SOEGOV: A SERVICEORIENTED E-GOVERNANCE APPROACH FOR EFFECTIVE SERVICE DELIVERY

Rama Krushna Das[1], Manas Ranjan Patra[2] and Sujata Patnaik[2]

[1]National Informatics Centre, Berhampur, India
[2]Department of Computer Science, Berhampur University, India

ABSTRACT

In recent years there has been an exponential growth of e-Governance in India. It is growing to such a scale that requires full attention of the Government to ensure collaboration among different government departments, private sectors and Non-Governmental Organisations(NGOs). In order to achieve successful e-Governance, Government has to facilitate delivery of services to citizens, business houses and other public or private organisations according to their requirements. In this paper, we have proposed integration of different government departments using a Service Oriented e-Governance(SOeGov) approach with web services technology and Service Oriented Architecture(SOA). The proposed approach can be effectively used for achieving integration and interoperability in an e-Governance system.We have demonstrated the working of our approach through a case study where integration of several departments of the provincial Government of Odisha (India) has been made possible.

KEYWORDS

e-Governance, SOeGov, SWeP, Service Composition, Service Orchestration, Service Design, Service Monitoring, Service Deployment.

1. INTRODUCTION

E-Governance facilitates governmental information to citizens and other stakeholders in electronic form. Besides this, effective service delivery to citizens and empowerment of people through access to information without much intervention of bureaucracy are some of the primary goals of e-Governance [1].Typical characteristics of e-Governance systems include, need forhigh interoperable, large-scale, distributed, and heterogeneous systems cutting acrossgeographical boundaries and administrative domains. Therefore, achieving interoperabilityamong e-Governance applications towards seamless integration and information exchange is ofparamount importance.The above characteristics of e-Governance systems necessitate the choice of a suitable designapproach to build applications to realize a pragmatic system [2].The Service Oriented design approach to e-Governance combines Information and Communication Technology(ICT) with service delivery goals and enables various government departments to re-use services that are already developed. The purpose is to provide anagile Service Oriented solution for integrating different government departments irrespective of the underlying technology they are using. It requires to change existing applications, data, and content into web services using a special approach without touching the existing applications.For implementing these concepts, we have considered to develop an application named "***Single Window e-Governance Portal***" (SWeP) for transaction of Government services among different departments, irrespective of different underlying technologies they use. The SWeP works as a single source of information for all government content, and provides front end for all the government services provided by various government departments. Using web service technology the services and applications provided by different departments are made available on the SWeP.The government transactions made on this web service based implementation makes them interoperable, reusable, easily accessibleand easy to integrate.

2. SOeGov: Service Oriented e-Governance

A Service Oriented e-Governance (SOeGov) approach as discussed below is usedto change existing applications, data, and content into web services using a special approach without touching the existing applications. It presents a new software development paradigm that endorses direct association of different government departments and citizens, without depending on the delivery model. It involves several steps in realizing the SOeGov approach, namely, (1) Service identification, (2) Service Specification, (3) Service Realisation, (4) Service Publication, (5) Service Orchestration, (6) Service Deployment (Implementing Solution), and (7) Service Monitoring.The following Figure-1 depicts the activities associated with each of the above steps.

