product line, by model, by product grade, by industry segments, and by various other product categories. Often, these classifications are shown as

separate entities in the data model. The issue of showing these product categories separately is that other information may be based on any of these categories and should be related to the super type of PRODUCT CATEGORY to provide more flexibility. For instance, targeting market interest in products may be based on several product categories. Another example is that product pricing may be based on and related to many product categories. Figure illustrates that a PRODUCT may be classified in one or more PRODUCT CATEGORYs to group products together that may be useful for several purposes, such as catalog organization, sales analysis, listings, or other types of product analysis. The PRODUCT CATEGORY can include more than one PRODUCT, and therefore the associative entity, PRODUCT CATEGORY CLASSIFICATION describes which products are in which category. For example, all pens, pencils, paper, notebooks, desk sets, and diskettes are classified into the category of office supplies. Product categories may change over time; therefore, the PRODUCT CATEGORY CLASSIFICATION has the attributes **from date** and **thru date**, which state when the product was classified into its grouping. PRODUCT CATEGORYs may be made up of other PRODUCT CATEGORYs and hence the recursive entity PRODUCT CATEGORY ROLLUP that allows each product category to be made up of many other categories as well as a subcategory to be in a different parent category. To assist in sales forecasting and prospecting, the entity MARKET INTEREST is included in this model. It is an intersection entity linking information about PARTY TYPE and PRODUCT CATEGORY that allows an enterprise to record the category of products for which particular types of parties may be interested. If the PARTY TYPE includes specific industries or industry segments as types and these have been associated with actual parties, then an enterprise can easily identify organizations within segments of a target industry that may be interested in types of products or services. These organizations could then be the focus of a new sales campaign for those types of products. Because interests change over time, the attributes **from date** and **thru date** are also included.



**Fig. 3(b) Product Category**

### 3.1.2.3 Product Pricing

there are several aspects to pricing a product: the base price for which the organization sells the product, various discounts applied against the base price such as quantity breaks, surcharges such as freight and handling charges, and the manufacturer's suggested price. The PRICE COMPONENT stores these aspects of prices for each supplier's products.

### 3.1.2.4 Pricing Subtypes

This entity is broken down into two non-mutually exclusive sets of subtypes. One subtyping that occurs is the subtypes of BASE PRICE, which has the starting price for the product, DISCOUNT COMPONENT, which stores valid reductions to the base price, SURCHARGE COMPONENT, which adds on possible charges, and MANUFACTURERS SUGGESTED PRICE. Another subtyping that categorizes types of prices is that the PRICE COMPONENT may be a ONE TIME CHARGE (such as for buying a good), a RECURRING

CHARGE (an ongoing charge such as a monthly fee for a standard type of service), or a UTILIZATION CHARGE, which is a price component based on billing for usage of a product. The RECURRING CHARGE is based on per TIME FREQUENCY MEASURE (per hour, per day, per month), which is a type of UNIT OF MEASURE. The UTILIZATION CHARGE is based on a UNIT OF MEASURE, such as per a certain **quantity** of "internet hits" to describe the charge for Web hosting services.

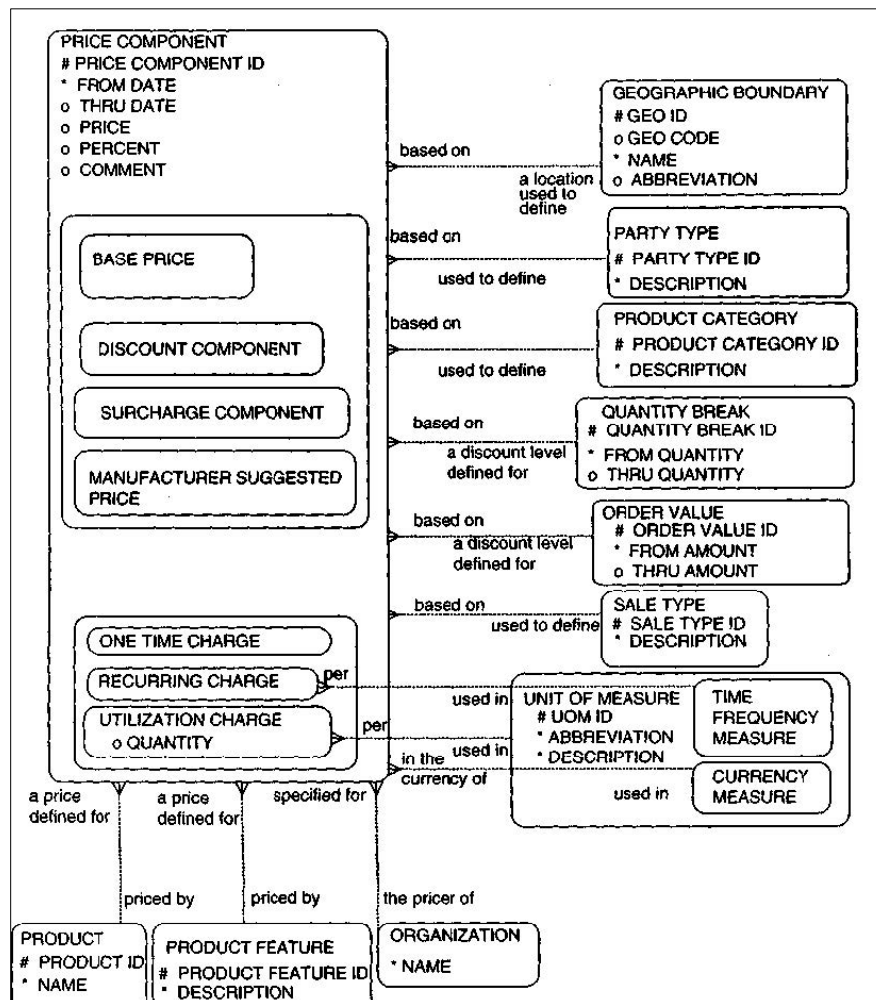


Fig. 3(c) Product Pricing

### 3.1.2.5 Product Feature

In Figure a PRODUCT FEATURE entity is used to define these ways in which the product may be modified or tweaked. Because a PRODUCT FEATURE may be used in many products and each product may have many features, the PRODUCT FEATURE APPLICABILITY maintains which PRODUCTS may be available with which PRODUCT

FEATURES. The subtypes, STANDARD FEATURE, REQUIRED FEATURE, SELECTABLE FEATURE, and OPTIONAL FEATURE allow the capability to specify if the features come as part of the standard configuration of the product, if the feature is mandatory as part of the product, if the feature needs to be selected (e.g., color), or if the feature is an optional component. The UNIT OF MEASURE entity helps define the product in terms of the type of measurement for the product. It also helps further define the DIMENSION subtype of PRODUCT FEATURE. The UNIT OF MEASURE CONVERSION allows different units of measures to be converted to a common unit in order to assess inventory levels of a type of good.

