BPEL & ITS IMPLEMENTATION IN GARMENT SECTOR

Rajib Saha
Student ID: 02201110

Department of Computer Science and Engineering
January 2007

BRAC University, Dhaka, Bangladesh

Supervisor

Mrs. Sadia Hamid Kazi
Lecturer, Department Of Computer Science and Engineering
BRAC University[URL – www.bracuniversity.net]
66-Mohakhali, Dhaka – 1212, Bangladesh
Email – skazi@bracuniversity.ac.bd

Abstract

This document defines a notation for specifying business process behavior based on Web Services. Web services are touted to turn the internet into a general platform for distributed computing as required for B2B eCommerce. Web services are the glue that will link together web deployed components to form web applications. Just as there are issues with traditional component adaptation due to versioning and independent development so there are with web services. Many organizations are moving from an object-oriented paradigm for managing business processes toward a service-oriented approach; indeed, services are becoming the fundamental elements of application development. At the same time, Business Process Execution Language (BPEL) has become the standard for orchestrating these services and managing flawless execution of business process. The confluence of these trends is presenting some interesting opportunities for more flexible, cost-effective management of business processes. For our local garment sector we are developing a BPEL model so that it enhances process of the decision making of buyer.

Table of Contents

Keyword: Web Service, SOAP, WSDL, UDDI, XML, XLANG, WSFL, Orchestration, Choreography

1.0 Introduction

The goal of the Web Services effort is to achieve universal interoperability between applications by using Web standards. Web Services use a loosely coupled integration model to allow flexible integration of heterogeneous systems in a variety of domains including business-to-consumer, business-to-business and enterprise application integration. The following basic specifications originally defined the Web Services space: SOAP, Web Services Description Language (WSDL) and Universal Description, Discovery, and Integration (UDDI). SOAP defines an XML messaging protocol for basic service interoperability. WSDL introduces a common grammar for describing services. UDDI provides the infrastructure required to publish and discover services in a systematic way. Together, these specifications allow applications to find each other and interact following a loosely coupled, platform independent model.

Systems integration requires more than the ability to conduct simple interactions by using standard protocols. The full potential of Web Services as an integration platform will be achieved only when applications and business processes are able to integrate their complex interactions by using a standard process integration model. The interaction model that is directly supported by WSDL is essentially a stateless model of synchronous or uncorrelated asynchronous interactions. Models for business interactions typically assume sequences of peer-to-peer message exchanges, both synchronous and asynchronous long-running interactions involving two or more parties. To define such business interactions, a formal description of the message exchange protocols used by business processes in their interactions is needed. The definition of such *business protocols* involves precisely specifying the mutually visible message exchange behavior of each of the parties involved in the protocol, without revealing their internal implementation. There are two good reasons to separate the public aspects of business process behavior from internal or private aspects. One is that businesses obviously do not want to reveal all their internal decision making and data management to their business partners. The other is that, even where this is not the case, separating public from private process provides the freedom to change private aspects of the process implementation without affecting the public business protocol.

Business protocols must clearly be described in a platform-independent manner and must capture all behavioral aspects that have cross-enterprise business significance. Each participant can then understand and plan for conformance to the business protocol without engaging in the process of human agreement that adds so much to the difficulty of establishing cross-enterprise automated business processes today.

What are the concepts required to describe business protocols? And what is the relationship of these concepts to those required to describe executable processes? To answer these questions, consider the following:

- Business protocols invariably include data-dependent behavior. For example, a supply-chain protocol depends on data such as the number of line items in an

order, the total value of an order, or a deliver-by deadline. Defining business intent in these cases requires the use of conditional and time-out constructs.

• The ability to specify exceptional conditions and their consequences, including recovery sequences, is at least as important for business protocols as the ability to define the behavior in the "all goes well" case.

• Long-running interactions include multiple, often nested units of work, each with its own data requirements. Business protocols frequently require cross-partner coordination of the outcome (success or failure) of units of work at various levels of granularity.

If we wish to provide precise predictable descriptions of service behavior for cross-enterprise business protocols, we need a rich process description notation with many features reminiscent of an executable language. The key distinction between public message exchange protocols and executable internal processes is that internal processes handle data in rich private ways that need not be described in public protocols.

In thinking about the data handling aspects of business protocols it is instructive to consider the analogy with network communication protocols. Network protocols define the shape and content of the protocol envelopes that flow on the wire, and the protocol behavior they describe is driven solely by the data in these envelopes. In other words, there is a clear physical separation between protocol-relevant data and "payload" data. The separation is far less clear cut in business protocols because the protocol-relevant data tends to be embedded in other application data.

BPEL4WS uses a notion of message properties to identify protocol-relevant data embedded in messages. Properties can be viewed as "transparent" data relevant to public aspects as opposed to the "opaque" data that internal/private functions use. Transparent data affects the public business protocol in a direct way, whereas opaque data is significant primarily to back-end systems and affects the business protocol only by creating non-determinism because the way it affects decisions is opaque. We take it as a principle that any data that is used to affect the behavior of a business protocol must be transparent and hence viewed as a property.

The implicit effect of opaque data manifests itself through non-determinism in the behavior of services involved in business protocols. Consider the example of a purchasing protocol. The seller has a service that receives a purchase order and responds with either acceptance or rejection based on a number of criteria, including availability of the goods and the credit of the buyer. Obviously, the decision processes are opaque, but the fact of the decision must be reflected as behavior alternatives in the external business protocol. In other words, the protocol requires something like a switch activity in the behavior of the seller's service but the selection of the branch taken is nondeterministic. Such non-determinism can be modeled by allowing the assignment of a nondeterministic or opaque value to a message property, typically from an enumerated set of possibilities. The property can then be used in defining conditional behavior that captures behavioral alternatives without revealing actual decision processes. BPEL4WS explicitly allows the use of non-deterministic data values to make it possible to capture the essence of public behavior while hiding private aspects.

The basic concepts of BPEL4WS can be applied in one of two ways. In *abstract process* allow specification of the public message exchange between parties only. They do not

include the internal details of process flows and are not executable. It is also possible to use BPEL4WS to define an executable business process which allow you to specify the exact details of business processes.
Even where private implementation aspects use platform-dependent functionality, which is likely in many if not most realistic cases, the continuity of the basic conceptual model between abstract and executable processes in BPEL4WS makes it possible to export and import the public aspects embodied in business protocols as process or role templates while maintaining the intent and structure of the protocols. This is arguably the most attractive prospect for the use of BPEL4WS from the viewpoint of unlocking the potential of Web Services because it allows the development of tools and other technologies that greatly increase the level of automation and thereby lower the cost in establishing cross-enterprise automated business processes.
In summary, we believe that the two usage patterns of business protocol description and executable business process description require a common core of process description concepts. In this specification we clearly separate the core concepts from the extensions required specifically for the two usage patterns. The BPEL4WS specification is focused on defining the common core, and adds only the essential extensions required for each usage pattern.
BPEL4WS defines a model and a grammar for describing the behavior of a business process based on interactions between the process and its partners. The interaction with each partner occurs through Web Service interfaces, and the structure of the relationship at the interface level is encapsulated in what we call a *partner link*. The BPEL4WS process defines how multiple service interactions with these partners are coordinated to achieve a business goal, as well as the state and the logic necessary for this coordination. BPEL4WS also introduces systematic mechanisms for dealing with business exceptions and processing faults. Finally, BPEL4WS introduces a mechanism to define how individual or composite activities within a process are to be compensated in cases where exceptions occur or a partner requests reversal.

2.0 What is BPEL

Business Process Execution Language for Web Services (BPEL or BPEL4WS) is a language used for the definition and execution of business processes using Web services. BPEL enables the top-down realization of Service Oriented Architecture (SOA) through composition, orchestration, and coordination of Web services. BPEL provides a relatively easy and straightforward way to compose several Web services into new composite services called *business processes*.

3.0 Background

First, some background. BPEL builds on the foundation of XML and Web services; it uses an XML-based language that supports the Web services technology stack, including SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination, and WS-Transaction.
BPEL represents a convergence of two early workflow languages; Web Services Flow Language (WSFL) and XLANG. WSFL was designed by IBM and is based on the

concept of directed graphs. XLANG, a block-structured language, was designed by Microsoft. BPEL combines both approaches and provides a rich vocabulary for description of business processes.

The first version of BPEL was developed in August 2002. Since then, many major vendors have joined (including Oracle), resulting in several modifications and improvements, and the adoption of version 1.1 in March 2003. In April 2003, BPEL was submitted to the Organization for the Advancement of Structured Information Standards (OASIS) for standardization purposes, and the Web Services Business Process Execution Language Technical Committee (WSBPEL TC) was formed. This effort has led to even broader acceptance in industry.

Within the enterprise, BPEL is used to standardize enterprise application integration as well as to extend the integration to previously isolated systems. Between enterprises, BPEL enables easier and more effective integration with business partners. BPEL stimulates enterprises to further define their business processes, which in turn leads to business process optimization, reengineering, and the selection of the most appropriate processes, thus further optimizing the organization. Definitions of business processes described in BPEL do not affect existing systems, thereby stimulating upgrades. BPEL is the key technology in environments where functionalities are already or will be exposed via Web services. With increases in the use of Web services, the importance of BPEL will increase as well.

4.0 Standard Building Block of BPEL

BPEL4WS is layered on top of several XML specifications: WSDL 1.1, XML Schema 1.0, and XPath1.0. WSDL messages and XML Schema type definitions provide the data model used by BPEL4WS processes. XPath provides support for data manipulation. All external resources and partners are represented as WSDL services. BPEL4WS provides extensibility to accommodate future versions of these standards, specifically the XPath and related standards used in XML computation. [See Figure 1]

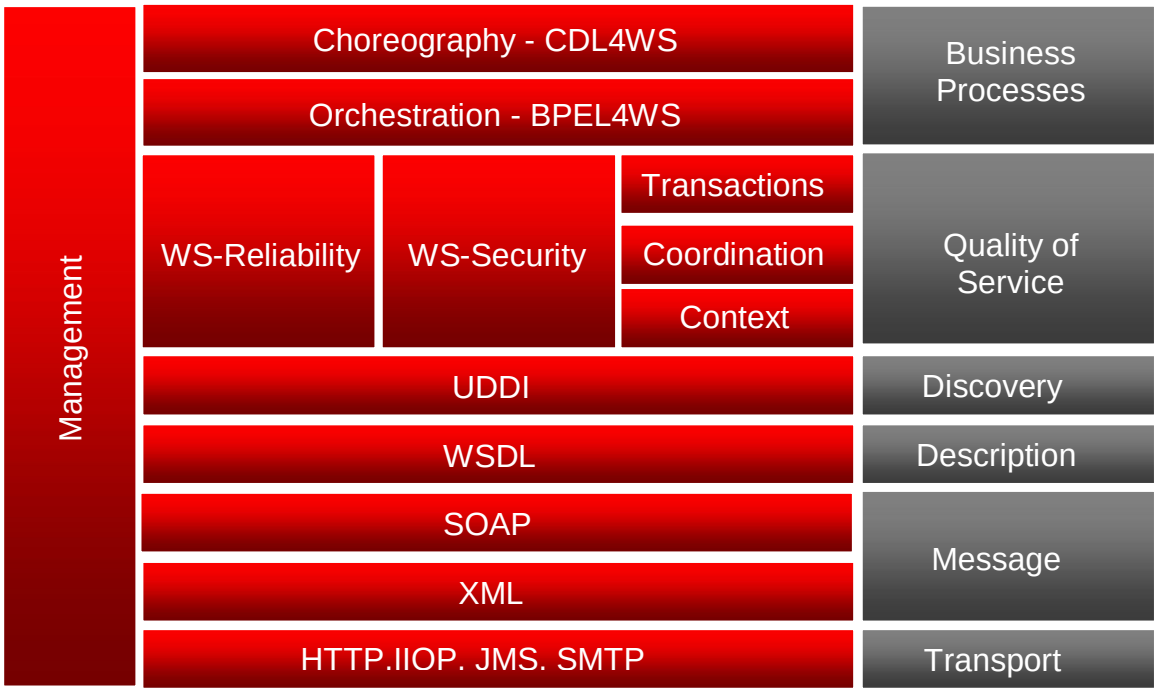

Figure – 1 [Photo courtesy – Oracle Group]

5.0 Web Service Model

Before move to Web Service Model, the term *Web services* is used very often nowadays, although not always with the same meaning. Nevertheless, the underlying concepts and technologies are to a large extent independent of how they may be interpreted. Existing definitions range from the very generic and all-inclusive to the very specific and restrictive. Often, a Web service is seen as an application accessible to other applications over the Web. This is a very open definition, under which just about anything that has a URL is a Web service. It can, for instance, include a CGI script. It can also refer to a program accessible over the Web with a stable API, published with additional descriptive information on some service directory. A more precise definition is provided by the UDDI consortium, which characterizes Web services as *"self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces"*. This definition is more detailed, placing the emphasis on the need for being compliant with Internet standards. In addition, it requires the service to be open, which essentially means that it has a published interface that can be invoked across the Internet. In spite of this clarification, the definition is still not precise enough. For instance, it is not clear what it is meant by a modular, self-contained business application. A step further in refining the definition of Web services is the one provided by the World Wide Web consortium (W3C), and specifically the group involved in the Web Service Activity: *"a software application identified by a URI, whose interfaces and bindings are capable of being defined, described and discovered as XML artifacts. A Web service supports direct interactions with other software agents using XML-based messages exchanged via Internet-based protocols"*. The W3C definition is quite accurate and also hints at how Web services should work. The definition stresses that Web services should be capable of

being “defined, described, and discovered,” thereby clarifying the meaning of “accessible” and making more concrete the notion of “Internet-oriented, standards-based interfaces.” It also states that Web services should be “services” similar to those in conventional middleware. Not only they should be “up and running,” but they should be described and advertised so that it is possible to write clients that bind and interact with them. In other words, Web services are components that can be integrated into more complex distributed applications. This interpretation is very much in line with the perspective we take in this book, and explains why we place so much emphasis on the need to understand middleware as the first step toward understanding Web services.

The W3C also states that XML is part of the solution. Indeed, XML is so popular and widely used today that, just like HTTP and Web servers, it can be considered as being part of Web technology. There is little doubt that XML will be the data format used for many Web-based interactions. Note that even more specific definitions exist. For example, in the online technical dictionary *Webopedia*, a Web service is defined as *“a standardized way of integrating Web-based applications using the XML, SOAP, WSDL, and UDDI open standards over an Internet protocol backbone. XML is used to tag the data, SOAP is used to transfer the data, WSDL is used for describing the services available, and UDDI is used for listing what services are available”*. Specific standards that could be used for performing binding and for interacting with a Web service are mentioned here. These are the leading standards today in Web services. As a matter of fact, many applications that are “made accessible to other applications” do so through SOAP, WSDL, UDDI, and other Web standards. However, these standards do not constitute the essence of Web services technology: the problems underlying Web services are the same regardless of the standards used. This is why, keeping the above observations in mind, we can adopt the W3C definition and proceed toward detailing what Web services really are and what they imply.