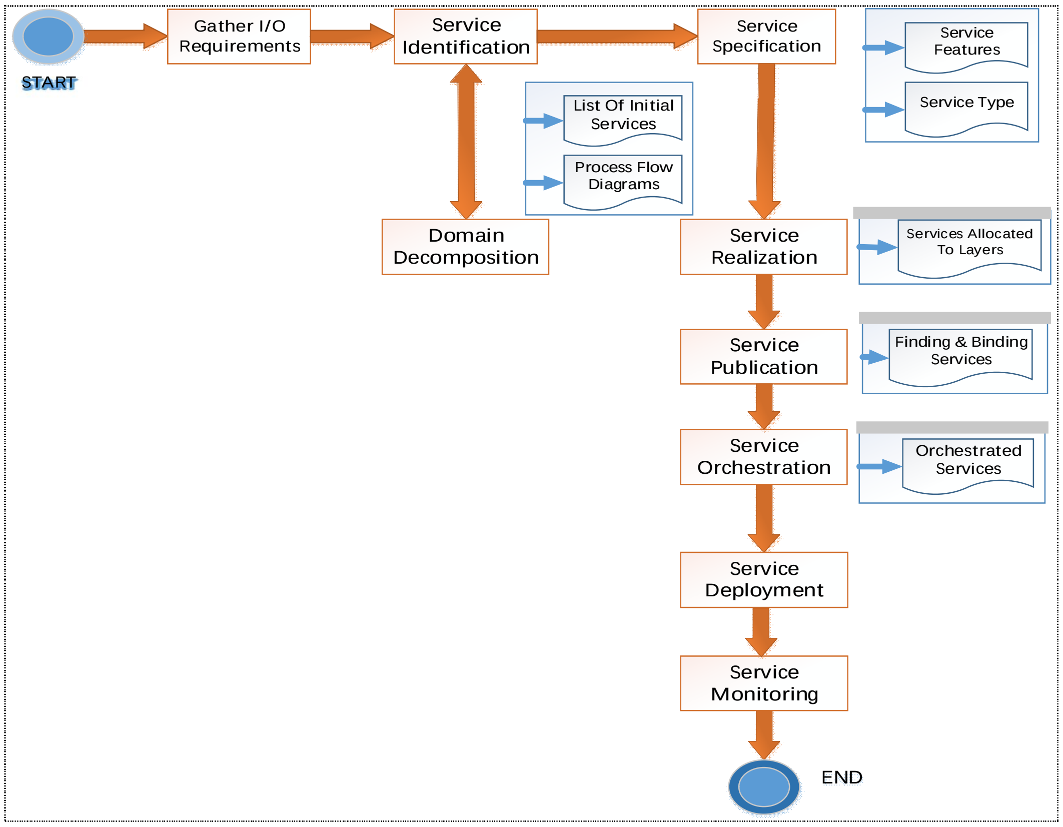


Figure 1. Steps involved in executing the SOeGov approach

A brief description of each step follows:

Step 1: Initially, we gather input requirements and necessary data by studying the system thoroughly.
Step 2: Next, we identify the services using certain use cases and domain decomposition techniques. The main artefacts at the end of this step are a list of initial services and process flows.
Step 3: In this step, we give a detailed specification of services. The artefacts of this step are service features and service types.

Step 4: This step involves service realization, where we assign the identified services to various layers in a Service Oriented stack.
Step 5: Next, all the identified services are published using Universal Description, Discovery, and Integration(UDDI) specifications, for finding and binding the services in the application.
Step 6: At this stage, the services are orchestrated using Business Process Execution Language(BPEL) tool.
Step 7: Next, the web services are implemented by choosing a language like PHP, Java or dot net.
Step 8: Finally,the services are monitored at run time as per the monitoring framework, which is independent of the business logic.

3. SERVICE IDENTIFICATION

Service identification can start with identifying different applicable services for the SWeP, The software developmentbegins by analysing an application domain and identifying a set of services to be provisioned.These services form the fundamental design objects upon which a complete system can be built. The process flow of the identified services are depicted in Figure 2.