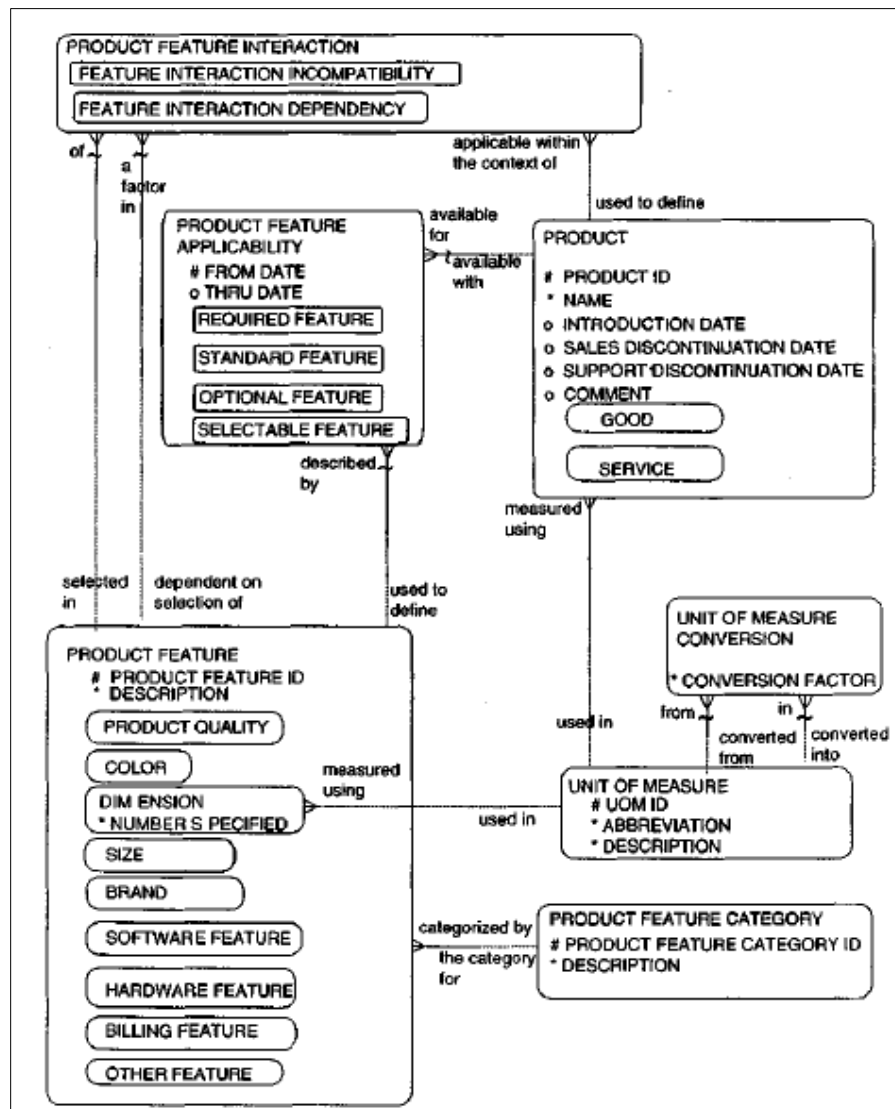


Fig. 3(d) Product Features

### 3.1.2.6 Summary Overall Product Model

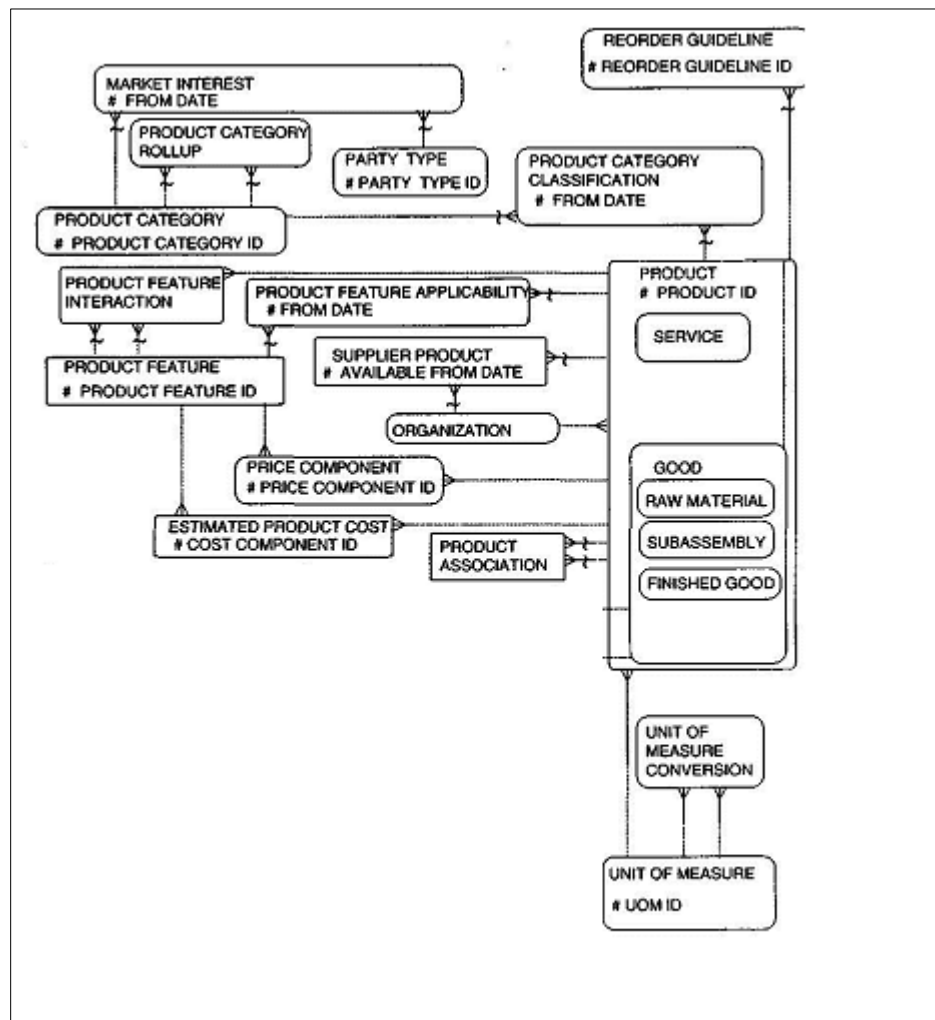


Fig. 3(e) Overall Product model

## **3.2 HR manager**

### **3.2.1 Introduction**

One critical section of the business is Human Resource, or HR. Without human resource, an enterprise cannot employ and use the key resource it needs to stay in business. Information that an enterprise may want to keep includes the following:

1. Who is employed, and what is the history of employments?
2. What positions exists in the company?
3. Are they filled? If so, who has what position and what are his or her responsibilities?
4. Who reports to whom?
5. What is the rate of pay for these positions?
6. Who received raises and when?
7. What benefits does the enterprise provide and to whom?
8. What is the cost of these benefits?
9. What is the status of employment application?
10. What are the skills of employees?
11. What is the performance of employees?
12. What are the preference, deduction, and payroll information needed to process payroll?
13. What applicant have there been, and how many of them have turned into employee?
14. What has been the rate of turnover and causes of turnover?

### **3.2.2 Database Schema Human Resource**

#### **3.2.2.1 Employment Schema**

Figure provides a data model to maintain employment information. EMPLOYMENT is a subtype of PARTY RELATIONSHIP and represents a relationship between the PARTY ROLES of EMPLOYEE and INTERNAL ORGANIZATION. This assumes that the enterprise is interested only in tracking its own employees. If the model needs to accommodate employees of external organizations as well, then the EMPLOYMENT entity should be between the EMPLOYEE and EMPLOYER instead. The EMPLOYMENT inherits



the properties of PARTY RELATIONSHIP because it is a type of relationship between two parties. It inherits the fact that