Web services usually expose operations of certain applications or information systems. Consequently, combining several Web services actually involves the integration of the underlying applications and their functionalities. Web services can be combined in two ways:

- Orchestration
- Choreography

5.1.1 Orchestration

In orchestration, which is usually used in private business processes, a central process (which can be another Web service) takes control of the involved Web services and coordinates the execution of different operations on the Web services involved in the operation. The involved Web services do not "know" (and do not need to know) that they are involved in a composition process and that they are taking part in a higher-level business process. Only the central coordinator of the orchestration is aware of this goal, so the orchestration is centralized with explicit definitions of operations and the order of invocation of Web services. [See Figure 2 in next page]

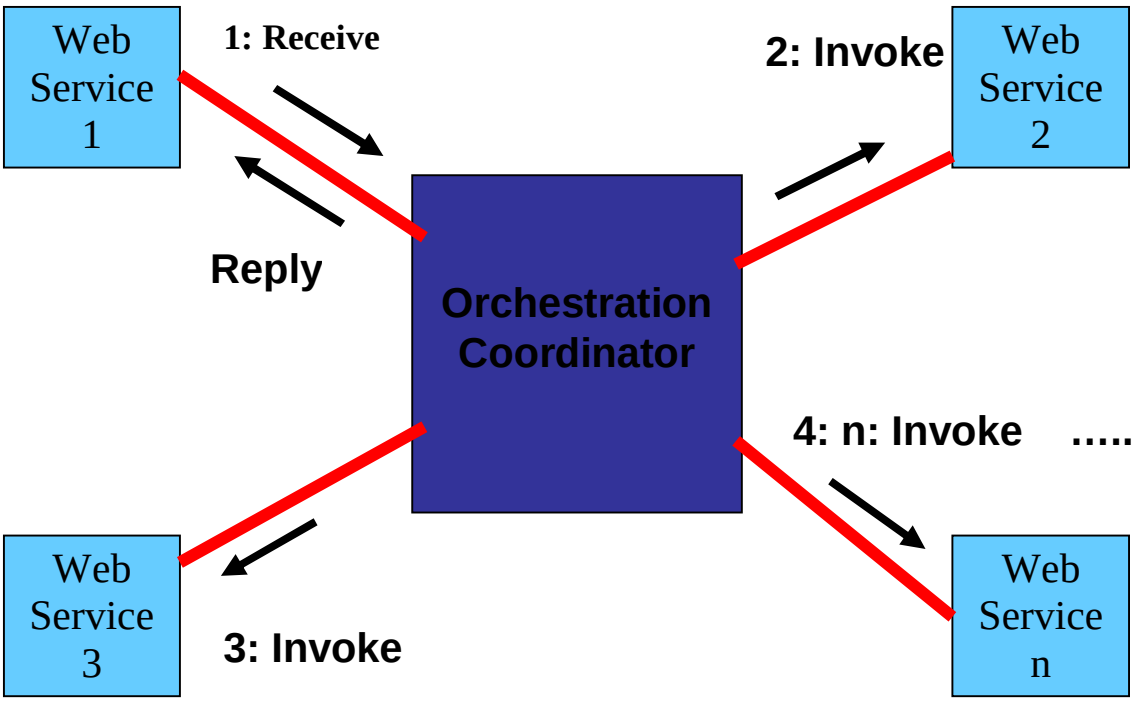

Figure 2

5.1.2 Choreography

Choreography, in contrast, does not rely on a central coordinator. Rather, each Web service involved in the choreography knows exactly when to execute its operations and with whom to interact. Choreography is a collaborative effort focusing on the exchange of messages in public business processes. All participants in the choreography need to be aware of the business process, operations to execute, messages to exchange, and the timing of message exchanges. [See Figure 3 in next page]

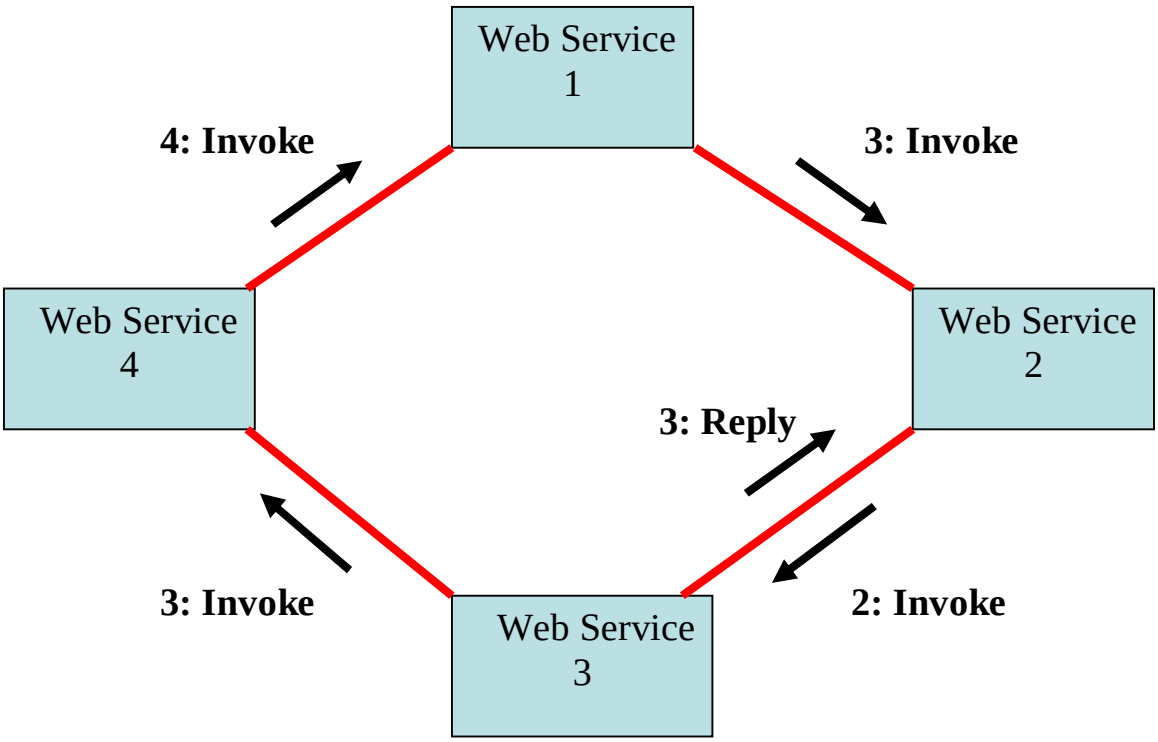

Figure 3

From the perspective of composing Web services to execute business processes, orchestration is a more flexible paradigm and has the following advantages over choreography:

►The coordination of component processes is centrally managed by a known coordinator.

►Web services can be incorporated without their being aware that they are taking part in a larger business process.

►Alternative scenarios can be put in place in case faults occur.

5.2 Orchestration or Choreography

BPEL supports two different ways of describing business processes that support orchestration and choreography:

● **Executable Processes:** The logic and state of the process determine the nature and sequence of the Web Service interactions conducted at each business partner, and thus the interaction protocols. While a BPEL4WS process definition is not required to be complete from a private implementation point of view, the language effectively defines a portable execution format for business processes that rely exclusively on Web Service resources and XML data. Moreover, such processes execute and interact with their partners in a consistent way regardless of the supporting platform or programming model

used by the implementation of the hosting environment. They follow the orchestration paradigm and can be executed by an orchestration engine.

● **Abstract Business Protocols:** A BPEL4WS process can define a business protocol role, using the notion of *abstract process*. For example, in a supply-chain protocol, the buyer and the seller are two distinct roles, each with its own abstract process. Their relationship is typically modeled as a partner link. Abstract processes use all the concepts of BPEL4WS but approach data handling in a way that reflects the level of abstraction required to describe public aspects of the business protocol. Specifically, abstract processes handle only protocol-relevant data. BPEL4WS provides a way to identify protocol-relevant data as *message properties*. In addition, abstract processes use non-deterministic data values to hide private aspects of behavior. They follow the choreography paradigm.

6.0 Business Process

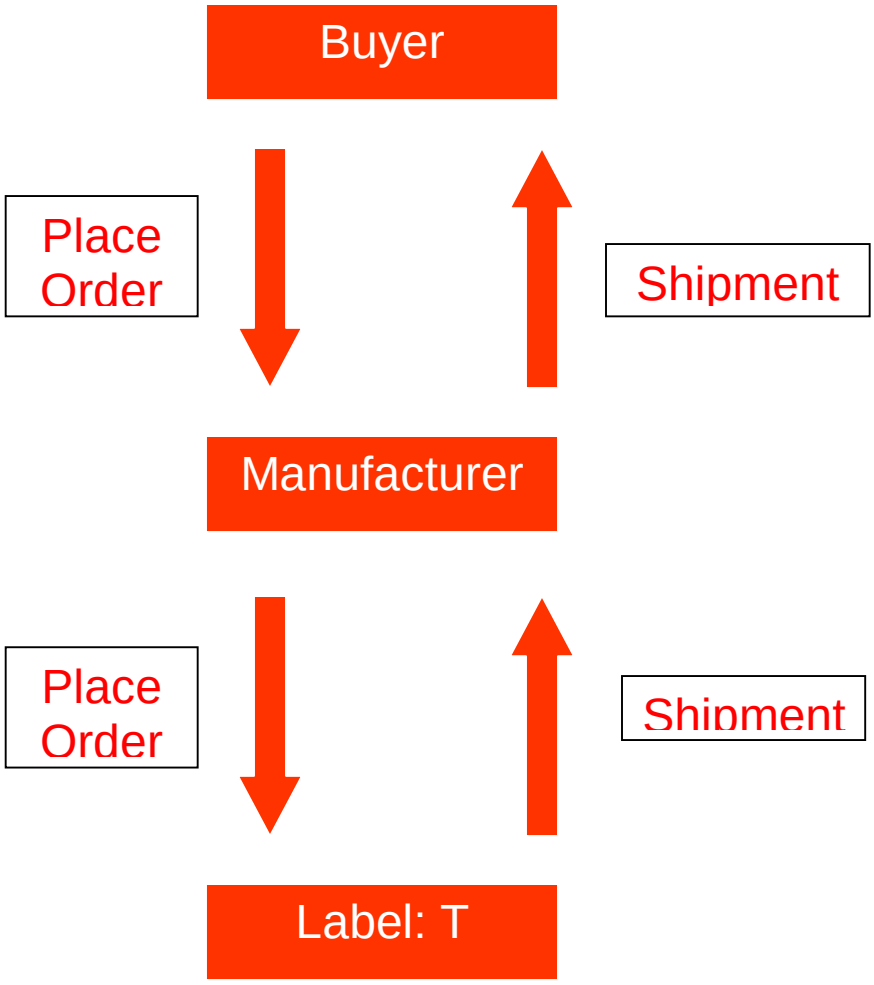

Figure 4

Above picture [Figure 4] is a typical business process. We can call it a segment or portion of a business chain. In a business chain, more than one manufacturer may be involved. In many cases buyer becomes a supplier and this way going until it reaches to the ultimate customer. As this figure is related to our thesis paper so it's really necessary to understand how a buyer becomes a supplier. Suppose AA gives an order to BB and only BB can give what AA wants. At this point AA is buyer and BB is supplier. But to make shape what AA wants BB needs to depend on CC. Now BB becomes buyer and CC becomes supplier. This process will be going on until the product is finished and ready for consumer.

7.0 Design

Let's talk about the scenario. In Bangladesh there are more than six thousand and five hundred garments. Let say, a buyer wants to place an order to the local garment manufacturer → that put 'X' brand button on 1000 pieces of shirt.

7.1.1 Old System

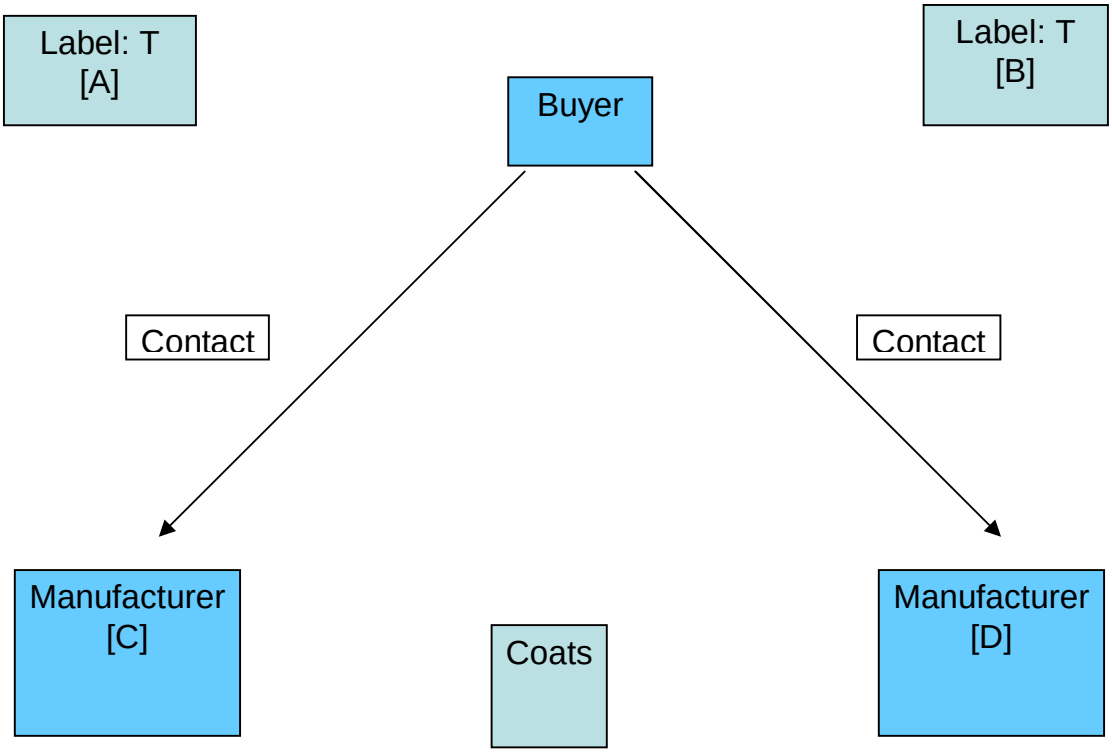

Figure 5

According to old system buyer [let say, XYZ Company] first needs to find out which companies add [put] button on the shirts. Then it needs to find out whether they are capable of taking this order. Because there are many companies who don't have the capacity to put buttons on 1000 pieces of shirt within 10 days. After finding out who are capable it needs to make an agreement with that company about price, delivery date etc.