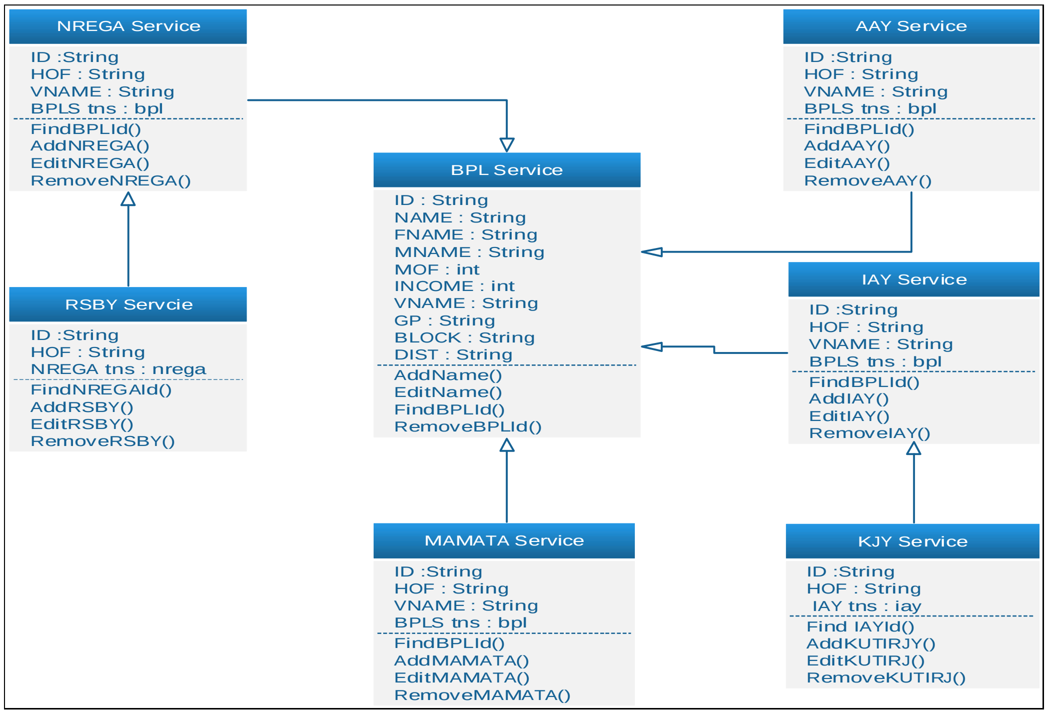


Figure 2. Process flow of the identified Services

4. SERVICE SPECIFICATION

In this, we have specified the comprehensive details about a service such as service messages, interfaces, Web Services Description Language(WSDL), service dependency, service invocation etc. During service specification, the artefacts comprising of the service orientation are formally defined, for example,atomicand composite services, also the components implementing them alongside with their interfaces.

5. SERVICE REALISATION

In Service realization, services and components are allocated to specific layers of SOA.

- Layer 1–**Citizen Access Layer** This layer provides users with a means to interact with services. At this layer, we have provided several interface channels such as computer (PC), Mobile, PDA or cellular, kiosks as shown in Figure 3.Using these interfaces, a citizen can access a service through the Service Interface Layer.
- Layer 2–**Service Interface Layer**: This layer gives access to the e-government portal through the application layer interface. It manages the interface proposed for the clients interacting with the e-Governanceapplication. For example, it contains a web server for the users connected via a web browser, a WAP server for the users connected via Mobile phones or PDAs. Separating this layer from the application layer makes the application accessible via various channels such as web browsers and even cellular phones, without having to change the application's implementation. Finally, the communications with the application layer will be done using the Simple Object Access Protocol(SOAP) over Hypertext Transfer Protocol(HTTP).
- Layer 3–**Application Layer**: As illustrated in Figure 3, the application in government Department "1" as a consumer starts a business process that includes executing tasks at governmentdepartments "2" and "3" as service providers. In addition to the notification service provided by the e-Governancecentral platform, the application at "A" will communicate with theBPEL at its premises to execute the complete process. The BPEL component invokes the department "2" Web service (IAY), department "3" Web service (KJY) and they communicate with each other through SOAP messages and execute according to the rules that had been set earlier in its orchestration engine.
- Layer 4–**Shared Service Layer**: At this layer both atomic and composite services are allocated. This Layer can be considered as a bridge between the higher and lower layers, and is characterized by a number of services that are carrying out individual business functions. The service layer usually includes service interfaces and message types. There are six types of services as shown in Figure 3. at this layer and the core services are integrated/orchestrated as per the rules predefined and stored in the Business Process Execution Language (BPEL) at the upper layer.
- Layer 5 – **Data Layer:** The data layer ensures proper storage and persistence of the government data. The management of the access rights to the data is ensured by the Database Management System (DBMS). A separate data layer reduces the architectural complication required to provide access to the government data. It allows departments to create and leverage services. The concept of data layer leads to increased opportunities for reusability and reduces system cost for long-term operations. We have achieved loose coupling with definition of data layer such that physical data can be reorganized without affecting applications.

6. SERVICE PUBLICATION

Universal Description, Discovery, and Integration (UDDI) is widely accepted as the standard for Web services publishing, querying, and discovery. A UDDI registry allows us to publish and browse web service references via SOAP (Simple Object Access Protocol) and HTTP interfaces.It is a standard Web Services discovery protocoland Web Services description format; a UDDI registry can contain metadata for any type of service, which are described by Web Services Description Language (WSDL). For theimplementationof SWeP, a "private" UDDI registry that runs as a servlet under Apache Tomcat is created as "OdisaGovt". MySQL is used as the persistent store for the local UDDI registry. The client API for the UDDI registry is developed using PHP.