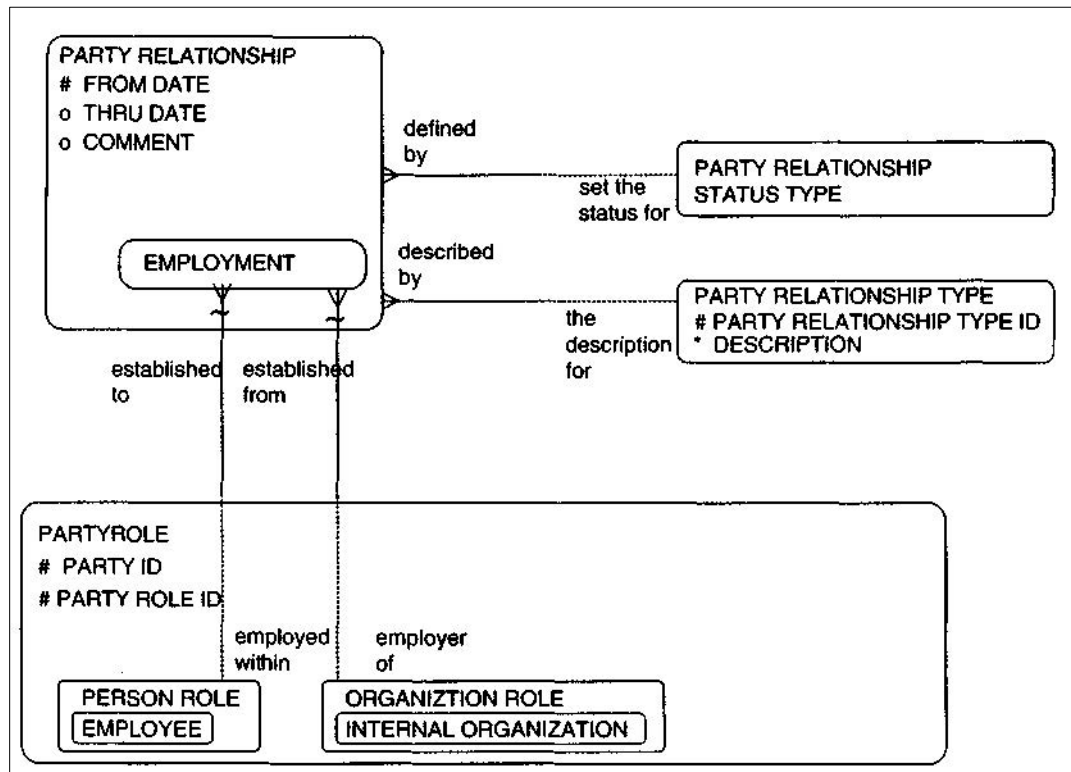


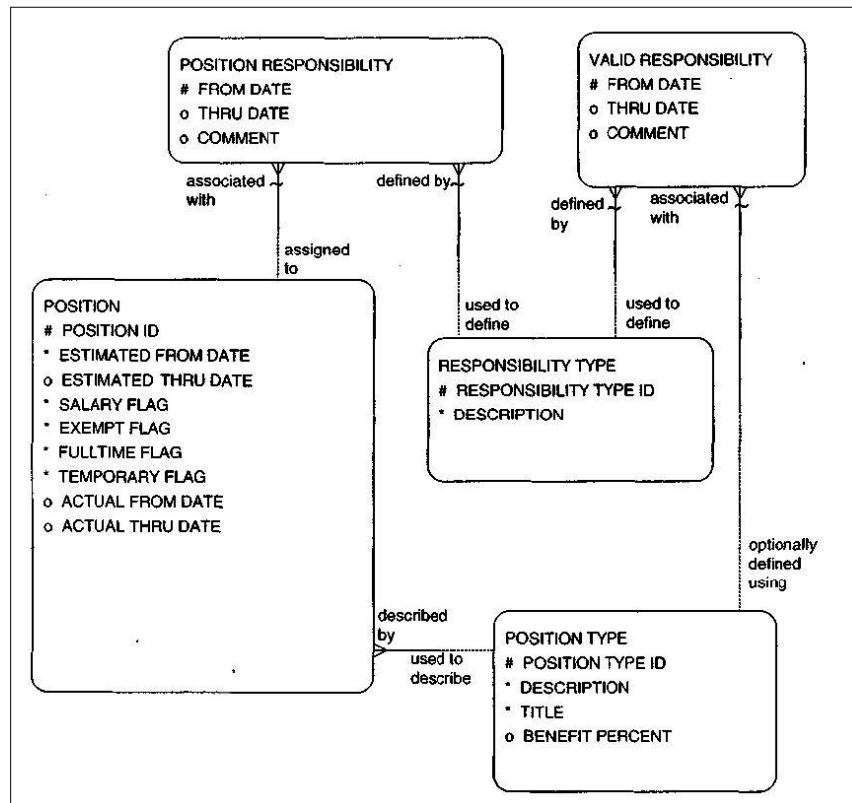
Fig. 3(f) Employment

it has a **from date**, which would represent the start of the relationship or the hire date, as well as the **thru date**, which would be the last date of the employment. It inherits the relationship to **PARTY RELATIONSHIP STATUS TYPE** because each employment will have a status such as "active," "inactive," "pending," "terminated," and so on.

### 3.2.2.2 Position Definition

A **POSITION** represents a job slot in an enterprise that can be occupied by more than one person over time. In Figure the entities used to define a position are modeled. The **POSITION** entity contains the basic information to track about any given job slot. The **POSITION TYPE** entity provides information for further defining and categorizing a job, the entities **RESPONSIBILITY TYPE**, **VALID RESPONSIBILITY**, and **POSITION**

RESPONSIBILITY provide a mechanism for assigning and tracking what are the possible and actual responsibilities for any position.



**Fig. 3(g) Position Definition**

TYPE. Using these entities, additional groupings of types of positions can be done. Because it is possible that a POSITION TYPE could be reclassified over time, the POSITION TYPE CLASS entity with its attributes of **from date** and **thru date** are included to support the many-to-many relationship between POSITION TYPE and POSITION CLASSIFICATION TYPE.

### 3.2.2.3 Position Fulfillment and Tracking

Figure shows a model that will allow the enterprise to track each person's position history within the organization. It includes entities for tracking POSITION FULFILLMENT and POSITION STATUS TYPE. The model needs the entities POSITION and PERSON for context. ORGANIZATION is also included to provide information on the hiring company.

#### 3.2.2.4 Position Fulfillment

A person can, of course, occupy more than one position either over time or at the same time. Conversely, a position may be filled by more than one person over time (and even at the same time through job sharing). The POSITION FULFILLMENT entity provides a very flexible way to retain the history of this activity. The attributes **from date** and **thru date** will allow the enterprise to keep historically accurate information about this data. It is a convenient and effective way to resolve the many-to-many relationship that really exists between PERSON and POSITION.

#### 3.2.2.5 Position Status Type

The POSITION STATUS TYPE identifies the current state of a position. When a position is first identified, it is in a state of "planned for." When the enterprise decides to pursue fulfillment of the position, it may then change to a state of "active" or "open." If the enterprise then decides

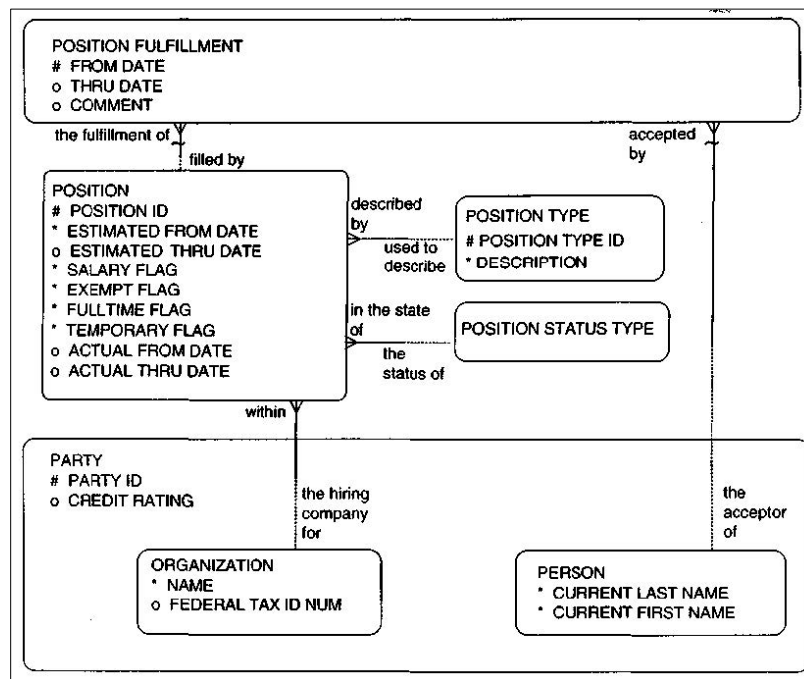


Fig. 3(h) Position Fulfillment

That it no longer needs that position; the status may be "inactive" or "closed." A "fulfilled" status would not be a value because this information can be derived from the POSITION FULFILLMENT entity.

### 3.2.2.6 Position Reporting Structure

The model in Figure shows the entity POSITION REPORTING STRUCTURE, which links POSITION back to itself. The attributes **from date** and **thru date** are provided to allow for tracking organizational changes through time, as previously discussed. The **primary flag** attribute is included to help model flexible, matrix-type structures. In these cases, certain positions may report to more than one position at the same time. This indicator allows the enterprise to indicate which reporting relationship is the overriding one.