This is really a difficult task if we have 100 choices in front of us and we have to go thorough 100 records to find-out optimal or best one.

7.1.2 New System

To make it more clear we have divided the total process in some steps. In the scenario we have made field for Buyer, Manufacturer and Brand Button X maker [we use in picture Label]. Here manufacturer and label has multiple blocks just because we are making the system when there are more than one party involved. Again if we can show it for two companies than it will be applicable in n number of companies.

1st Step:

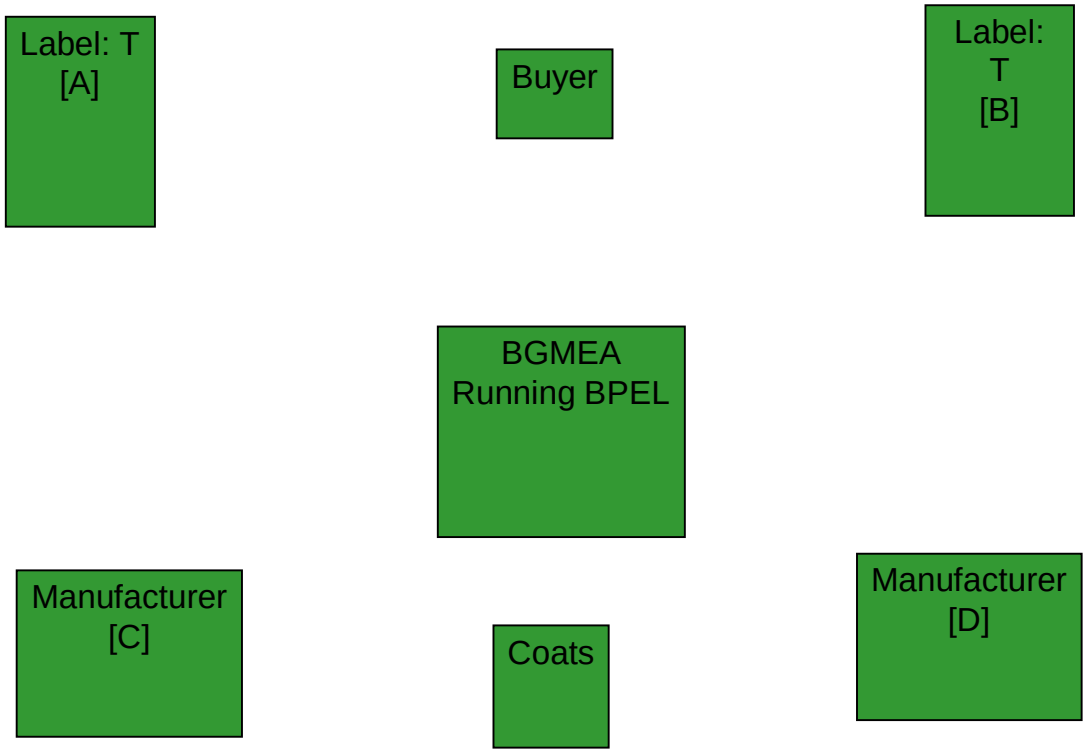

Figure T

This is basically the initial step. Here we have just put the variables who are directly involved in our design. The software we are designing that will run on BGMEA local server. From there every buyer will access it to find out which one is optimal in case of making a business contract. Now we have put a variable called ‘Coat’ in our environment. This is because actually a company except like coat make contract with other company when they have an order. But coat or something like this company make a long term contract with a supplier i.e. five years or more. This type of companies will also be benefited from our system.

2nd Step:

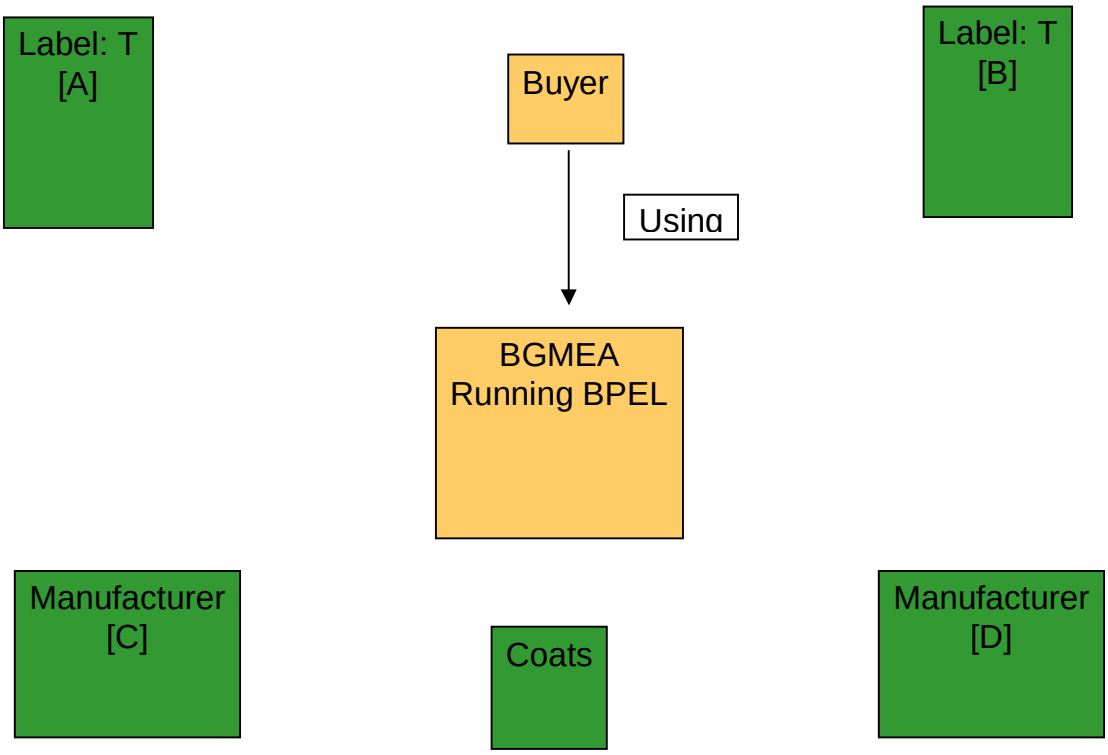

Figure 6

This is the first step where buyer will make a request through BPEL. Go back to the example. XYZ Company will use the BPEL to find out a company which will put button on their given number of shirts.

3rd Step:

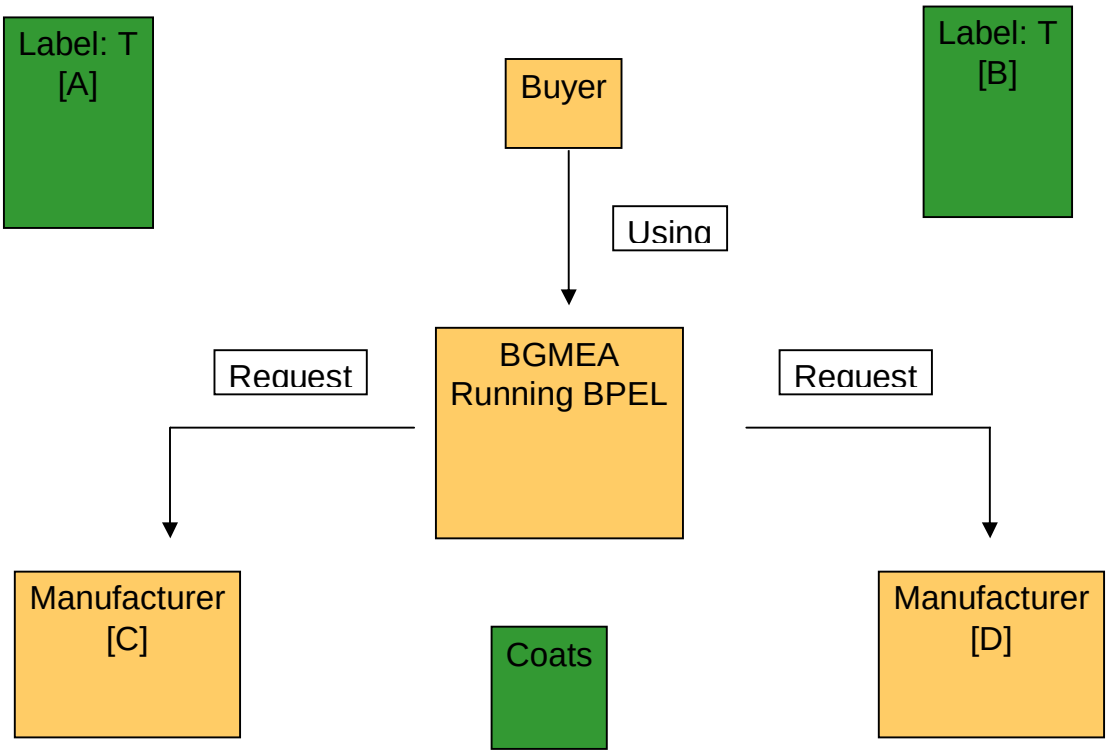

Figure 7

In this step, a request by the system will go to those companies who put button on the shirt. Actually the system is asking for their current status.

4th Step:

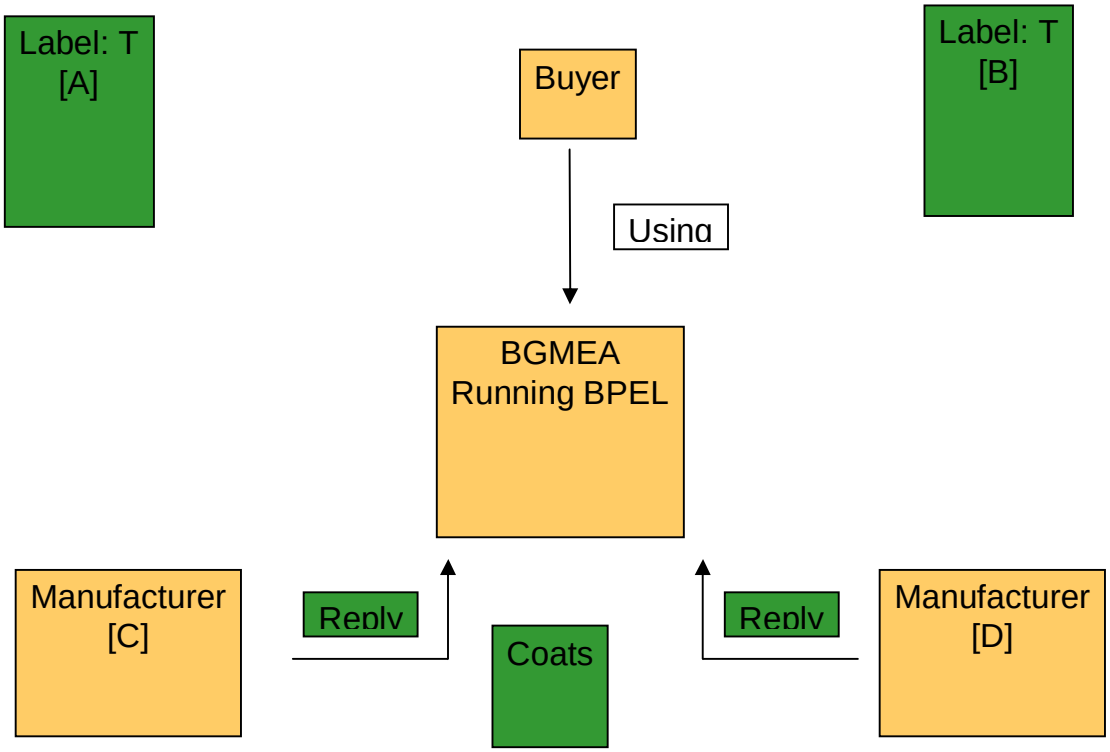

Figure 8

In this step, those companies will reply with their current status.

5th Step:

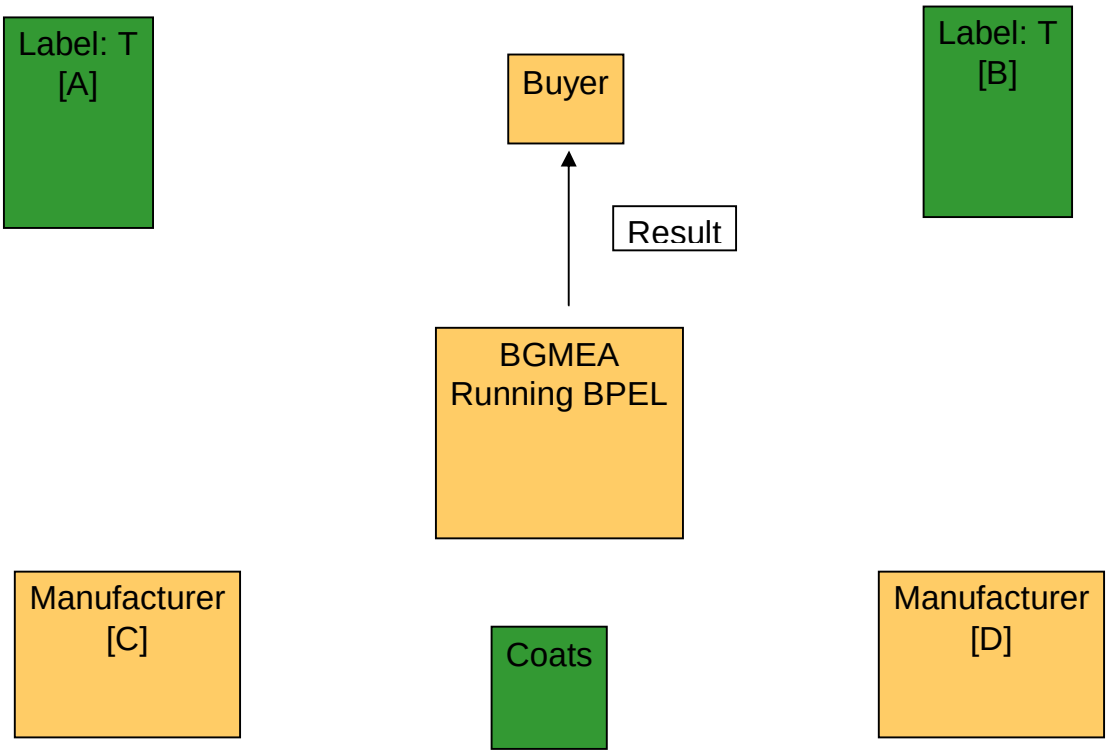

Figure 9

Now the system will analyze all the information it got from those companies as a reply. After analyzing all the information, system will give a result to the buyer that this one is optimal for you.