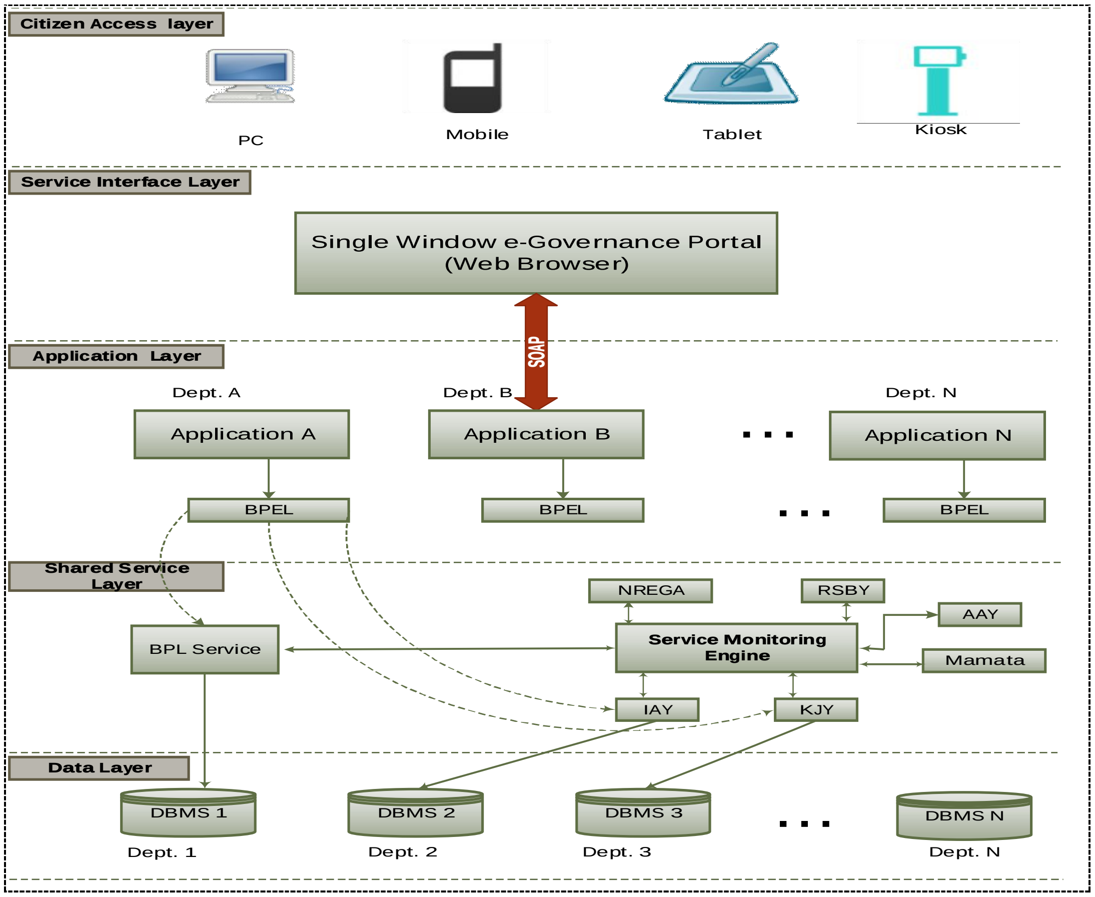


Figure 3. Different Layers of SWeP

7. Service Composition and Orchestration

Service Composition encourages the design of services that can be reused in multiplesolutions that are themselves made up of composed services. The capability of the serviceto be recomposed is independent of the size and complexity and size of the service composition, which is directly responsible for the agility promised bySOA. Bu reusing existing services SOA composes new solutions. The following section describes in detail the service composition and orchestration for our case study SWeP.

7.1. Building BPEL Process for SWeP

We have used Oracle Jdeveloper 11g (11.1.1.5.0) software for process orchestration and for building BPEL. JDeveloper[4] is a freeware IDE supplied by Oracle Corporation. It offers features for development in Java, XML,PHP SQL and PL/SQL, HTML, JavaScript, and BPEL. JDeveloper covers the complete development lifecycle from design through coding, debugging, optimization and profiling to deploying.With JDeveloper, Oracle has aimed to simplify application development by focusing on providing a visual and declarative approach to application development in addition to building an advanced coding-environment. Oracle JDeveloper integrates with the Oracle Application Development Framework (ADF) that simplifies application development. This Jdeveloper studio is used for SOA by installing an extension for SOA via the JDeveloper Updates wizard “check for updates”. Through this, the

SOA composite editor is installed. Now it is referred as Oracle SOA suite. Oracle SOA Suite[3] is a comprehensive, hot-pluggable software suite to build, deploy and manage Service-Oriented Architecture (SOA). This comes under the Oracle Fusion Middleware family.