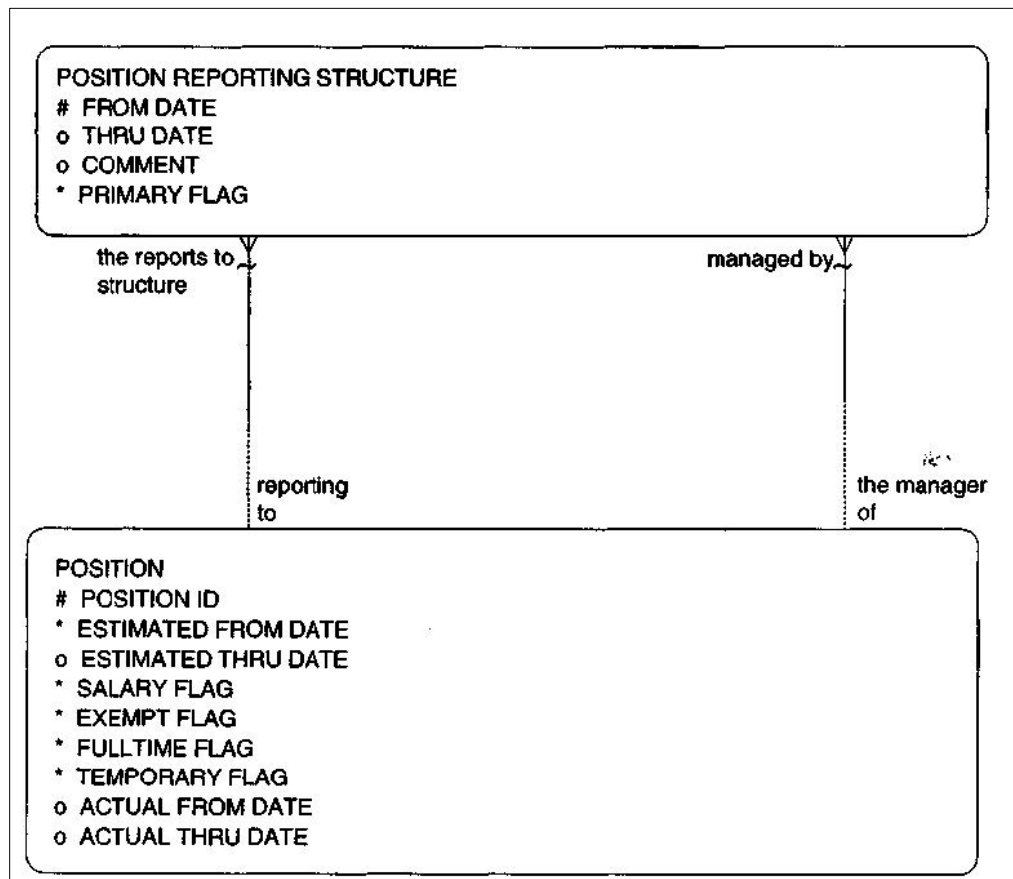


Fig. 3(i) Position Reporting Structure Schema

### 3.2.2.7 Position Type Rate

The POSITION TYPE RATE and RATE TYPE entities were defined to provide the costs and standard rates for various types of positions in order to capture appropriate cost estimates for work efforts. This section will expand the use of this entity to incorporate other types of position rates for tracking standard pay rates and ranges. The POSITION RATE TYPE entity

may be used to record the allowable or acceptable salary and salary ranges for a particular position type. This information could be used by managers during the hiring process when negotiating salary. A **from date** and **thru date** are included so that a history of these standards can also be kept. The relationship to RATE TYPE provides the ability to record reference information such as "highest pay rate," "lowest pay rate," "average pay rate," and "standard pay rate," which represent different types of pay rates that can be captured for each type of position. This would indicate such things as the upper limit, average amount paid, lower limit, and the default, standard amount of pay for a type of position. The relationship to PERIOD TYPE will allow an enterprise to define various pay period types for which rates can be recorded. Examples of PERIOD TYPE include "per year," "per week," "per month," and so on. This relationship is part of the primary key to allow an enterprise to record information for multiple period types.