6th Step:

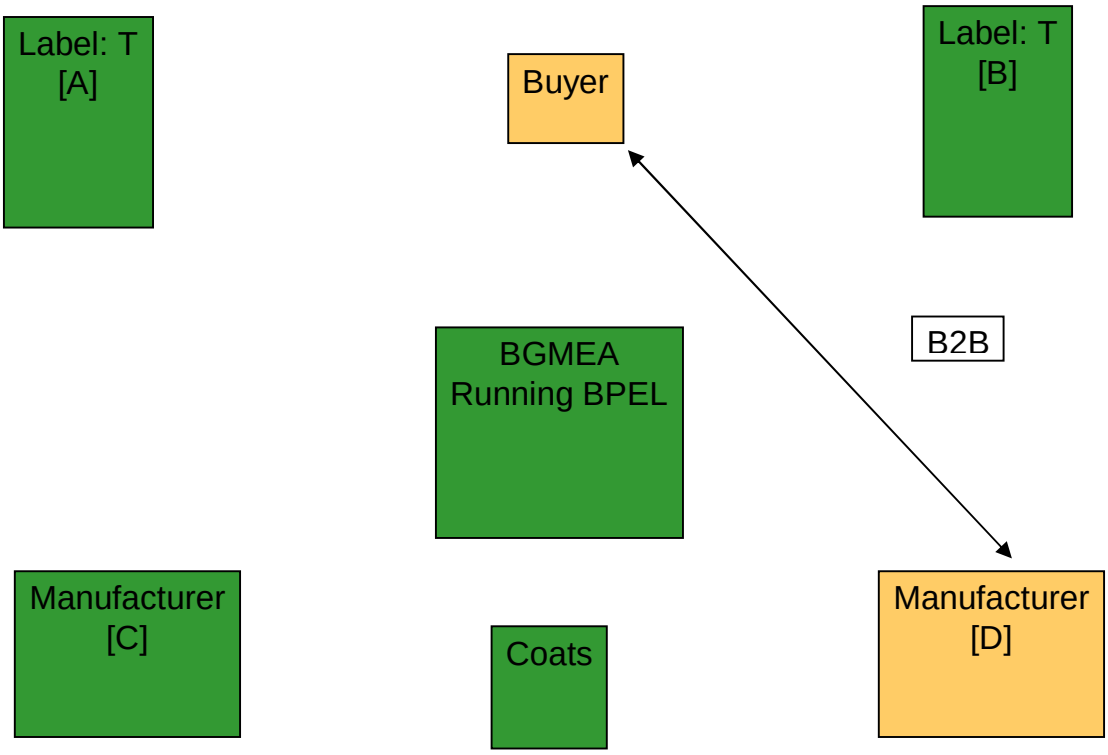

Figure 10

As the system has replied that Company D is optimal for that buyer in terms of price and supply deadline it will implement a B2B with that company. One think make sure that it up to the company whether it will implement B2B with the resulted company. Up to this XYZ Company has found out with which it will make contract.

7th Step:

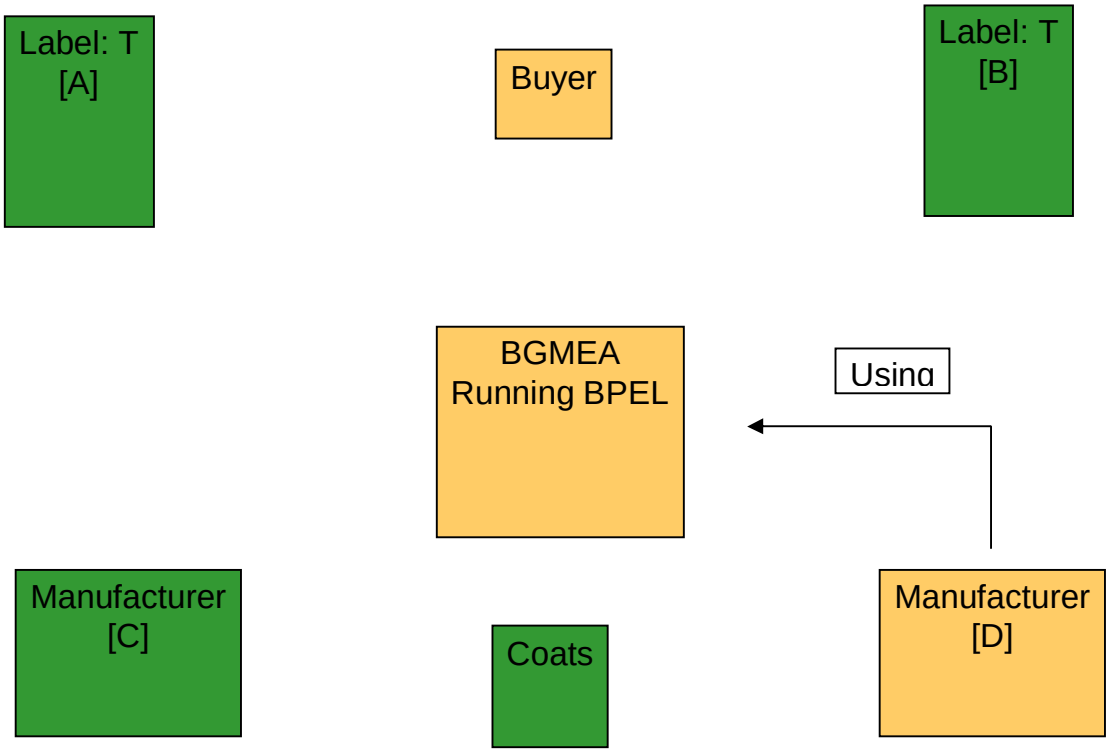

Figure 11

Now company D will find out to which it will put the order. In this scenario we have variable A and B which are X brand button makers. D will use the BPEL.

8th Step:

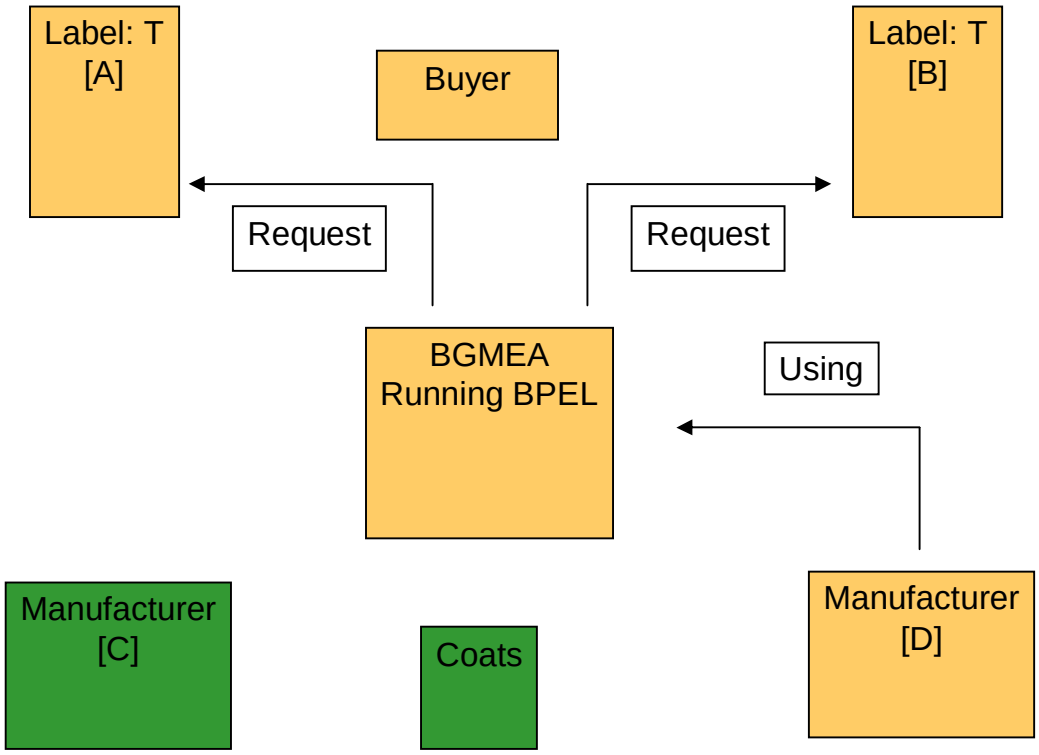

Figure 12

In this step on request of D the system make request to those companies who make X brand buttons.

9th Step:

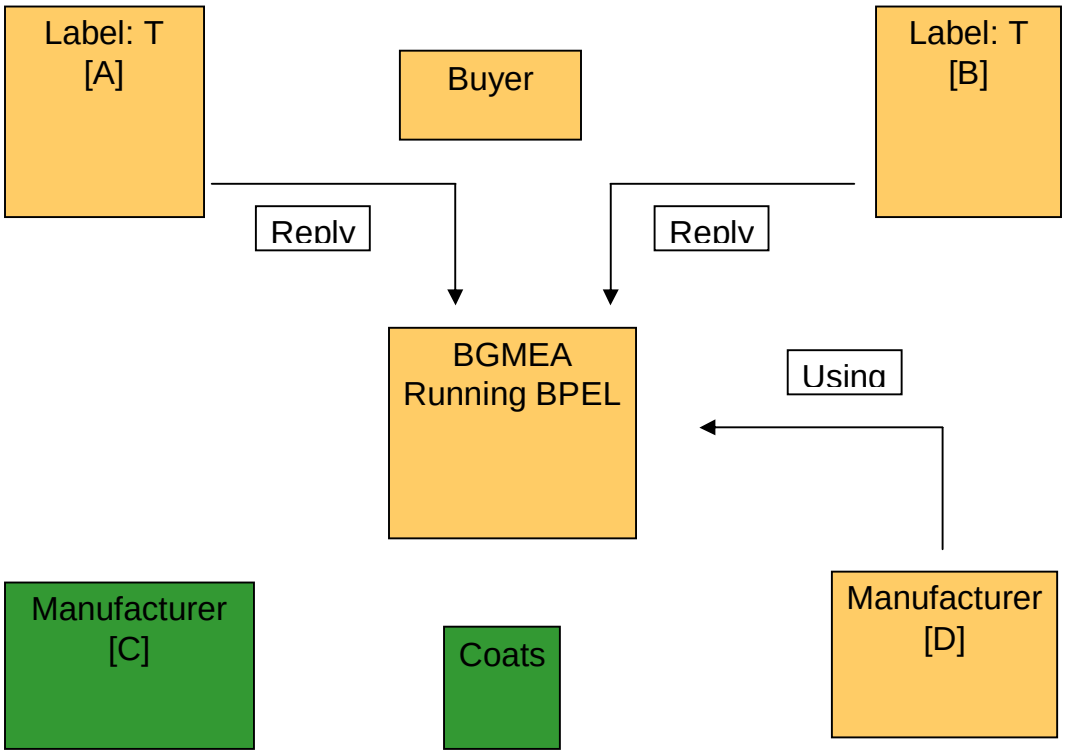

Figure 13

In this step, same way those brand button makers will reply.

10th Step:

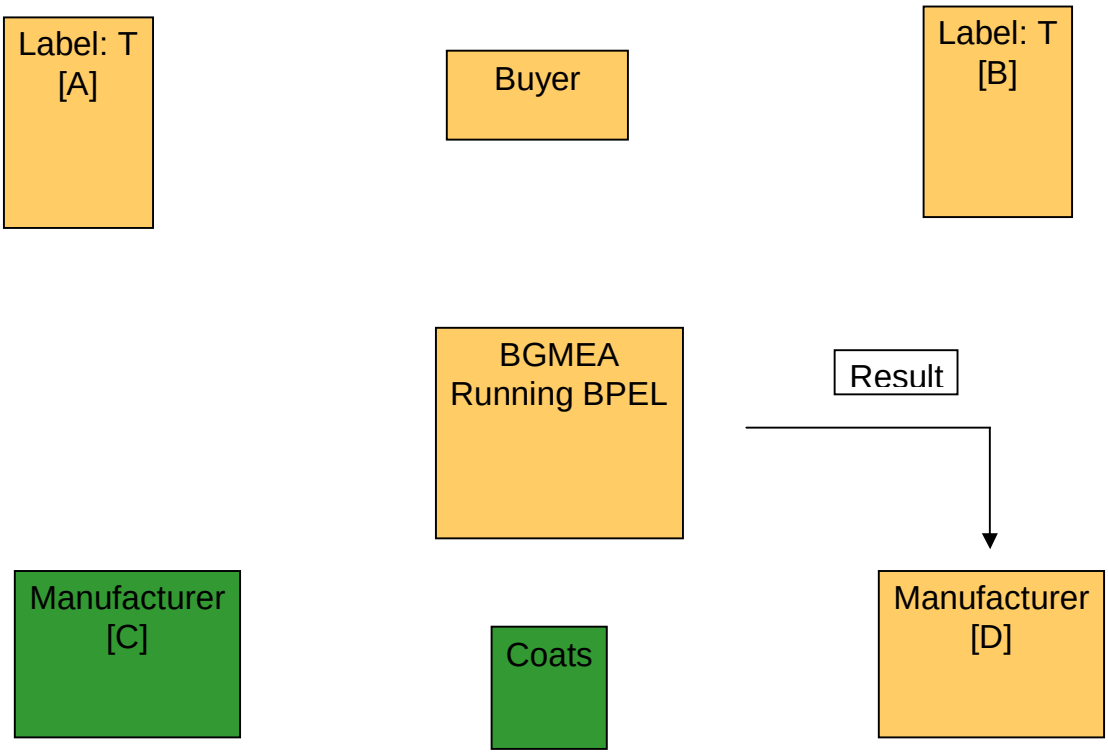

Figure 14

In this step, BPEL will display the result of its query to those companies.