A sample screen shot of the Oracle SOA suite is shown in Figure 4.

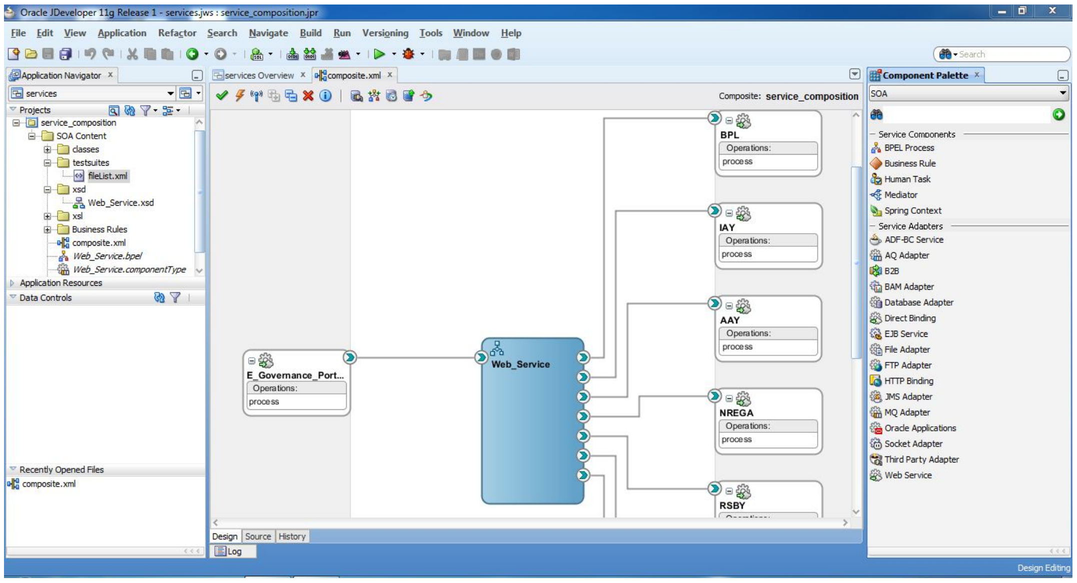


Figure 4. Oracle SOA suite screen shot

7.2 Developing a BPEL Process

In a typical case, the BPEL business process receives a request, the process invokes the involved Web services and then responds to the original caller. As the BPEL process communicates with other Web services, it depends on the WSDL descriptions of the Web services.

In BPEL defining a business process means essentially to create a new Web service that is a composite of existing services. In the new BPEL interfacethe composite Web service uses a set of port types, through which it provides operations like any other Web service.

To develop a new BPEL process, we go through the steps as follows:

1. Understand about the services involved in the project
2. Define the new WSDL for the BPEL process
3. Define partner link types
4. Define the BPEL process:
   - Define the namespaces
   - Define the partner links
   - Define variables
   - Define the process logic

These steps are applied to build BPEL process for our SWeP.

Step1: Services involved in the composition

The initiating point for writing BPEL process definition is to get information about the web services involved in the process. The proposed Single Window e-Governance Portal(SWeP) system involves the following autonomous services and composite web services like KJY and RSBY. The Complete SWeP service composition is shown in the Figure -7 below. The corresponding service composition code is shown in detail in the Table-1 below for reference.

RSBY Service: This web service is used to add names of RSBY beneficiaries. It is exposed to the service consumers with interface named *RSBY* as shown in Figure-5. This interface contains operation named "*execute( )*". Through this interface and operation, the consumer invokes the RSBY service. The invoked *RSBY service* routes the request to the BPL service to find that the candidate is enrolled in BPL or not. If it is enrolled in BPL then it invokes the NREGA service to find that the candidate is enrolled in NREGA or not. If it is not enrolled in NREGA, the NREGA service adds the candidate's name in NREGA database.