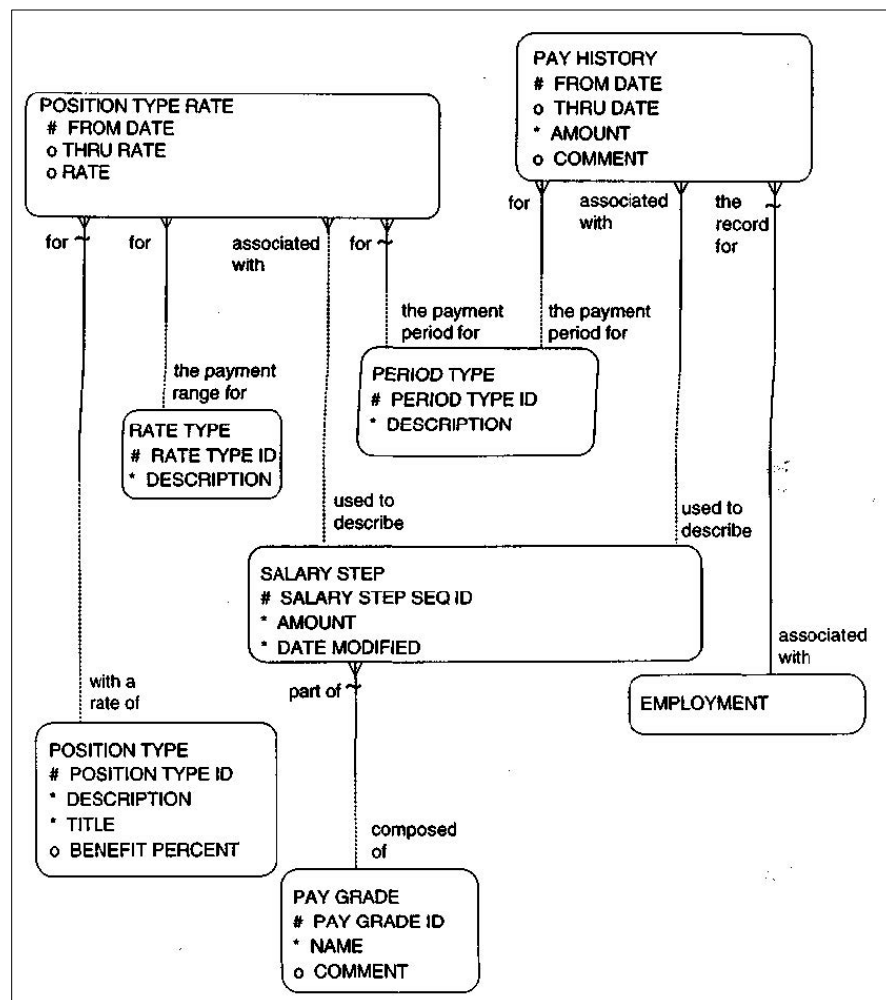


Fig. 3(j) Salary Determination

### 3.2.2.8 Overall Human Resource Schema

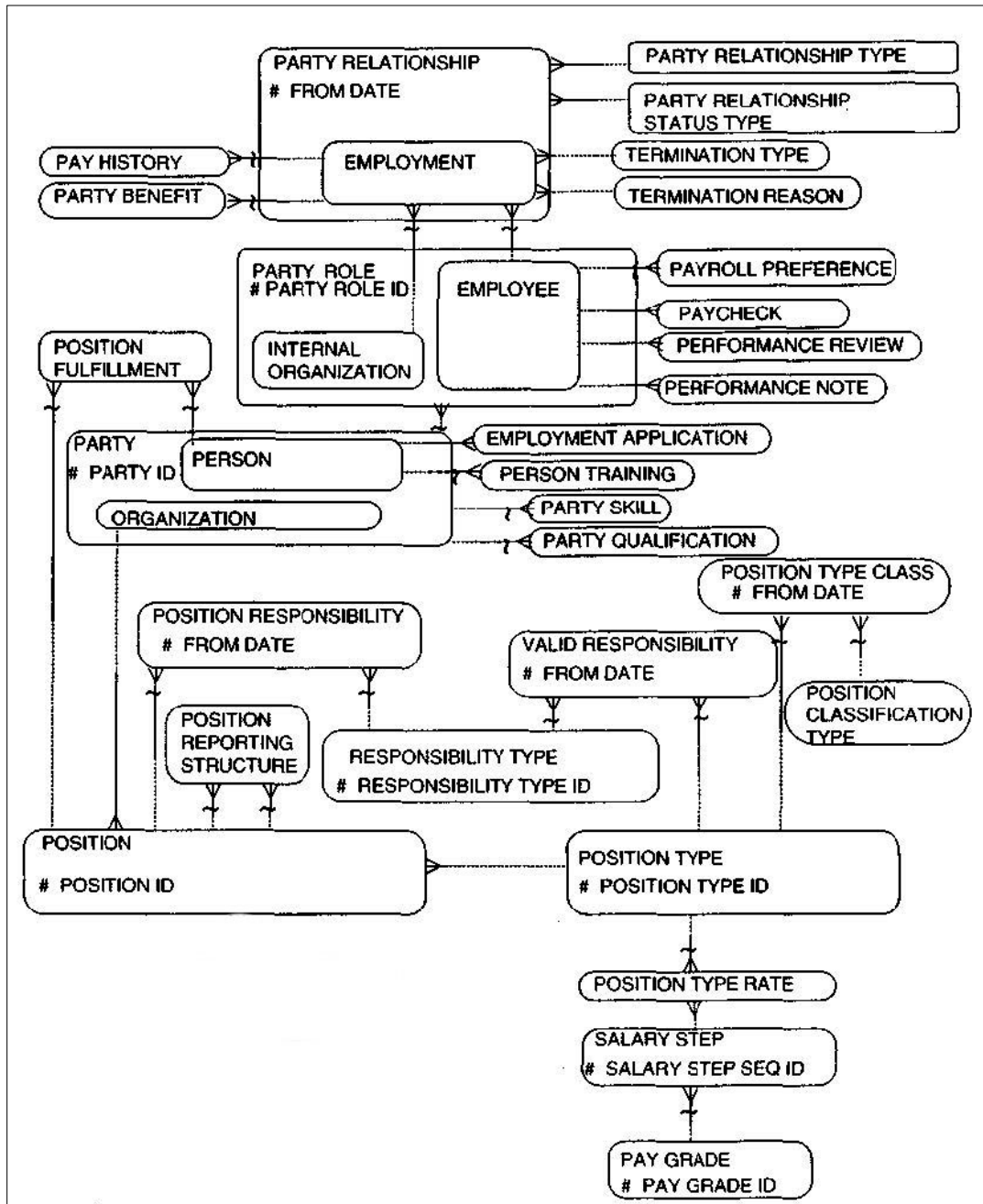


Fig. 3(k) Human Resource Schema

## **3.3 Order Manager**

### **3.3.1 Introduction**

This module focuses on how parties obtain these products—in other words, the ordering of products. Businesses need information to answer many questions about orders; they need to know the terms associated with each order. For instance:

1. When the expected delivery is time, and is there consequences for late delivery?
2. Who is responsible for paying for the order?
3. What is the negotiated price for each product that is ordered?
4. What people and organizations are involved in the order?
5. Who placed the order? To whom is the order being shipped?
6. Was there an approved requirement for the order?
7. Were there previous quotes for the order?
8. Were there requests to many vendors to bid for the order?

### **3.3.2 Database Schema**

#### **3.3.2.1 Overall order model.**

These models incorporate both sales and purchase order perspectives and cover services as well as goods. Orders go through a process, beginning with a requirement or a need for a product(s). The requirement may be directly fulfilled by an order, or it may lead to a request to suppliers, a quote, and then an order.

#### **3.3.2.2 REQUEST AND REQUEST ITEM**

It contains request made by organization or person. Each request contains its request id, and request date. This request is in relation with request item which contains unique request item sequence id of organization or person by its Date, Quantity and Description to maintain each request in order.

#### **3.3.2.3 QUOTE AND QUOTE ITEM**

Before directly request for an order organization or person may ask for the quotation of its product which they request. Quote contains its unique quote id for each organization or person. This Quote is relate with quote item which contains unique Quoted Item Sequence Id for maintain sequence order of every organization

#### **3.3.2.4 ORDER ITEM**

It specifies the product(s) that are to be ordered. It contains Order Item Sequence Id, Quantity, Unit Price, Estimated Delivery Date, Item Description & Comment. It relates to Product & Product Features. Each ORDER ITEM may be for one and only one Product or may be for one and only one Product Features.

#### **3.3.2.5 PRODUCT**

Product can be specific goods or services. Each product contain unique Product ID, Name, Date, Discount, Comment

#### **3.3.2.6 PRODUCT FEATURES**

Each product contains its own feature with its unique Product Feature Id with Color, Size, Software Feature, Billing Feature.

#### **3.3.2.7 Standard Order Model**

Most organizations model orders using the standard data model that is shown throughout many textbooks on data modeling. Figure illustrates this standard data model. A SUPPLIER is related to one or more PURCHASE ORDERS, which have PURCHASE ORDER LINE ITEMS that relate to PRODUCT. This model is similar for sales. A CUSTOMER can be related to one or more SALES ORDERS, which have SALES ORDER LINE ITEMS that relate to specific PRODUCTS. The attributes of sales orders and purchase orders are very similar, if not the same. An order is an order; the only real difference between a sales order and a purchase order is a matter of perspective. In other words, the information on the seller's sales order should, for the most part, correspond to the information on the purchaser's purchase order. The only reason that the attributes may be different is that the two parties may require different information depending on whether one is the seller or the buyer. For example, the enterprise taking the sales order may need to know the commissions for the salespeople involved in the order, whereas when a purchase order is given to a vendor, the purchaser is usually not privileged enough to know what the commission is for the salespeople. As Figure shows, the relationships and data structures between sales and purchase orders are very similar. By having two separate models for both of these types of orders, the model does not recognize the similar properties of sales and purchase orders and developers cannot take advantage of common applications to handle both sales and purchases. For instance, if the structures are the same, developers can create common routines against the same data structures. There could be common routines to look up the terms of an order, to monitor the status of the order against the expected delivery date, or to calculate line-item price extensions.

There is another embedded assumption in this standard order data model:

Only one organization is involved in the sales order model, namely the customer, and only one organization is involved in the order in the purchase order model, namely the supplier. Most systems model information from an "I" perspective. This perspective states that the model doesn't have to indicate the party receiving sales orders or placing purchase orders; it's obviously us (meaning the enterprise building the system)! The problem with this perspective



is that there may be several internal organizations within the enterprise that may receive a sales order or place a purchase order and this information needs to be recorded for each order. Even if an organization is a small business, there may be a broker who takes a sales order or the organization may grow later on to include subsidiaries, divisions, or other related internal organizations. There are actually many parties involved in the order who could be involved in many roles. There could be a placing customer, delivery customer, bill-to customer, installation customer, person taking the order, organization that books the order, and similar roles for purchase orders.