11th Step:

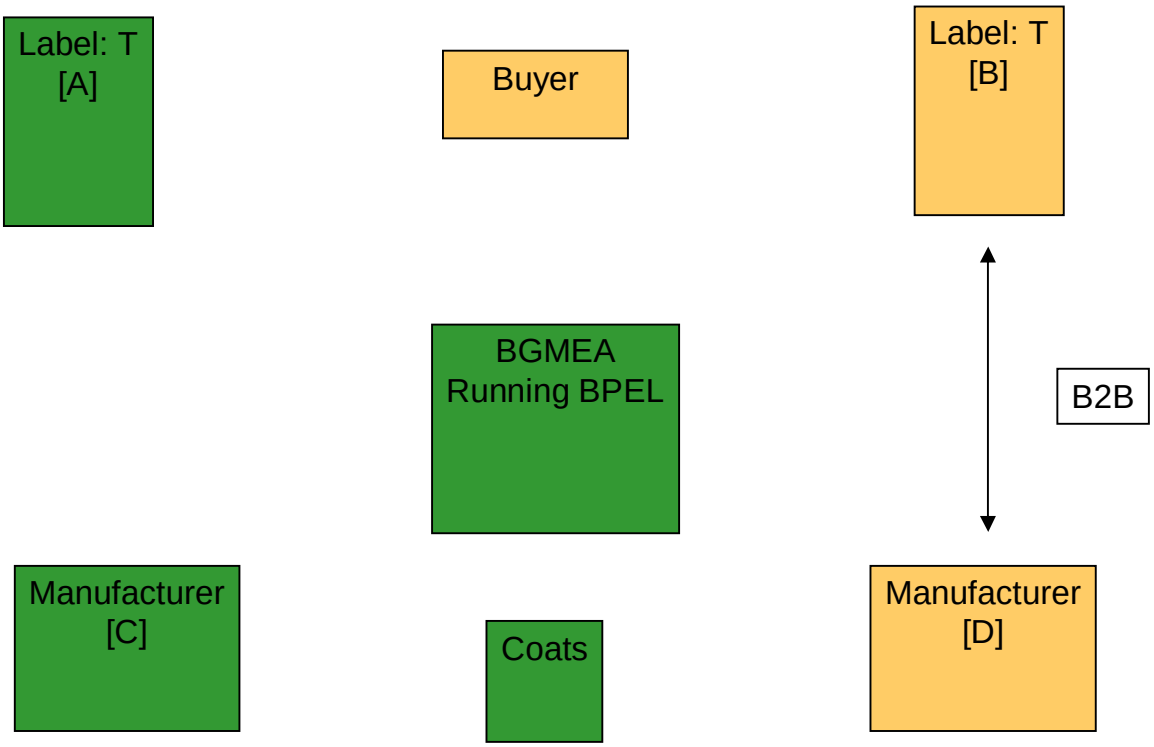

Figure 15

As the company D found B will be the optimal for this company it will implement B2B with it.

12th Step:

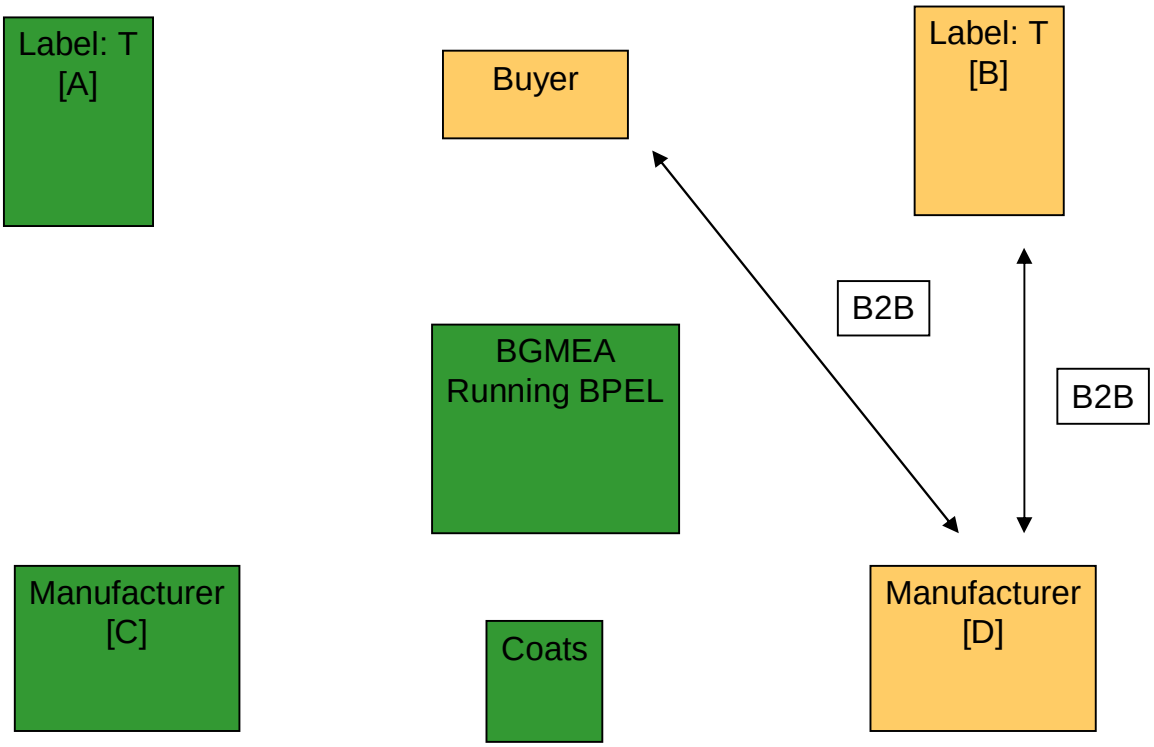

Figure 16

This picture is the ultimate business process between parties. With a little bit of information, provided by the buyer, the system doing job which reduce the pressure from each of the company when it is in buyer position and it needs a suitable supplier.

7.1.3 Typical Database

Reg_Num	Name	Type	Link	Per_Cap	Database

Figure 17

For our thesis, as the scenario demands we have made a simple database design. This database will run under BGMEA. In the database design, we put the Registration Number [in short Reg_Num] as a primary key. Because two companies don't have the same registration number. Name field is for keeping the name of the company. Type field, to identify what type of company it is – is it button or knitting. Link is for accessing the company website. Per_Cap [fully, Per day Capacity] is for keeping track which company produces how much in a day. There is one reason for keeping this information in BGMEA server rather than keeping it to all companies local server. Local company may

manipulate the information to attract buyer i.e. they show a wrong per day capacity and a buyer found oh! X produces 200 units a day where as Y produces 100 units. But the real fact is Y gives the right information where as X gives a wrong one to attract buyer, actually X produces 90 units per day. This will cause a serious loss for a buyer again the new system will generate a wrong result. So from our point of view we think this will be in BGMEA server and if any company thinks that they have changed their per day capacity to a new one they will make a request to BGMEA to upload their new per day capacity and BGMEA will do that after making necessary test/query to that company. The field, Database is for the location of each company's local database, from where our new system will read/retrieve information.

7.2 Old System vs New System

In case of old system, it is totally,

→ Time consuming: Need to spend more time finding out optimal one.
→ Costly: Need to contact with the companies.

In case of new system, it is totally,

→ Time saving: Only the time needed by the system to process the information.
→ Cost effective: Require some mouse clicks and keyboard input.

8.0 The Way Software Works

8.1.1 User-end

This is the portion where user I mean buyer will be involved.

8.1.1.1 First Phase

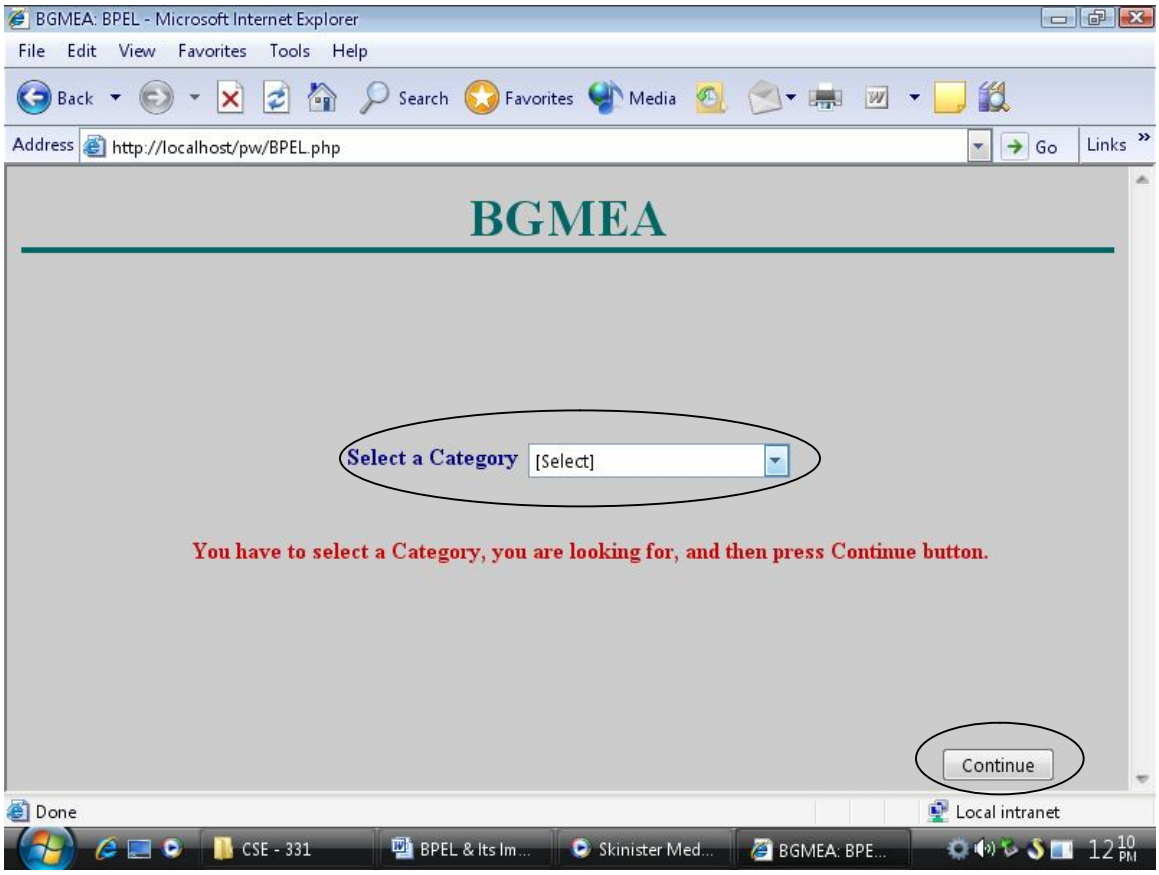

Figure 18

When the software will run at BGMEA's own server, at the beginning user will find the above page. User needs to select a category and then needs to click on "Continue" button. User will move to new page.

8.1.1.2 Second Phase

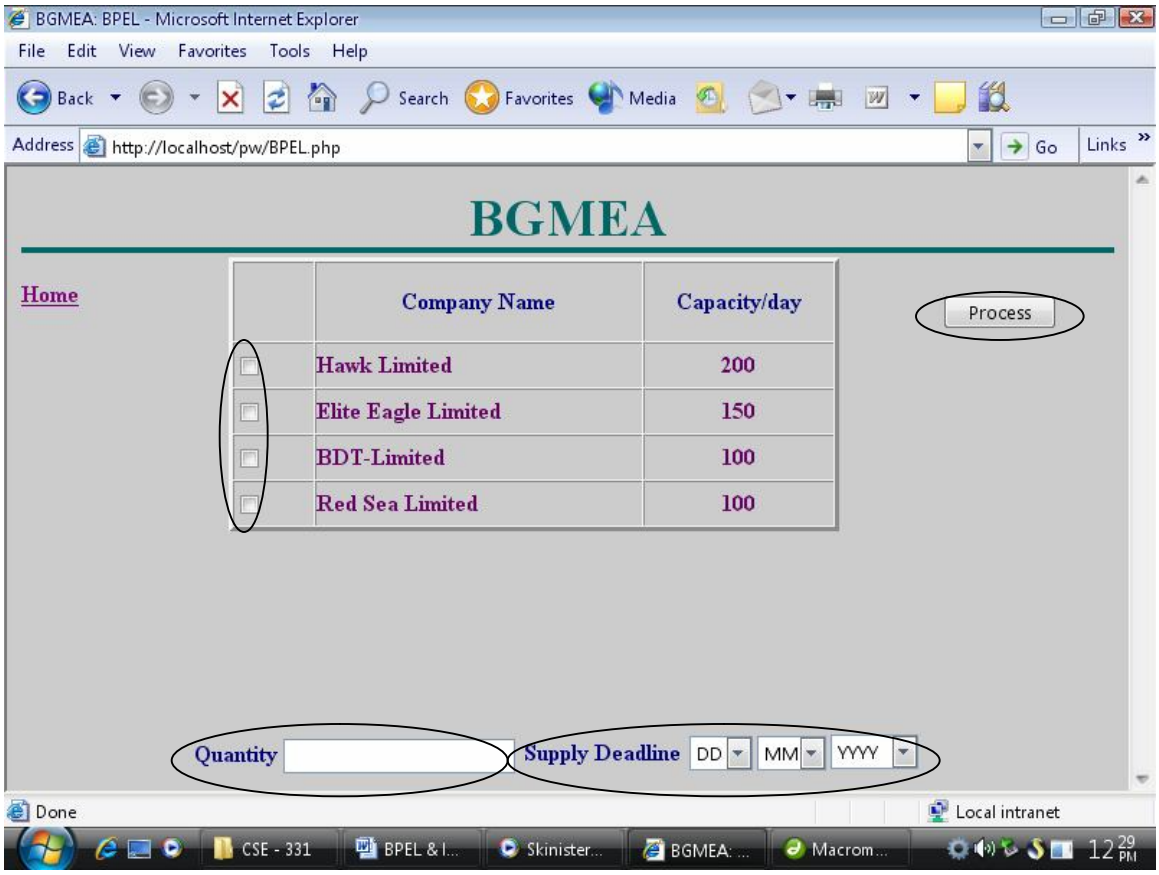

Figure 19

From the last page, if user selects “Button” [in the category field] and click on “Continue” then the above page will be shown. Here, we see company name with its per day capacity. There is a checkbox on the left side of each company name. User needs to select as many as s/he wants to select. Then s/he has to provide “Quantity”, his or her company is going to order. Then need to choose a deadline and then click on “Process” button. User will move to next page.