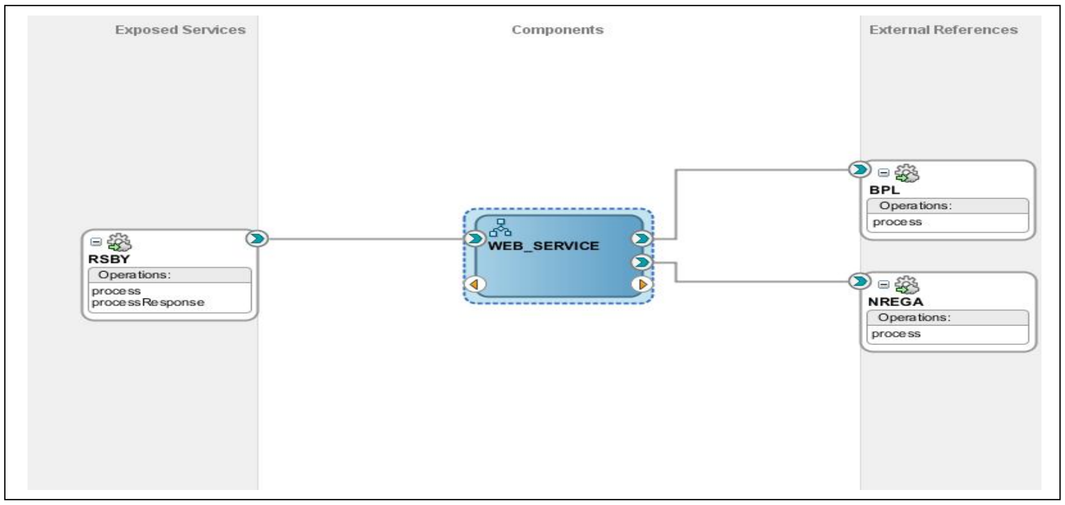


Figure 5. RSBY Service composition

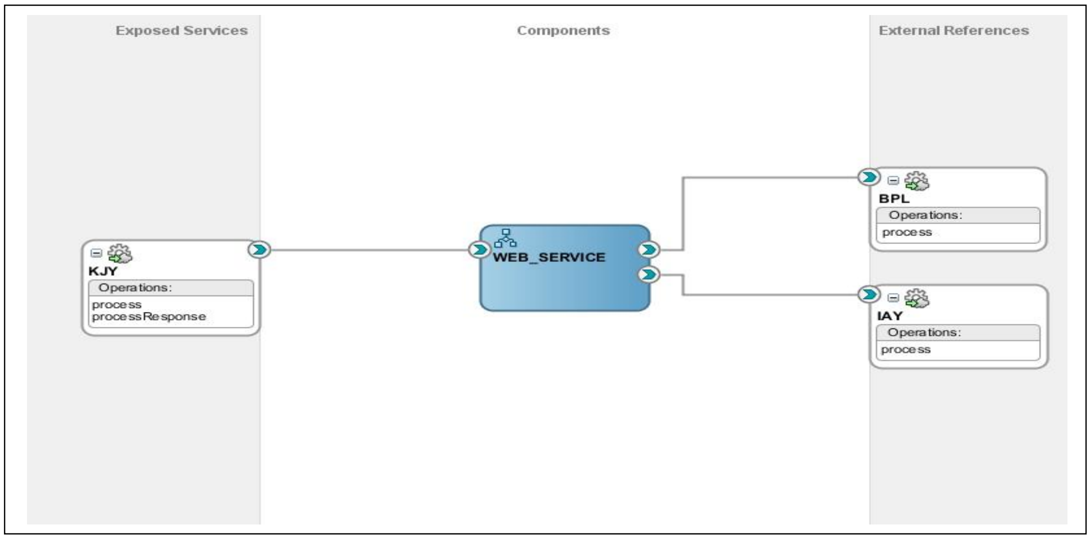


Figure 6. KJY Service composition

KJY Service: This web service is used to add names of KJY beneficiaries. It is exposed to the service consumers with interface named *KJY* as shown in Figure 6. This interface contains operation named "*execute( )*". Through this interface and operation, the consumer invokes the KJY service. The invoked *KJY service* routes the request to the BPL service to find that the candidate is enrolled in BPL or not. If it is enrolled in BPL then it invokes the IAY service to find that the candidate is enrolled in IAY or not. If it is not enrolled in IAY, the IAY service adds the candidates name in IAY database.

Step 2: Define the WSDL for SWeP

Since WSDL describes the complete contract for application communication,it plays an important role in the overall Web services architecture. This ensure interoperability at the service description layer. The WSDL code of the SWeP is shown in Table 1. The WSDL specification of the SWeP defines the available port types for its clients and its operations, messages, partner link types, and properties of interest to the process. The first part of WSDL is the target namespace. The name space is just like a XML schema definition. Anything that we name under WSDL definition automatically becomes part of name space.