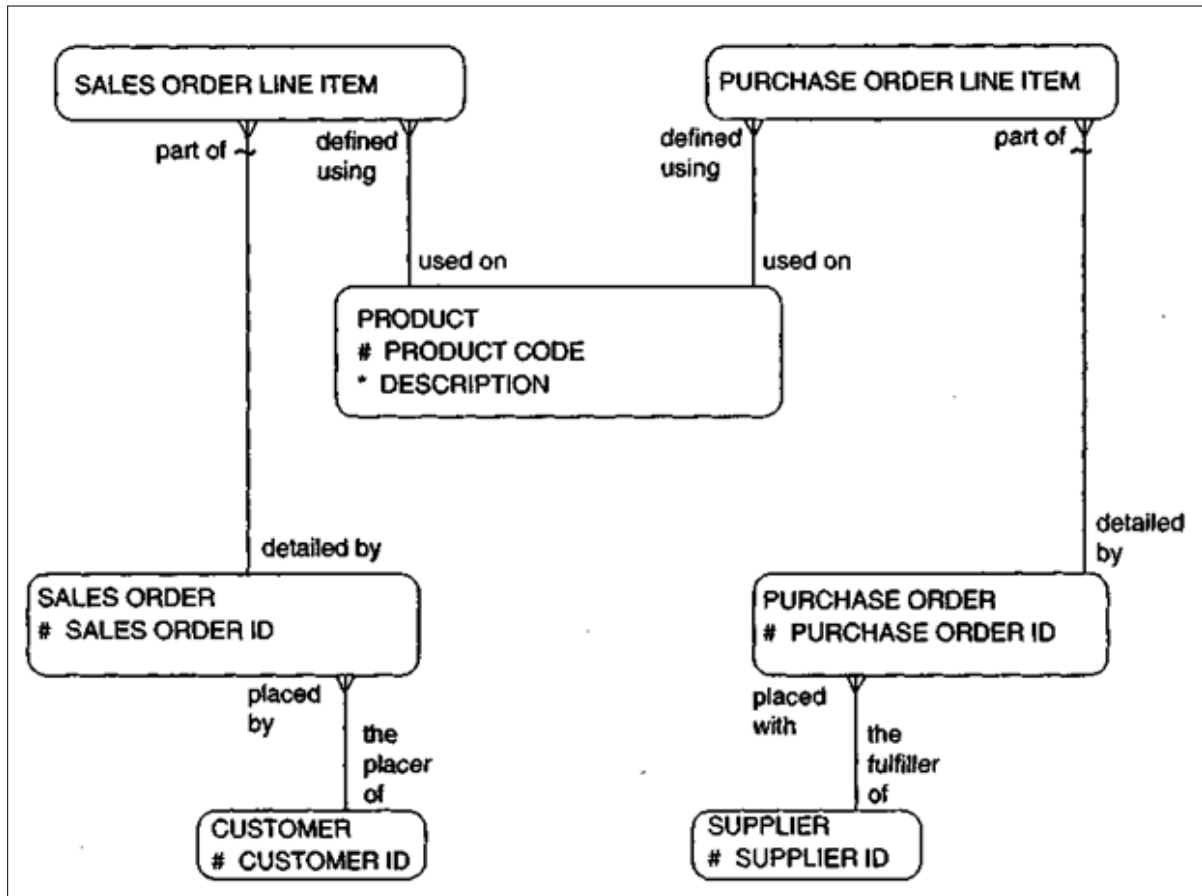


Fig. 3(l) Standard order model.

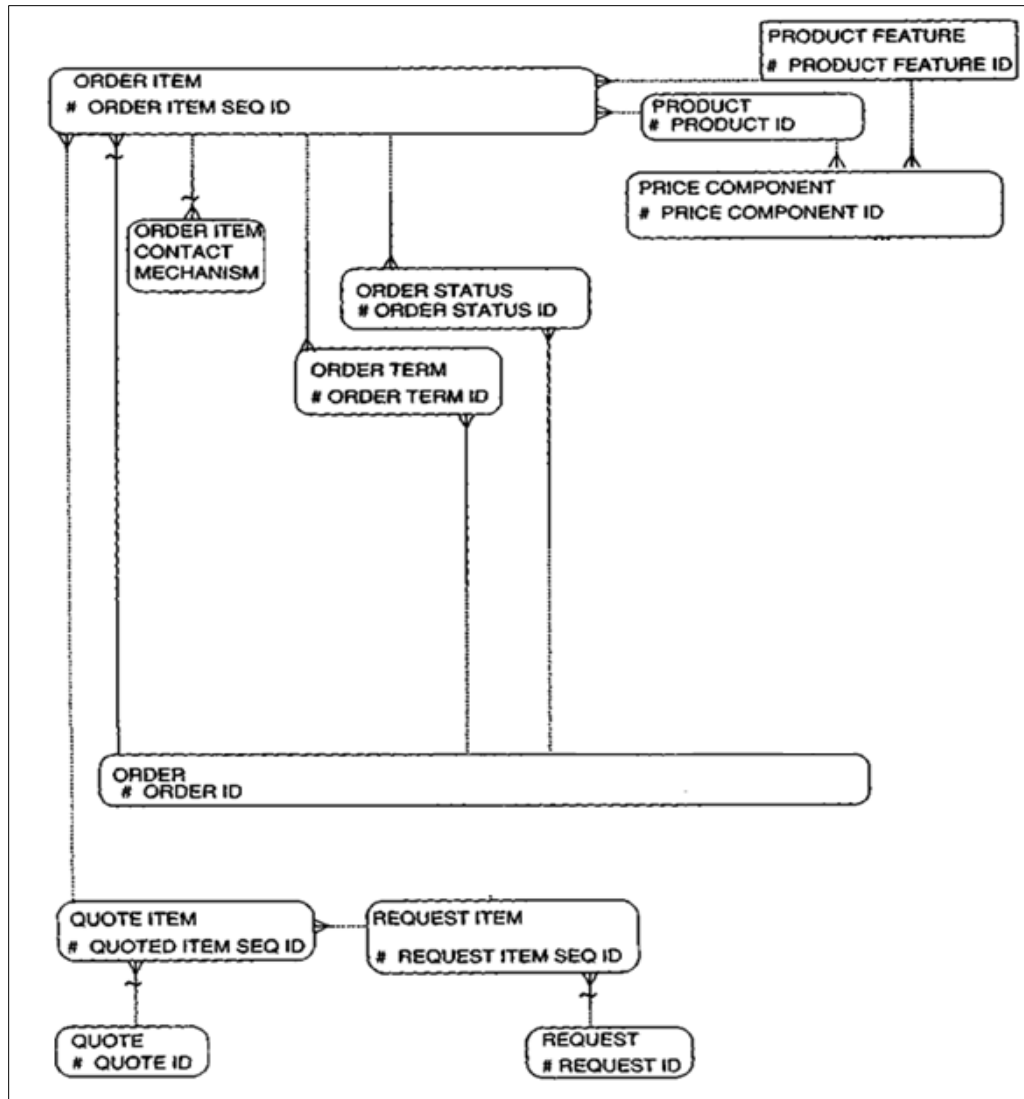


Fig. 3(m) Overall order model.

## **3.5 CRM(Trouble Ticketing) Manager**

### **3.5.1 Introduction**

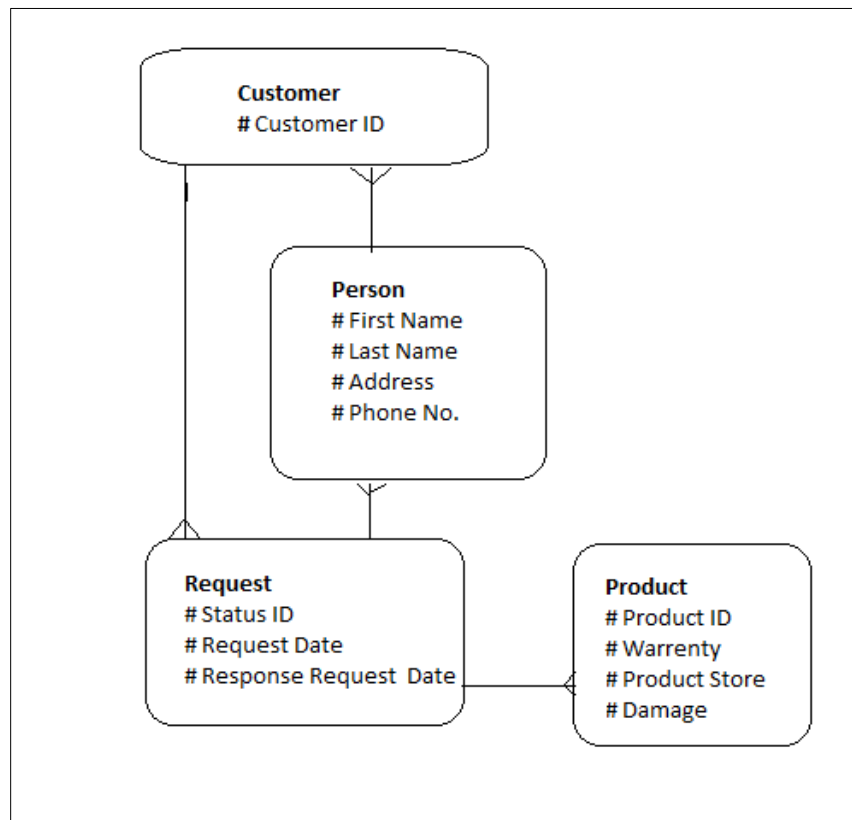
Customer Relatoin Manager will handle the taks and maintain the relations with customers.