8.1.1.3 Third Phase

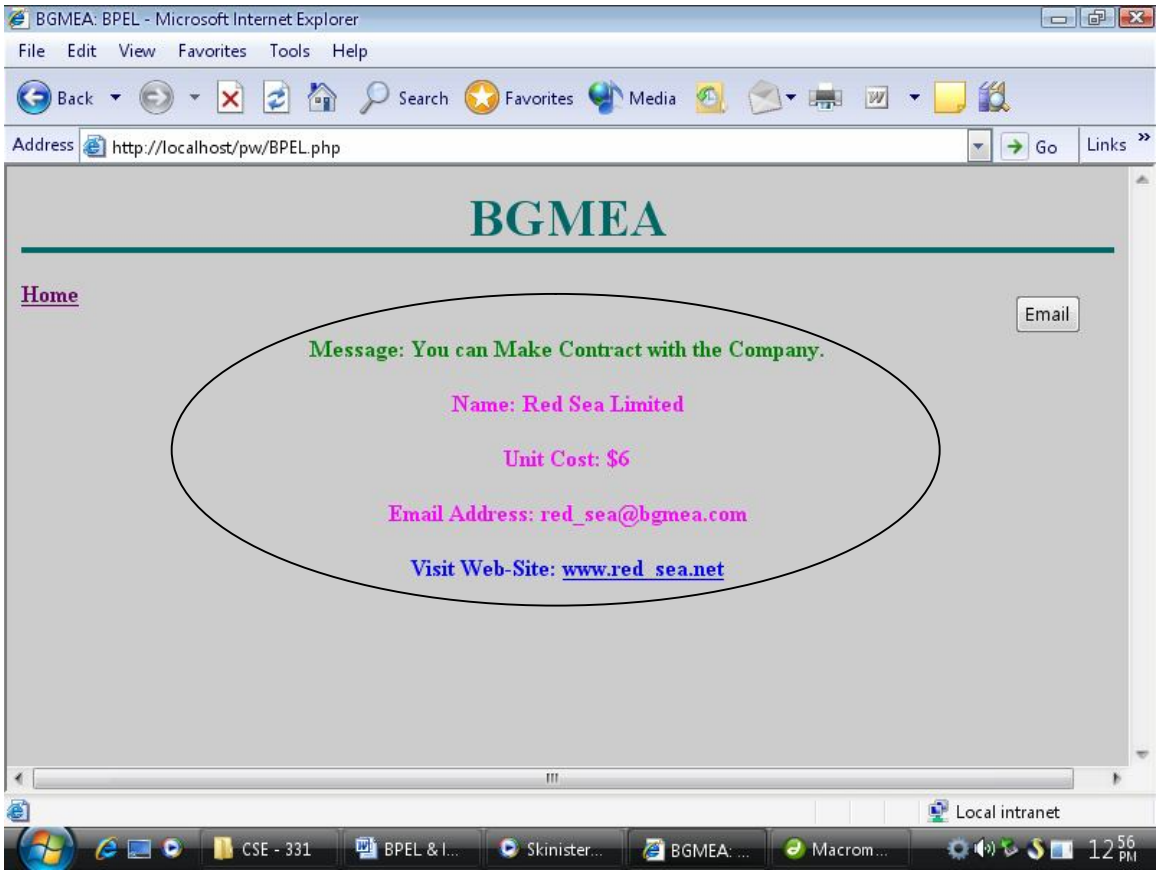

Figure 20

On success, that is, from the number of selected companies or company [if user has interest only on a single company] user will get a message which company can take the order when there are multiple companies involved. For a single company it will just show whether it can take the order or not.
From here, there will be a link to that company or user either chooses the email button to email to that company.

8.1.2 Back-end

The total code is divided into some sub phases which we call function and inside a function there may reside some phases. From our point of view we are explaining how the code works. To make it more clear we will give each phase a name.

Part # 01

```
function index()
{
……….
        <blockquote>
                <p align="left">Select a Category
                        <select name="category" style="position:absolute;
                width: 182px; left: 206px; top: 0px; height: 24px;">
                                <option value="select"
                        selected="selected">[Select]</option>
                                        <?php while($row =
                                mysql_fetch_array($result))
                                                {
                                                echo "<option value =
                        '".$row["Type"]."'>".$row["Type"]."</option>";
                                                }
                                        ?>
                                </select>
                        <input type="submit" style="position:absolute; left: 493px; top:
                211px; width: 79px; height: 24px;" value="Continue"/>
                        </p>
        </blockquote>
……….
}
```

Figure 21

Above code is a function block named index (). It actually tells that user will get the chance to select a category from a number of categories. Then user will click on the continue button to proceed further operation. No selection will result on an error in further processing.

Part # 02

```
Function result()
{
……….
<tr>
    <td width="63" height="54"> </td>
    <td align="center" class="style2" width="267">Company Name</td>
    <td align="center" class="style2" width="142">Capacity/day</td>
</tr>
……….
for($i = 0; $i < 6; $i++)
{
    $temp = $td + $i;
    echo"<option>$temp</option>";
}
……….
}
```

Figure 22

This is a function block called result (). User will get the company with its per day capacity. It will be displayed in descending order incase of per day capacity i.e. highest per day capacity holder will be in the top and lowest will be at the bottom.

Part # 03

```
Function process()
{
……….
foreach ($links as $key => $value)
{
    $result01 = mysql_query("SELECT * FROM category WHERE Link
    = '$value'");
          while($row01 = mysql_fetch_array($result01))
          $db_info[] = $row01["Database"].".xml";
}
……….
// inside it there is another function
……….
}
```

Figure 23

This portion is to format database name with .XML extension so that it becomes available for use. That is when local server will give a reply to the central server [BGMEA Server] it will get data as a XML document.

Part # 04

```
function XmlStartElement($Parser, $Name, $Attrs)
{
……….
if($Name == "REG_NUM")
                    $IsRightTag1 = true;
……….
}
function XmlEndElement($Parser, $Name)
{
……….
if($Name == "REG_NUM")
                    $IsRightTag1 = false;
……….
}
function XmlTextData($Parser, $Data)
{
……….
if($IsRightTag1)
                    $ReturnValue1 = $Data;
……….
}
```

Figure 24

This portion is for parsing the XML data. We know the xml format. For example, in XML we write like this: <message>hello world</message>. Here '<message>' is starting tag and '</message> is ending tag. In between start tag and end tag we have 'hello world' which is data. We need to parse an xml file to retrieve data which will get central server as a reply of local server.

Part # 05

```
……….
foreach ($db_info as $key => $value)
{
    $XmlParser = xml_parser_create();
    xml_set_element_handler($XmlParser, "XmlStartElement",
    "XmlEndElement");
    xml_set_character_data_handler($XmlParser, "XmlTextData");

    $strXML = implode("",file($value));              /* TAKING EACH XML
    FILE */
    xml_parse($XmlParser, $strXML, true);
    xml_parser_free($XmlParser);

    $combine_detail[] =
    $ReturnValue1."|".$ReturnValue2."|".$ReturnValue3."|".$ReturnValu
    e4."|".$ReturnValue5;                    /* FORMATTING DATA */
    $iter++;
}
……….
```

Figure 25

Here basically helping those functions we have seen just before this figure by sending the file name with data. There is loop because we know we have for example 100 companies which makes the same thing.

Part # 06

```
……….
$new_supply_con = mktime(0, 0, 0, $mm, $dd, $yyyy);
$today_con = mktime(0, 0, 0, $sys_mm, $sys_dd, $sys_yyyy);
$buf_diff = ($new_supply_con - $today_con) / ((60*60)*24);
if($iter == 1)
{
……….
list($parse_reg, $parse_ucost, $com_dmy, $parse_fworker,
    $parse_tworker) = split('[|]', $combine_detail[$iter]);
$parse_day = substr($com_dmy, 0, 2);
……….
$parse_day = (int) $parse_day;
……….
$per_worker_cap = $capacity_day / $parse_tworker;
$calc01 = ($per_worker_cap * $parse_tworker) * $buf_diff;
……..
```

Figure 26

Here actually we find out the day supplier will get from the date the order will be given. Now there may happen one thing, a buyer is determined his company will give the order a company named X. Now the buyer needs to know the status of the company that is will it take the order? Again we have to find out per worker capacity so that our job becomes easy.

Part # 07

```
……..
if($calc01 < $quantity)
    // Sorry can't take the order
else
{
    ……..
    if(calc02 > 0)
        // Employed worker become free and join new task
    else
        // Don't get other worker
    if($calc03 < $quantity)
        // Can't complete the order
    else
        // Complete the task
    ……..
}
……..
```

Figure 27

We are making decision depending on some attributes. Within time limit, if company doesn't have the ability to complete the order, the system will show that selected company cannot take the order. Now there may be case when company can complete the task only if all the workers are free. So we need to find out that. If all the condition meet system will show a success message that you can make a contract with the selected company.

Part # 08

```
if($flag_c == -$iter)
    // No company can take the order
else
{
    for($i = 0; $i < $iter; $i++)
    {
        if($flag[$i] == "true")
        {
            $calc03 = ($buf_diff * ($parse_fworker[$i] *
    $per_worker_cap[$i])) + $comp_add_worker[$i];
            if($click > 0)
            {
                If($dec_parse_ucost[$i] > $dec_parse_ucost[$i-1])
                {
                    $temp_reg = $parse_reg[$i-1];
                    $temp_cost = $dec_parse_ucost[$i-1];
                }
                else
                {
                    $temp_reg = $parse_reg[$i];
                    $temp_cost = $dec_parse_ucost[$i];
                }
            }
            $click++;
        }
}
```

Figure 28

This part if there are more than one company involved we need to first sort out which companies are capable in case of taking the order. Then we have to consider whether any of the selected companies is engaged in an order right now. After completing the old order should it be able to complete the new order. Which one will be the optimal in terms of price? Then decision is made.

9.0 Complete Code

Code: Design.CSS

```
/*********************************************************************/

/* CSS Document */
.style1 {
        color:#660066;
        font-weight: bold;
}
```

```
.style2 {
	color: #000080;
	font-weight: bold;
}
.style3 {
	color: #CC0000;
	font-weight:900;
}
.style4 {
	color:#006666;
	font-size:36px;
}

/*************************************************************************/
```