Step 3: Define the Partner Link Types

The next step is to define the partner link types. Partner link types represent the communication between a BPEL process and the web services (Partners)involved, which the BPEL process invokes.

BPL Service: It is an atomic service with certain operations as shown in Figure-7. This service is used to find BPL status of Citizens for getting other BPL related benefits.

NREGA Service: It is an atomic service with certain operations as shown in Figure-7.This service is used to find NREGA status of Citizens for getting minimum 100 days of work and its related payment if he/she belongs to BPL category.

IAY Service: It is an atomic service with certain operations as shown in Figure-7. This service is used to find IAY status of Citizens for getting financial help for building a pucca house he/she belongs to BPL category.

AAY Service: It is an atomic service with certain operations as shown inFigure-7. This service is used to find AAY status of Citizens for getting minimum food grains each month under food security programme if he/she belongs to BPL category.

Mamata Service: It is an atomic service with certain operations as shown in Figure-7This service is used to find Mamata status of pregnant ladies of the BPL household.

RSBY Service: It is a composite service and hence this service maintains two more partner services, namely BPL Service and NREGA Service.This service is used to find RSBY status of Citizens, who is already enrolled in NREGA.

KJY Service: It is a composite service and hence this service maintains two more partner services, namely BPL Service and IAY Service.This service is used to find KJY status. of Citizens, who is already enrolled in IAY.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="Service" xmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="Service"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="Service" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
>
<xsd:documentation></xsd:documentation>
<!-- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                    TYPE DEFINITION - List of services participating in this BPEL process
   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<wsdl:types>
<xsd:schemaxmlns:xsd="http://www.w3.org/2001/XMLSchema" targetNamespace="Service">
<xsd:complexType name="Bpl Service">
<xsd:sequence>
<xsd:element name="ID" type="xsd:string"></xsd:element>
<xsd:element name="NAME" type="xsd:string" ></xsd:element>
                                                  <xsd:element name="FNAME" type="xsd:string" ></xsd:element>
                                                  <xsd:element name="MNAME" type="xsd:string" ></xsd:element>
                                                  <xsd:element name="MOF" type="xsd:int" ></xsd:element>
                                                  <xsd:element name="INCOME" type="xsd:int" ></xsd:element>
                                                  <xsd:element name="VNAME" type="xsd:string" ></xsd:element>
                                                  <xsd:element name="GP" type="xsd:string" ></xsd:element>
                                                  <xsd:element name="BLOCK" type="xsd:string" ></xsd:element>
                                                  <xsd:element name="DIST" type="xsd:string" ></xsd:element>
            </xsd:sequence>
</xsd:complexType>
<!—~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   PORT TYPE DEFINITION – It  groups a set of operations into  a logical service unit.
 ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->
<wsdl:portType name="Service">
<wsdl:operation name="searchBpls">
<wsdl:input message="tns:searchBplsRequest"/>
<wsdl:output message="tns:searchBplsResponse"/>
</wsdl:operation>
<wsdl:operation name="searchAays">
<wsdl:input message="tns:searchAaysRequest"></wsdl:input>
<wsdl:output message="tns:searchAaysResponse"></wsdl:output>
</wsdl:operation>
            <wsdl:operation name="searchNregs">
<wsdl:input message="tns:searchNregsRequest"></wsdl:input>
<wsdl:output message="tns:searchNregsResponse"></wsdl:output>
</wsdl:operation>
            <wsdl:operation name="searchRsbys">
<wsdl:input message="tns:searchRsbysRequest"></wsdl:input>
<wsdl:output message="tns:searchRsbysResponse"></wsdl:output>
</wsdl:operation>
            <wsdl:operation name="searchIays">
<wsdl:input message="tns:searchIaysRequest"></wsdl:input>
<wsdl:output message="tns:searchIaysResponse"></wsdl:output>
</wsdl:operation>
            <wsdl:operation name="searchKutirjs">
<wsdl:input message="tns:searchKutirjsRequest"></wsdl:input>
<wsdl:output message="tns:searchKutirjsResponse"></wsdl:output>
</wsdl:operation>
            <wsdl:operation name="searchMamatas">
<wsdl:input message="tns:searchMamatasRequest"></wsdl:input>
<wsdl:output message="tns:searchMamatasResponse"></wsdl:output>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="Service" type="tns:Service">
<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="searchBpls">
<soap:operationsoapAction="http://localhost/arati/Service.php" />
<wsdl:input>
<soap:body use="literal" namespace="Service" />
</wsdl:input>
<wsdl:output>
<soap:body use="literal" namespace="Service" />
</wsdl:output>
```

Step 4: Define the Business Process

The next step is to start writing the BPEL process definition. The BPEL process starts the execution of the business processafter receiving an incoming message from the client. In our example, the client invokes the BPEL process by sending an input message. Below, we briefly describe the main elements of the BPEL process construction:

Namespaces: In this, we define the target namespace and the namespaces to access the partner links WSDL, and the BPEL process WSDL description files. The name space for our problem is coded as shown in Table 2.

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<!--
//////////////////////////////////////////////////////////////////////////////////////////////////////////
  Oracle JDeveloper BPEL Designer
  Created: Thu May 29 19:36:13 IST 2014
  Author:  R K Das
  Type: BPEL 1.1 Process
  Purpose: Synchronous BPEL Process
//////////////////////////////////////////////////////////////////////////////////////////////////////
-->
<process name="Web_Service"
             targetNamespace="http://xmlns.oracle.com/services/service_composition/Web_Service"
xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
             xmlns:client="http://xmlns.oracle.com/services/service_composition/Web_Service"
xmlns:ora="http://schemas.oracle.com/xpath/extension"
xmlns:bpelx="http://schemas.oracle.com/bpel/extension"
        xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/">
```