CRM module deals with these issues.

1. Register Customer Complaints?
2. Check for product is valid for warranty?
3. Estimate Delivery Time for Customer Request.

### **3.5.2 Database**

#### **3.5.2.1 Database Schema**



**Fig 3(n). CRM Schema**

Person contains all the details about person or organization. It contains name, address, phone no to maintain relation between system.it relates with customer. Here, Customer maintains information of person or organization with unique Customer Id. This customer is related with

request which maintains request of each person or organization with unique status Id. Request date maintains at which date request is made by person or organization. Response Request Date maintains delivery of products which shortlisted by date. This request relates with product each products differentiate with its unique product id.

### **3.5.3 Functionalities**

- Find Customer and details
- Create new Request of complaint
- Manage the Request
- Finding Sales order by department
- Finding the Purchase Order by Department

### 4.1 Startup Ofbiz

To start the Ofbiz run the file **startofbiz.bat** in windows and in linux opens the terminal and navigates to the Ofbiz installation directory and then enter the command **./startofbiz.sh**. This will take 20-30 approx. time to start up the Ofbiz server and then navigate to any of the application manager and start using Ofbiz.

### 4.2 Create Product

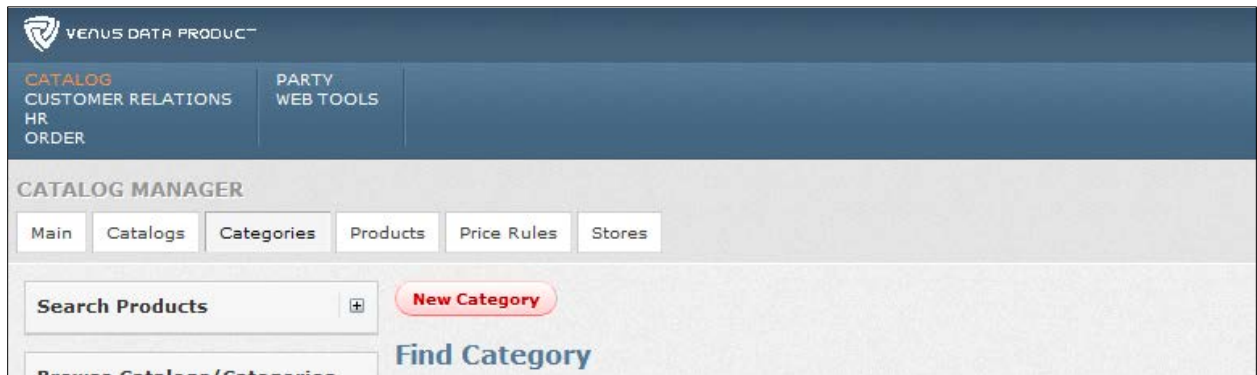
From main catalog page select Create New Catalog.

Fig.4(a) Create Catalog

Add appropriate information of new catalog and update it.

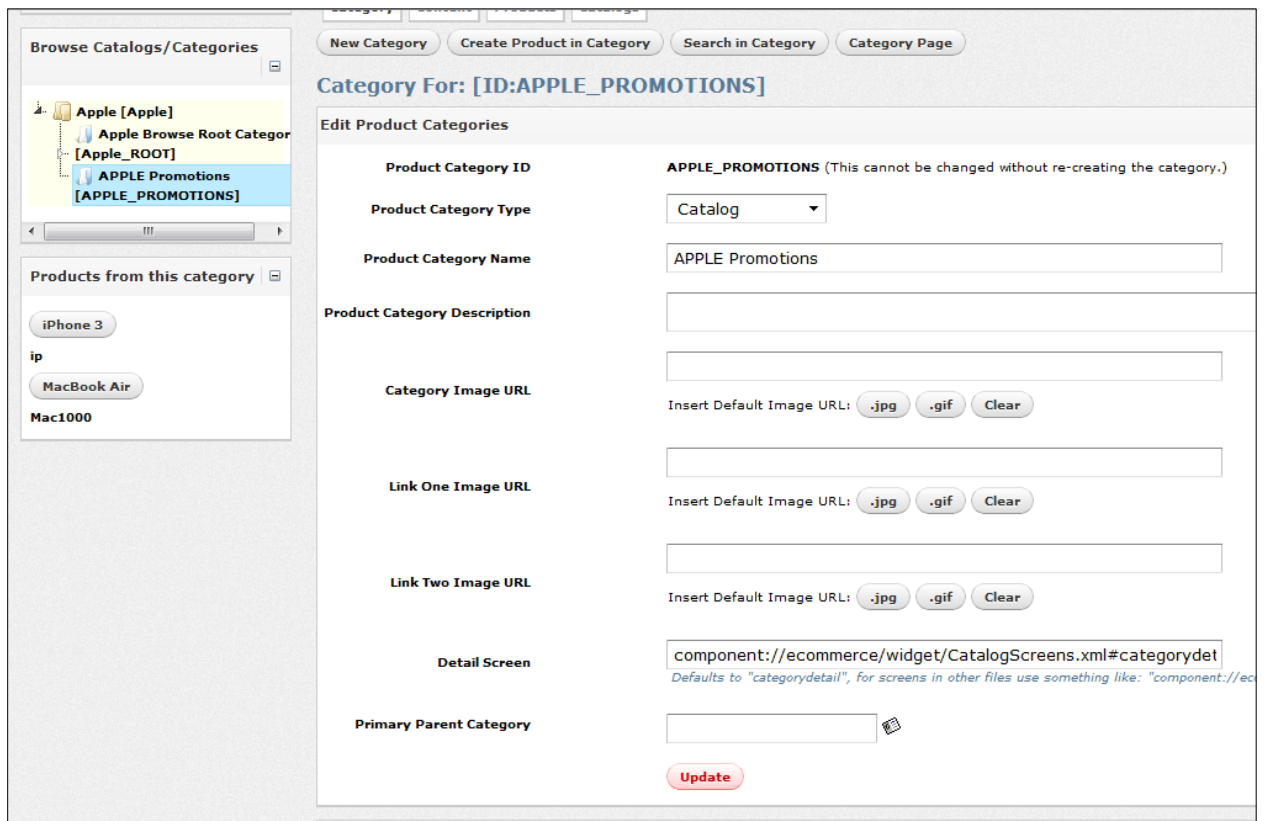
Fig. 4(b) Edit Product Catalog

Select category option in main menu bar and create new category in that catalog.



**Fig.4(c) New Category**

Fill appropriate information of new category and update it.



**Fig.4(d) Update Category**

Select Create New Product in this category from category menu and after giving a name click on check existing.

The screenshot shows the 'CATALOG MANAGER' interface. The top navigation bar includes 'CATALOG', 'CUSTOMER RELATIONS', 'HR', 'ORDER', 'PARTY', and 'WEB TOOLS'. The 'CATALOG' section is active. Below the navigation bar, there are tabs for 'Main', 'Catalogs', 'Categories', 'Products', 'Price Rules', and 'Stores'. The 'Categories' tab is selected. On the left, there is a 'Search Products' field and a 'Browse Catalogs/Categories' section showing a tree structure with 'Apple [Apple]' selected. On the right, there are buttons for 'New Category', 'Create Product in Category' (highlighted in red), 'Search in Category', and 'Category Page'. Below these buttons, the text 'Create Product in Category For: APPLE Promotions [ID:APPLE\_PROMOTIONS]' is displayed, along with a '[Back to Edit Category]' link.