Code: BPEL.PHP

```
/*************************************************************************/

 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>BGMEA: BPEL</title>
<link rel="stylesheet" href="Design.css" type="text/css" />
</head>

<!-- THIS IS FOR HEADER IN EVERY PAGE -->
<h1>
	<p align="center" class="style4" style="border-bottom:solid">BGMEA</p>
</h1>

<body  bgcolor="#CCCCCC">
<?php
			/* FUNCTION FOR START UP / HOME PAGE */
function index()
{
			/* CONNECTING TO MYSQL SEVER */
	$con = mysql_connect("localhost", "root", "microsoft");
	if(!$con) die("FAILED TO CONNECT WITH MYSQL SERVER");
	mysql_select_db("category_database", $con);
	$result = mysql_query("SELECT DISTINCT Type FROM category ORDER BY
Type");
?>
```

```
<!-- DESIGN OF START UP / HOME FORM -->

	<form action="BPEL.php" method="post" name="form1" class="style3"
id="form1" style="position:absolute; height: 21px; top: 257px; left: 129px;">
		<input type="hidden" name="action" value="result" />
			<label>You have to select a Category, you are looking for, and
then press Continue button.</label>
			<div align="center" class="style2" style="position:absolute; left:
27px; top: -65px; width: 298px; height: 23px;">
			<blockquote>
				<blockquote>
				<p align="left">Select a Category
					<select name="category" style="position:absolute;
width: 182px; left: 206px; top: 0px; height: 24px;">
<option value="select"
selected="selected">[Select]</option>
<?php
while($row =
mysql_fetch_array($result))
{
echo "<option
value = '".$row["Type"]."'>".$row["Type"]."</option>";
}
?>
					</select>
<input type="submit"
style="position:absolute; left: 493px; top: 211px; width: 79px; height: 24px;"
value="Continue"/>
				</p>
				</blockquote>
			</blockquote>
			</div>
	</form>
<?php
}

			/* FUNCTION FOR 2ND PAGE / SELECTION ZONE */
function result()
{
			/* CONNECTING TO MYSQL SERVER */
	$con = mysql_connect("localhost", "root", "microsoft");
	if(!$con) die("FAILED TO CONNECT WITH MYSQL SERVER");
	mysql_select_db("category_database", $con);
	$result = mysql_query("SELECT * FROM category WHERE Type =
'".$_POST['category']."' ORDER BY Per_Cap DESC");
```

```
?>
                        <!-- DESIGN OF 2ND FORM / SELECTION ZONE -->

        <form action="BPEL.php" method="post" name="form1" class="style3"
id="form1">
                <input type="hidden" name="action" value="process" />
                <a href="BPEL.php" target="_parent">Home</a>

                        <!-- DESIGNING TABLE -->
                <table style="position:absolute; left: 154px; top: 63px; width: 424px;
height: 124px;" width="93%" height="84" border="3" cellpadding="0" cellspacing="0">
                        <tr>
                        <td width="63" height="54"> </td>
                        <td align="center" class="style2" width="267">Company
Name</td>
                        <td align="center" class="style2" width="142">Capacity/day</td>
                        </tr>
<?php
        $counter = 0;

        while($row = mysql_fetch_array($result))
        {
?>
                        <!-- DISPLAYING COMPANY NAME WITH PER DAY
CAPACITY -->
                <tr>
                        <td class="style1" width="63" height="30"><input
type="checkbox" name="links[<?php echo $counter;?>]" value="<?php echo
$row["Link"] ?>" />     </td>
                        <td width="267" height="30"><span class="style1"><?php echo
$row["Name"] ?></span></td>
                        <td align="center" width="142" height="30"><span
class="style1"><?php echo $row["Per_Cap"] ?></span></td>
                </tr>
<?php
                $counter++;
        }
?>
                </table>

                        <!-- INPUT PORTION FOR QUANTITY AND SUPPLY
DEADLINE -->
                <input name="Submit" type="submit" style="position:absolute; left:
650px; top: 89px; width: 79px; height: 24px;" value="Process"
onclick="windows.location.href='BPEL.php?action=process'"/>
```

```
		<p class="style2" style="position:absolute; left: 130px; top:
398px;">Quantity</p>
		<label><input style="position:absolute; left: 192px; top: 397px;"
type="text" name="quantity"/></label>
		<p class="style2" style="position:absolute; left: 351px; top: 397px; width:
111px;">Supply Deadline</p>

			<select style="position:absolute; left: 467px; top: 395px; width:
44px;" name="dd">
				<option value="select" selected="selected">DD</option>
				<?php for($i = 1; $i < 32; $i++) echo
"<option>$i</option>"; ?>
			</select>

			<select style="position:absolute; left: 514px; top: 395px; width:
48px;" name="mm">
				<option value="select" selected="selected">MM</option>
				<?php for($i = 1; $i < 13; $i++) echo
"<option>$i</option>"; ?>
			</select>

			<select style="position:absolute; width: 61px; left: 565px; top:
394px;" name="yyyy">
				<option value="select"
selected="selected">YYYY</option>
				<?php $temp = 0;
				$td = date("Y");

					for($i = 0; $i < 6; $i++)
					{
						$temp = $td + $i;
						echo "<option>$temp</option>";
					}?>
			</select>
	</form>
<?php
}
				/* FUNCTION FOR 3RD PAGE / DECISION ZONE */
function process()
{
?>
	<a href="BPEL.php" target="_parent" class="style1">Home</a>
<?php
				/* CONNECTING TO MYSQL SERVER */
	$con = mysql_connect("localhost", "root", "microsoft");
	if(!$con) die("FAILED TO CONNECT WITH MYSQL SERVER");
```

```
	mysql_select_db("category_database", $con);
	$links = $_POST["links"];

		foreach ($links as $key => $value)
		{
			$result01 = mysql_query("SELECT * FROM category WHERE
Link = '$value'");
			while($row01 = mysql_fetch_array($result01))
				$db_info[] = $row01["Database"].".xml";
		}

				/* START PARSING XML FORMAT FILE */

	global $ReturnValue1;			/* DECLARATION */
	global $ReturnValue2;
	global $ReturnValue3;
	global $ReturnValue4;
	global $ReturnValue5;

		$ReturnValue1 = "?"; /* INITIALIZATION */
		$ReturnValue2 = "?";
		$ReturnValue3 = "?";
		$ReturnValue4 = "?";
		$ReturnValue5 = "?";

	global $IsRightTag1;
	global $IsRightTag2;
	global $IsRightTag3;
	global $IsRightTag4;
	global $IsRightTag5;

		$IsRightTag1 = false;
		$IsRightTag2 = false;
		$IsRightTag3 = false;
		$IsRightTag4 = false;
		$IsRightTag5 = false;

	/* FOR START TAG */

	function XmlStartElement($Parser, $Name, $Attrs)
	{
		global $IsRightTag1;
		global $IsRightTag2;
		global $IsRightTag3;
		global $IsRightTag4;
		global $IsRightTag5;
```

```
		if($Name == "REG_NUM")
			$IsRightTag1 = true;
		if($Name == "UNIT_COST")
			$IsRightTag2 = true;
		if($Name == "DEADLINE")
			$IsRightTag3 = true;
		if($Name == "FREE_WORKER")
			$IsRightTag4 = true;
		if($Name == "TOTAL_WORKER")
			$IsRightTag5 = true;
}

/* FOR END TAG */

function XmlEndElement($Parser, $Name)
{
	global $IsRightTag1;
	global $IsRightTag2;
	global $IsRightTag3;
	global $IsRightTag4;
	global $IsRightTag5;

		if($Name == "REG_NUM")
			$IsRightTag1 = false;
		if($Name == "UNIT_COST")
			$IsRightTag2 = false;
		if($Name == "DEADLINE")
			$IsRightTag3 = false;
		if($Name == "FREE_WORKER")
			$IsRightTag4 = false;
		if($Name == "TOTAL_WORKER")
			$IsRightTag5 = false;
}

/* TAKING DATA IN BETWEEN START TAG AND END TAG */

function XmlTextData($Parser, $Data)
{
	global $ReturnValue1;
	global $ReturnValue2;
	global $ReturnValue3;
	global $ReturnValue4;
	global $ReturnValue5;

		global $IsRightTag1;
```

```
                        global $IsRightTag2;
                        global $IsRightTag3;
                        global $IsRightTag4;
                        global $IsRightTag5;

                if($IsRightTag1)
                        $ReturnValue1 = $Data;
                if($IsRightTag2)
                        $ReturnValue2 = $Data;
                if($IsRightTag3)
                        $ReturnValue3 = $Data;
                if($IsRightTag4)
                        $ReturnValue4 = $Data;
                if($IsRightTag5)
                        $ReturnValue5 = $Data;
        }

        global $iter;
        $iter = 0;

        /* TAKING EVERY .XML FORMAT DATA AS INPUT AND PARSE*/
        /* AS THE REPLY WILL COME AS A PACKET OF XML DATA */
        /* - WE NEED TO PARSE AS APPROPIATE */

        foreach ($db_info as $key => $value)
        {
                $XmlParser = xml_parser_create();
                xml_set_element_handler($XmlParser, "XmlStartElement",
"XmlEndElement");
                xml_set_character_data_handler($XmlParser, "XmlTextData");

                        $strXML = implode("",file($value));          /* TAKING EACH
XML FILE */
                        xml_parse($XmlParser, $strXML, true);
                        xml_parser_free($XmlParser);

                        $combine_detail[] =
$ReturnValue1."|".$ReturnValue2."|".$ReturnValue3."|".$ReturnValue4."|".$ReturnValu
e5;             /* FORMATTING DATA */
                        $iter++;
        }

        /* TAKING SUPPLY DEADLINE */
        /* ACTUALLY CONVERTING VALUE AS INTEGER */

        $dd = $_POST["dd"];                     /* DAY */
```

```
	$dd = (int)$dd;
	$mm = $_POST["mm"];			/* MONTH */
	$mm = (int)$mm;
	$yyyy = $_POST["yyyy"];		/* YEAR */
	$yyyy = (int)$yyyy;

		/* READING DATE FROM SYSTEM */
		/* ACTUALLY CONVERTING VALUE AS INTEGER */

		$sys_dd = date("d");		/* DAY */
		$sys_dd = (int)$sys_dd;
		$sys_mm = date("m");			/* MONTH */
		$sys_mm = (int)$sys_mm;
		$sys_yyyy = date("Y");			/* YEAR */
		$sys_yyyy = (int)$sys_yyyy;

	/* SELECTING COMPANY -> ACTUALLY TAKING THE LINKS */
	/* TAKING QUANTITY [NEEDS TO SUPPLY] FROM BUYER */

	$links = $_POST["links"];
	$quantity = $_POST["quantity"];

	/* VALIDITY FOR QUANTITY IF ANY ONE GIVES STRING RATHER
THAN DECIMAL VALUE */

	try {
			$quantity = (int)$quantity;
	}
	catch(Exception $e) {
			echo "<br><label><center><font color = \"red\"><b>Error: Enter
a Valid Quantity. Value must a Decimal Number without any
Precision.</b></font></center></label>";
	}

		/* IF BUYER SELECTS INVALID DATE */
		/* VALIDITY FOR QUANTITY IF ITS EMPTY */
		/* IF PROVIDE A NEGATIVE VALUE OR ZERO IN QUATITY FIELD
*/

		if(($yyyy-1 < $sys_yyyy) && (($dd < $sys_dd) || ($mm < $sys_mm)))
			echo "<br><br><br><label><center><font color =
\"red\"><b>Error: You have Selected an Invalid Date. Make Sure You Know the Actual
Date.</b></font></center></label>";

		else if($_POST["quantity"] == "")
```

```
                    echo "<br><br><br><label><center><font color =
\"red\"><b>Error: You have Put the Quantity Field
Empty.</b></font></center></label>";

               else if($quantity <= 0)
                    echo "<br><br><br><label><center><font color =
\"red\"><b>Error: Enter a Valid Quantity. Value must a Decimal Number [> 0] without
any Precision.</b></font></center></label>";

               else          /* DECISION BLOCK */
               {
                    /* SENDING SUPPLY DATE TO CONVERT IT INTO
SECONDS */
                    /* SENDING TODAY'S DATE TO CONVERT IT INTO
SECONDS */
                    /* DIVIDING THOSE BY TOTAL SECONDS IN A DAY [24
HOUR] WE GET DAY REMAINING */

                    $new_supply_con = mktime(0, 0, 0, $mm, $dd, $yyyy);
                    $today_con = mktime(0, 0, 0, $sys_mm, $sys_dd, $sys_yyyy);
                    $buf_diff = ($new_supply_con - $today_con) / ((60*60)*24);

                         /* ONLY INFORMATION REGARDING A SINGLE
COMPANY */

                         if($iter == 1)
                         {
                              $iter = $iter - 1;        /* ACCESS 0'TH INDEX */

                                   /* PARSING FORMATTED
INFORMATION BELOW */

                                   list($parse_reg, $parse_ucost, $com_dmy,
$parse_fworker, $parse_tworker) = split('[|]', $combine_detail[$iter]);

                              /* FINDING OUT INFORMATION FROM
COMBINE DATE [00/00/0000] */

                              $parse_day = substr($com_dmy, 0, 2);        /*
TAKING DATE */
                              $parse_month = substr($com_dmy, 3, 2);     /*
TAKING MONTH */
                              $parse_year = substr($com_dmy, 6);          /*
TAKING YEAR */

                                   /* CONVERTING INTO INTEGER */
```

```
                                        $parse_day = (int) $parse_day;
                                        $parse_month = (int) $parse_month;
                                        $parse_year = (int) $parse_year;
                                        $parse_ucost = (int) $parse_ucost;
                                        $parse_fworker = (int) $parse_fworker;
                                        $parse_tworker = (int) $parse_tworker;

                                /* RETRIEVING INFORMATION REGARDING
PER-DAY CAPACITY FROM DATABASE */

                                foreach ($links as $key => $value)
                                {
                                        $result02 = mysql_query("SELECT *
FROM category WHERE Link = '$value'");
                                        while($row01 =
mysql_fetch_array($result02))
                                        {
                                                $capacity_day =
$row01["Per_Cap"];
                                                $comp_name = $row01["Name"];
                                        }
                                        $web_site = $value;             /* TAKING
WEB ADDRESS */
                                }

                                        /* FINDING OUT PER WORKER
CAPABILITY */
                                        /* PER WORKER CAPABILITY
MULTIPLY BY TOTAL WORKER AND AGAING */
                                        /* - MULTIPLYING RESULTING VALUE
BY DATE THAT WILL GATE FROM TODAY */
                                        /* - TO UNTIL SUPPLY */

                                        $per_worker_cap = $capacity_day /
$parse_tworker;
                                        $calc01 = ($per_worker_cap *
$parse_tworker) * $buf_diff;

                                /* GIVE REPLY IF THE COMPANY DON'T
HAVE THE CAPACITY TO PRODUCE */
                                /* - QUANTITY THAT HAS BEEN ORDERED.
BASICALLY HERE WE FIND OUT */
                                /* - WHETHERCOMPANY CAN MANAGE HIS
WORKER TO TAKE THE ORDER. ACTUALLY */
```

```
                                        /* - COMPANY HAS PENDING ORDER SO IT
MAY HAPPEN IT CAN TAKE THE ORDER */
                                        /* - IF ALL THE WORKER OR A PORTION OF
WORKER IS NOT ASSING IN ANOTHER */
                                        /* - TASK. SO ITS NEED TO CHECK */

                                        if($calc01 < $quantity)
                                                echo "<br><br><br><label><center><font
color = \"red\"><b>Message: Sorry ... Selected Company don't have the Capability to
Take this Contract.</b></font></center></label>";
                                        else
                                        {
                                                /* THE DAY COMPANY WILL GET
WHEN IT WILL FINISH ITS CURRENT TASK */

                                                $deadline_con = mktime(0, 0, 0,
$parse_month, $parse_day, $parse_year);
                                                $calc02 = ($new_supply_con -
$deadline_con) / ((60*60)*24);

                                                /* CHECKING WHETHER FULL UNIT
CAN ENGAGE ON THE TASK */

                                                if(calc02 > 0)
                                                        $comp_add_worker = $calc02 *
(($parse_tworker - $parse_fworker) * $per_worker_cap);          /* RELEASE
WORKER CAN CONTRIBUTE HOW MUCH */
                                                else
                                                        $comp_add_worker = 0;

                                                /* COMBINING THE TASK OF TOTAL
UNIT */

                                                $calc03 = ($buf_diff * ($parse_fworker *
$per_worker_cap)) + $comp_add_worker;

                                                /* WHTHER WORKING WITH FULL
UNIT COMPANY CAN PERFORM THE TASK */

                                                if($calc03 < $quantity)
                                                        echo
"<br><br><br><label><center><font color = \"red\"><b>Message: Sorry ... Company
don't have Slot for this Contract.</b></font></center></label>";
                                                else
                                                {
```

```
                                                $result04 = mysql_query("SELECT
* FROM server WHERE Reg_Num = '$parse_reg'");
                                                while($row02 =
mysql_fetch_array($result04))
                                                        $email_addr =
$row02["Email_Addr"];

                                                echo
"<br><br><label><center><font color = \"green\"><b>Message: You can Make Contract
with the Company.</b></font></center></label>";
                                                echo "<br><center><font color =
\"magenta\"><b>Name: ".$comp_name."</b></font></center></label>";
                                                echo "<br><center><font color =
\"magenta\"><b>Unit Cost: $".$parse_ucost."</b></font></center></label>";
                                                echo "<br><center><font color =
\"magenta\"><b>Email Address: ".$email_addr."</b></font></center></label>";
                                                echo "<br><center><font color =
\"blue\"><b>Visit Web-Site: <a href =
>".$web_site."</a></b></font></center></label>";
                                                echo "<form method=\"post\"
action=\"Email.php\" style=\"position:absolute; left: 700px; top: 89px; width: 100px;
height: 24px;\">
                                                <input type=\"hidden\"
name=\"addr\" value=\"$email_addr\" />
                                                <input type=\"submit\"
name=\"Email\" value=\"Email\" />
                                                </form>";
                                        }
                                }
                        }

                        /* WHEN MORE THAN ONE COMPANY INVOLVED i.e. WE
ARE CHOOSING */
                        /* - ONE BEST COMPANY AMONG THOSE SELECTED
COMPANIES */

                        else if($iter > 1)
                        {
                                /* PARSING FORMATTED INFORMATION BELOW */

                                for($i = 0; $i < $iter; $i++)
                                {
                                        list($parse_reg[], $parse_ucost[], $com_dmy[],
$parse_fworker[], $parse_tworker[]) = split('[|]', $combine_detail[$i]);
                                }
```

```
                                        /* CONVERTING INTO INTEGER */

                                        for($i = 0; $i < $iter; $i++)
                                        {
                                                $parse_day[] = substr($com_dmy[$i], 0, 2);
                                                $parse_month[] = substr($com_dmy[$i], 3,
2);
                                                $parse_year[] = substr($com_dmy[$i], 6);
                                                $dec_parse_ucost[$i] = (int)
$parse_ucost[$i];
                                                $dec_parse_fworker[$i] = (int)
$parse_fworker[$i];
                                                $dec_parse_tworker[$i] = (int)
$parse_tworker[$i];
                                        }

                                        /* RETRIEVING INFORMATION REGARDING
PER-DAY CAPACITY FROM DATABASE */

                                        foreach ($links as $key => $value)
                                        {
                                                $result03 = mysql_query("SELECT *
FROM category WHERE Link = '$links[$key]'");
                                                while($row01 =
mysql_fetch_array($result03))
                                                        $capacity_day[] =
$row01["Per_Cap"];
                                        }

                                                /* FINDING OUT PER WORKER
CAPABILITY */
                                                /* PER WORKER CAPABILITY
MULTIPLY BY TOTAL WORKER AND AGAING */
                                                /* - MULTIPLYING RESULTING VALUE
BY DATE THAT WILL GATE FROM TODAY */
                                                /* - TO UNTIL SUPPLY */

                                                for($i = 0; $i < $iter; $i++)
                                                        $per_worker_cap[] =
$capacity_day[$i] / $dec_parse_tworker[$i];
                                                for($i = 0; $i < $iter; $i++)
                                                        $calc01[] = ($per_worker_cap[$i] *
$dec_parse_tworker[$i]) * $buf_diff;

                                        $flag_c = 0;            /* VALIDATE WHICH HAS
CAPACITY TO HANDLE GIVEN QUANTITY */
```

```
for($i = 0; $i < $iter; $i++)
{
	if($calc01[$i] < $quantity)
	{
		$flag[] = "false";	/* SETTING FLAG FALSE IF DON'T HAS CAPABILITY */
		$flag_c = $flag_c - 1;
	}
	else
	{
		$flag[] = "true";	/* SETTING FLAG FALSE IF DON'T HAS CAPABILITY */
		$flag_c = $flag_c + 1;
	}
}

	/* THE DAY COMPANY WILL GET WHEN IT WILL FINISH ITS CURRENT TASK */

	for($i = 0; $i < $iter; $i++)
		$deadline_con[] = mktime(0, 0, 0, $parse_month[$i], $parse_day[$i], $parse_year[$i]);
	for($i = 0; $i < $iter; $i++)
		$calc02[] = ($new_supply_con - $deadline_con[$i]) / ((60*60)*24);

	/* CHECKING WHETHER FULL UNIT CAN ENGAGE ON THE TASK */

	for($i = 0; $i < $iter; $i++)
	{
		if($calc02[$i] > 0)
			$comp_add_worker[$i] = $calc02[$i] * (($parse_tworker[$i] - $parse_fworker[$i]) * $per_worker_cap[$i]);
		else
			$comp_add_worker[$i] = 0;
	}

		$click = 0;
		$temp_reg = "?";
		$temp_cost = 0;

	/* THIS WILL SHOW WHEN NONE OF THE SELECTED COMPANIES CAN TAKE ORDER */
```

```
                                        if($flag_c == -$iter)
                                                echo
"<br><br><br><label><center><font color=\"red\"><b>Message: None of the Selected
Companies have Slot for Your Order.</b></font></center></label>";
                                        else
                                        {
                                                for($i = 0; $i < $iter; $i++)
                                                {
                                                        if($flag[$i] == "true")
                                                        {
                                                                $calc03 = ($buf_diff
* ($parse_fworker[$i] * $per_worker_cap[$i])) + $comp_add_worker[$i];
                                                                if($click > 0)
                                                                {

        if($dec_parse_ucost[$i] > $dec_parse_ucost[$i-1])
                                                                        {

        $temp_reg = $parse_reg[$i-1];

        $temp_cost = $dec_parse_ucost[$i-1];
                                                                        }
                                                                        else
                                                                        {

        $temp_reg = $parse_reg[$i];

        $temp_cost = $dec_parse_ucost[$i];
                                                                        }
                                                                }
                                                                $click++;
                                                        }
                                                }

                                                $result03 = mysql_query("SELECT
* FROM category WHERE Reg_Num = '$temp_reg'");
                                                while($row01 =
mysql_fetch_array($result03))
                                                {
                                                        $comp_name =
$row01["Name"];
                                                        $web_site = $row01["Link"];
                                                }

                                                $result04 = mysql_query("SELECT
* FROM server WHERE Reg_Num = '$temp_reg'");
```

```
                                                    while($row02 =
mysql_fetch_array($result04))
                                                        $email_addr =
$row02["Email_Addr"];

                                                    echo
"<br><br><label><center><font color = \"green\"><b>Message: You can Make Contract
with the Company.</b></font></center></label>";
                                                    echo "<br><center><font color =
\"magenta\"><b>Name: ".$comp_name."</b></font></center></label>";
                                                    echo "<br><center><font color =
\"magenta\"><b>Unit Cost: $".$temp_cost."</b></font></center></label>";
                                                    echo "<br><center><font color =
\"magenta\"><b>Email Address: ".$email_addr."</b></font></center></label>";
                                                    echo "<br><center><font color =
\"blue\"><b>Visit Web-Site: <a href =
>".$web_site."</a></b></font></center></label>";
                                                    echo "<form method=\"post\"
action=\"Email.php\" style=\"position:absolute; left: 700px; top: 89px; width: 100px;
height: 24px;\">
                                                    <input type=\"hidden\"
name=\"addr\" value=\"$email_addr\" />
                                                    <input type=\"submit\"
name=\"Email\" value=\"Email\" />
                                                    </form>";
                                                }
                                    }
                            }
}
?>
</body>

<?php
global $action;          /* HANDLING EACH EVENT */

switch($action)
          {
        case "result":   /* MOVE TO 2ND PAGE */
                                                    result();
        break;

        case "process":          /* MOVE TO 3RD PAGE */
                                                    process();
        break;

        default:
```

```
                                                    index();          /* MAIN PAGE */
          break;
        }
?>
</html>

/*************************************************************************/

Code: Email.PHP

/*************************************************************************/

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Email to the Company</title>
<link href="Design.css" rel="stylesheet" type="text/css" />
<h1 align="center" class="style4">Contact via Email</h1>
</head>

<body bgcolor="#CCCCFF">

        <form name="email" style="position:absolute; left: 86px; top: 5px; width: 595px;
height: 374px;" method="post" action="">
        <a href="BPEL.php" class="style3" style="position:absolute; left: -66px; top:
24px;">Home</a>
                <br/>
                <br/><br/><br/>
                        <p class="style1">
                                <label class="style1" style="position:absolute; left: 116px;
top: 145px;">Receiver Email</label>
                                <?php
                                echo " <input type=\"text\" name=\"remail\"
value=\"$addr\" disabled=\"disabled\" style=\"position:absolute; left: 223px; top: 146px;
width: 231px;\"/>";
                                ?>
                                <br/><br/>
                                <label class="style1" style="position:absolute; left: 116px;
top: 176px;">Sender Email</label>
                                <input type="text" name="semail"
style="position:absolute; left: 223px; top: 176px; width: 232px;"/>
                        </p>
```

```
                        <p class="style1">
                                <label class="style2" style="position:absolute; left: 116px;
top: 206px;">Subject</label>
                <input name="subject" style="position:absolute; left: 223px; width:
316px; top: 205px;" type="text" id="textfield" size="15" maxlength="20">
                                <label class="style1" style="position:absolute; left: 115px;
top: 252px;">Message</label>
                                <textarea name="message" style="position:absolute; left:
223px; top: 257px; width: 317px; height: 100px;" cols="50" rows="6"></textarea>
                        </p>
                <p align="right" class="bod"><br></p>
        </form>

                <input type="submit" name="post" style="position:absolute; left: 542px;
top: 391px;"value="Send Mail" />

</body>
</html>

/*************************************************************************/
Code: Reply in XML Format

/*************************************************************************/
# ak_infodb.XML

<?xml version="1.0" encoding="iso-8859-1"?>

<!--
- Database Information: 'ak_infodb'
-->

<INFODB>
        <REG_NUM>0011</REG_NUM>
    <UNIT_COST>10</UNIT_COST>
    <DEADLINE>07/01/2007</DEADLINE>
    <FREE_WORKER>30</FREE_WORKER>
    <TOTAL_WORKER>50</TOTAL_WORKER>
</INFODB>

# al_infodb.XML

<?xml version="1.0" encoding="iso-8859-1"?>

<!--
- Database Information: 'al_infodb'
-->
```

```
<INFODB>
        <REG_NUM>0003</REG_NUM>
      <UNIT_COST>8</UNIT_COST>
      <DEADLINE>15/12/2006</DEADLINE>
      <FREE_WORKER>5</FREE_WORKER>
      <TOTAL_WORKER>10</TOTAL_WORKER>
</INFODB>
```