**Fig.4(e) Create Product**

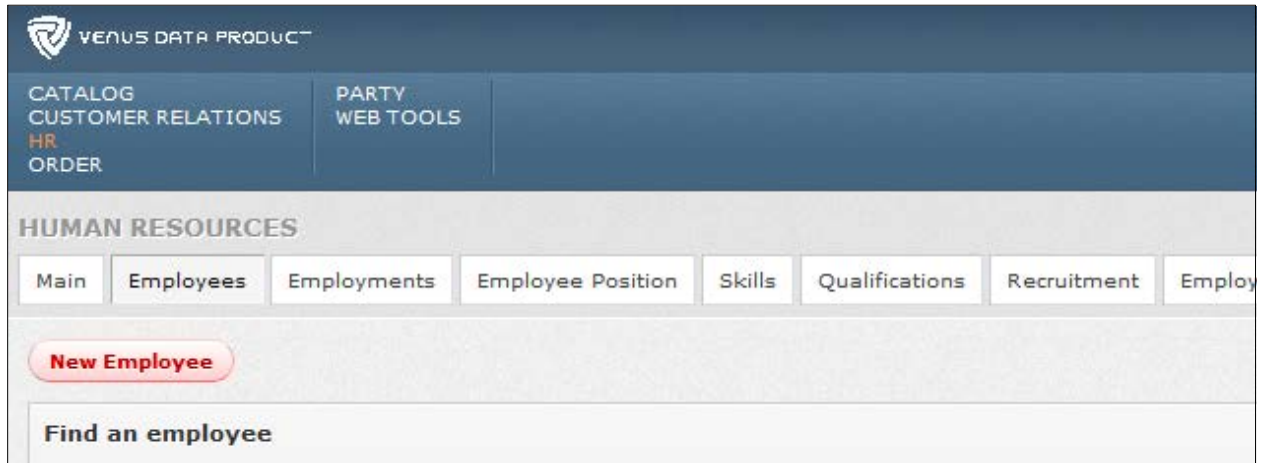
Fill appropriate information of new product and update it.

The screenshot shows the 'Edit Product' form for 'iPhone 3 [ID:ip]'. The left sidebar shows 'Browse Catalogs/Categories' with 'Apple [Apple]' selected and 'Products from this category' showing 'No category specified.' The main form has tabs for 'Agreements', 'Accounts', 'Payment Types', 'Maintenance', 'Meters', and 'Subscriptions'. The 'New Product' tab is selected. The form fields include: 'Product ID' (ip This cannot be changed without re-creating the product), 'Product Type' (Product), 'Wording And Comment' (Internal Name: iPhone 3, OEM Party ID, Comments), 'Virtual Product' (Primary Category, Dates, Inventory, Rate, Amount, Measures, Shipping, ShoppingCart, Miscellaneous), 'Content Info Text' (NOTE: For content options, use the Content tab.), and 'Last Modified By' ([admin] On 2012-04-07 14:55:18.000). An 'Update Product' button is highlighted in red.

**Fig.4(f) Update Product**

### 4.3 Create Employee

Select New Employee from HR main bar and click on New Employee.



**Fig. 4(g) Create New Employee**

Fill appropriate information of new employee and save it.

The screenshot shows the 'New Employee' form. It contains several input fields for employee information. The fields are grouped into sections: 'Title', 'First Name', 'Middle initial', 'Last Name', 'Internal Organization', 'Estimated Start Date', 'Address', 'Address 1', 'Address 2', 'City', 'State', 'Zip/Postal Code', 'Country', 'Primary Phone Number', 'Country Code', 'Area Code', 'Phone Number', 'ext', 'E-Mail Address', and 'Email'. Most fields are highlighted in yellow and marked as 'Required'. The 'Country' field is set to 'India - IND'. A 'Save' button is located at the bottom of the form.

**Fig. 4(h) Add new Employee**



## 4.4 Create Request complaints

In Customer Relation, Select Request from main bar and click on New Request.

The screenshot shows the 'Customer Relation Main Page' with a navigation bar at the top containing 'CATALOG', 'CUSTOMER RELATIONS', 'PARTY', 'WEB TOOLS', 'HR', and 'ORDER'. The 'CUSTOMER RELATIONS' tab is active. Below the navigation bar, there's a 'SERVICES' section with 'Main', 'Find Person', and 'Requests' tabs. The 'Requests' tab is selected, leading to the 'FindRequests' form. This form has two tabs: 'New Request' and 'ReviewedRequestList'. The 'New Request' tab is active, showing fields for 'Status ID', 'Product Id', 'From Party Id', 'Product Store', 'Cust Request Date', and 'Response Required Date'. Each date field has a dropdown menu for 'Equals' and 'Less Than'. A 'Find' button is at the bottom of the form. Below the form is a table titled 'Customer Request List' with columns: 'Cust Request Name', 'Priority', 'Response Required Date', 'From Party Id', 'Status ID', 'Product Id', 'ProductStatus', and 'Reject'. The footer of the page displays the date '24/5/12 12:36 PM', time 'India Standard Time', language 'English (India)', and 'Visual Themes'. It also includes a copyright notice: 'Copyright (c) 2001-2012 The Apache Software Foundation. Powered by Apache OFBiz.'

**Fig. 4(i) Customer Relation Main Page**

Fill appropriate information and submit it.

The screenshot shows the 'Create Request Page' with the same navigation bar as Fig. 4(i). The 'Requests' tab is selected, leading to the 'Request' form. This form has a 'New Request' tab. The 'Request' form contains fields for 'Requesting Party' (10016), 'Cust Request Name' (Mr Vijay), 'Product Id' (iPhone 4s), 'Serial No' (API12345), 'Product Status' (Warranty), 'Web Site Id' (crmSite), 'Request Date' (2012-05-24 15:56:16.55), and 'Description of the problem' (Handset Not Working Properly). There is also a 'PhysicalDamage' field with a value of 'No'. A red 'Submit' button is at the bottom of the form. The footer of the page displays the date '24/5/12 3:56 PM', time 'India Standard Time', language 'English (India)', and 'Visual Themes'. It also includes a copyright notice: 'Copyright (c) 2001-2012 The Apache Software Foundation. Powered by Apache OFBiz.'

**Fig. 4(j) Create Request Page**

- Helping Employee to work in efficient way
- Email Notification to customer support
- Online Store
- Quick and Powerful Advertise of Products and Current offers to customer via Ecommerce
- Staying ahead in the competition
- Efficient management of resources
- Cut the total cost by utilizing the resources in efficient manner
- Weekly Generation of Reports on single click
- Increasing the sales of organization advertising the product on the internet using the ecommerce application

The development of this ERP system is beneficial in many ways for company. The main benefit of this system is that it is cost-effective. It will save the money that is wasted in internal process because of its integrated and collaborative approach. It improve company's current market share in business by helping the employee to work and use available resources efficiently. Good customer support is provided by enabling quick response because of its integration with other module and a consolidated database giving a detail profile of the customer. This ERP system is easily upgradable so it helps the company to growth faster in fast changing environment by adopting new technologies and by e-commerce to reach to many customers.

## References

1. Cook Book of OfBiz by Ruth Hoffman
2. Apache OfBiz Development -The Beginner's Tutorial by Jonathon Wong & Rupert Howell
3. OfBiz in a Nutshell by Si Chen Open Source Strategies, Inc.
4. <http://ofbiz.apache.org/>
5. <http://www.hotwaxmedia.com/ofbiz-features.html>
6. <http://oss.org.cn/ossdocs/applications/ofbiz/ofbiz-2.1.1-docs/website/presentation/img5.html>
7. <http://www.opensourcestrategies.com/ofbiz/tutorials.php>
6. <http://www.hotwaxmedia.com/apache-ofbiz-blog/tag/ofbiz-catalog-manager/>
8. The Data Model Book By Silverstone