bo_infodb.XML

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!--
- Database Information: 'bo_infodb'
-->

<INFODB>
        <REG_NUM>0012</REG_NUM>
      <UNIT_COST>15</UNIT_COST>
      <DEADLINE>00/00/0000</DEADLINE>
      <FREE_WORKER>20</FREE_WORKER>
      <TOTAL_WORKER>20</TOTAL_WORKER>
</INFODB>
```

bt_infodb.XML

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!--
- Database Information: 'bt_infodb'
-->

<INFODB>
        <REG_NUM>0001</REG_NUM>
      <UNIT_COST>8</UNIT_COST>
      <DEADLINE>15/12/2006</DEADLINE>
      <FREE_WORKER>5</FREE_WORKER>
      <TOTAL_WORKER>10</TOTAL_WORKER>
</INFODB>
```

da_infodb.XML

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!--
```

```
- Database Information: 'da_infodb'
-->

<INFODB>
        <REG_NUM>0009</REG_NUM>
    <UNIT_COST>8</UNIT_COST>
    <DEADLINE>01/02/2007</DEADLINE>
    <FREE_WORKER>0</FREE_WORKER>
    <TOTAL_WORKER>25</TOTAL_WORKER>
</INFODB>

# dm_infodb.XML

<?xml version="1.0" encoding="iso-8859-1"?>

<!--
- Database Information: 'dm_infodb'
-->

<INFODB>
        <REG_NUM>0006</REG_NUM>
    <UNIT_COST>7</UNIT_COST>
    <DEADLINE>20/12/2006</DEADLINE>
    <FREE_WORKER>5</FREE_WORKER>
    <TOTAL_WORKER>15</TOTAL_WORKER>
</INFODB>

# ee_infodb.XML

<?xml version="1.0" encoding="iso-8859-1"?>

<!--
- Database Information: 'ee_infodb'
-->

<INFODB>
        <REG_NUM>0009</REG_NUM>
    <UNIT_COST>15</UNIT_COST>
    <DEADLINE>20/12/2006</DEADLINE>
    <FREE_WORKER>15</FREE_WORKER>
    <TOTAL_WORKER>30</TOTAL_WORKER>
</INFODB>

# hk_infodb.XML

<?xml version="1.0" encoding="iso-8859-1"?>
```

```
<!--
- Database Information: 'hk_infodb'
-->

<INFODB>
        <REG_NUM>0004</REG_NUM>
    <UNIT_COST>12</UNIT_COST>
    <DEADLINE>12/12/2006</DEADLINE>
    <FREE_WORKER>10</FREE_WORKER>
    <TOTAL_WORKER>20</TOTAL_WORKER>
</INFODB>

# mx_infodb.XML

<?xml version="1.0" encoding="iso-8859-1"?>

<!--
- Database Information: 'mx_infodb'
-->

<INFODB>
        <REG_NUM>0007</REG_NUM>
    <UNIT_COST>10</UNIT_COST>
    <DEADLINE>30/12/2006</DEADLINE>
    <FREE_WORKER>0</FREE_WORKER>
    <TOTAL_WORKER>15</TOTAL_WORKER>
</INFODB>

# nl_infodb.XML

<?xml version="1.0" encoding="iso-8859-1"?>

<!--
- Database Information: 'nl_infodb'
-->

<INFODB>
        <REG_NUM>0010</REG_NUM>
    <UNIT_COST>12</UNIT_COST>
    <DEADLINE>01/01/2007</DEADLINE>
    <FREE_WORKER>0</FREE_WORKER>
    <TOTAL_WORKER>20</TOTAL_WORKER>
</INFODB>

# rs_infodb.XML
```

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!--
- Database Information: 'rs_infodb'
-->

<INFODB>
        <REG_NUM>0002</REG_NUM>
      <UNIT_COST>6</UNIT_COST>
      <DEADLINE>15/12/2006</DEADLINE>
      <FREE_WORKER>5</FREE_WORKER>
      <TOTAL_WORKER>20</TOTAL_WORKER>
</INFODB>
```

ss_infodb.XML

```
<?xml version="1.0" encoding="iso-8859-1"?>

<!--
- Database Information: 'ss_infodb'
-->

<INFODB>
        <REG_NUM>0005</REG_NUM>
      <UNIT_COST>8</UNIT_COST>
      <DEADLINE>15/12/2006</DEADLINE>
      <FREE_WORKER>5</FREE_WORKER>
      <TOTAL_WORKER>20</TOTAL_WORKER>
</INFODB>
```

Figure List

Bibliography

Web Link:

1. www.w3schools.com
2. www.oracle.org
3. www.programmersheaven.com
4. www.w3.org
5. www.msdn.microsoft.com

Other:

1. [ADA] Adams, Holt. *Asynchronous operations and Web services, Part 3*. IBM developerWorks, Oct 2002.
2. [ARK] Arkin, Assaf, et. al. Web Service Choreography Interface 1.0. www.sun.com/software/xml/developers/wsci/wsci-spec-10.pdf, 2002.
3. [BAK] Baker, Jeanne, P. Fingar, and H. Smith. *Value Chain Integration: The Next Frontier*. Internet World, July 2002.
4. [BAU] Bau, David. *SOAP Conversation Protocol (SOAP Conversation) 1.0*. BEA dev2dev (www.dev2dev.bea.com), June 2002.
5. [CSC] CSC. *13th Annual Critical Issues of IS Management Survey*. CSC (www.csc.com/survey), 2000.
6. [DUF] Duftler, Matthew and R. Khalaf. *Business Process with BPEL4WS: Learning BPEL4WS, Part 3*. IBM developerWorks, Oct 2002.
7. [DUI] Duivestein, Sander. *Web Services and Workflow*. Web Services Architect (www.webservicesarchitect.com), Sep 2001.
8. [HAN] Hanson, James E., P. Nandi, and D. Levine. *Conversation-enabled Web Services for Agents and e-Business*. IBM Research Paper.
9. [HUR] Hurwitz Group. *Ten Pillars for World Class Business Process Management*. The Hurwitz Group (www.hurwitz.com), July 2001.
10. [KUM] Kumaran, Santhosh and P. Nandi, *Conversational Support for Web Services: The next stage of Web services abstraction*. IBM developerWorks, Sep 2002.
11. [LEE] Lee, Juhnyoung, J. Yang, and J. Chung. *Winslow: A Business Process Management System with Web Services*. IBM Research Report, Nov 2002.
12. [LEY] Leymann, Frank and D. Roller. *Business processes in a Web services world*. IBM developerWorks, Aug 2002.
13. [LEY2] Leymann, Frank, D. Roller, and M. Schmidt. *Web services and business process management*. IBM Systems Journal, Volume 41-2, 2002.
14. [META] Meta Group, *Web Services: Stages of Adoption*. The Meta Group, 2002.
15. [MIC] Microsoft. *BizTalk Orchestration: A Technology for Orchestrating Business Interactions*. Microsoft Corporation, June 2000.
16. [ORI] O'Riordan, David. *Business Process Standards for Web Services*. Tect.
17. [SHA] Shapiro, Robert. *A Comparison of XPDL, BPML, and BPEL4WS*. Cape Visions, May 2001.
18. [SHE] Sherman, Doron. *Orchestrating Asynchronous Web Services*. Collaxa (www.collaxa.com), Feb 2002.
19. [SNE] Snell, James. *Automating business processes and transactions in Web services*. IBM developerWorks, Aug 2002.

20. [WEE| Weerawarana, Sanjiva and C. Francisco. *Business processes: Understanding BPEL4WS, Part 1*. IBM developerWorks, Aug 2002.
21. [WES] Wesley, Ajamu. *WSFL in action, Part 1*. IBM developerWorks (www.ibm.com/developerworks), Jan 2002.