Table 2. Name Space for SWeP

Partner Links: In this process, we identify the partner links which define different partners that interact with the BPEL process. The following is the code for partner links for our *SWePservice* which is shown in Table 3.

```
<!-- //////////////////////////////////////////////////////////////////////////////////////////
      PARTNERLINKS
      List of services participating in this BPEL process
    //////////////////////////////////////////////////////////////////////////////////////////-->
<partnerLinks>
<!-- The 'client' role represents the requester of this service. It is
used for callback. The location and correlation information associated
with the client role are automatically set using WS-Addressing. -->
<partnerLink name="E_Governance_Portal" partnerLinkType="client:Web_Service" myRole="Web_ServiceProvider"/>
<partnerLink name="BPL" partnerLinkType="client:Web_Service"
partnerRole="Web_ServiceProvider"/>
<partnerLink name="IAY" partnerLinkType="client:Web_Service"
partnerRole="Web_ServiceProvider"/>
<partnerLink name="AAY" partnerLinkType="client:Web_Service"
partnerRole="Web_ServiceProvider"/>
<partnerLink name="NREGA" partnerLinkType="client:Web_Service"
partnerRole="Web_ServiceProvider"/>
<partnerLink name="RSBY" partnerLinkType="client:Web_Service"
partnerRole="Web_ServiceProvider"/>
<partnerLink name="KJY" partnerLinkType="client:Web_Service"
partnerRole="Web_ServiceProvider"/>
</partnerLinks>
```

Table 3. Partner Links of SWeP

Variables: These are used to store, reformat and transform messages. We generally need a variable for every message sent to the partners and receive back messages from them. Each variable has a specific type. The following code shown in Table 4demonstrates the variables used by *SWeP*BPEL process.

```
<!--
  ////////////////////////////////////////////////////////////////////////////////////////////////////
    VARIABLES
    List of messages and XML documents used within this BPEL process
  ////////////////////////////////////////////////////////////////////////////////////////////////////
 -->
<variables>
<!-- Reference to the message passed as input during initiation -->
<variable name="inputVariable" messageType="client:Web_ServiceRequestMessage"/>
<!-- Reference to the message that will be returned to the requester-->
<variable name="outputVariable" messageType="client:Web_ServiceResponseMessage"/>
</variables>
```

Table 4. Variables of SWeP

Process logic definition: In this we specify the order in which the partner Web services are invoked. For our case, we start with a <sequence>tag that defines several activities which will be performed sequentially. Within the that, we first specify the input message that starts the business process. We do that with the <receive> construct, which waits for the matching message. After that, we link the client partnerwith themessage reception. The process logic is shown in Table 5.

```
<!--  ////////////////////////////////////////////////////////////////////////////////////////////////////
ORCHESTRATION LOGIC
Set of activities coordinating the flow of messages across the
services integrated within this business process
////////////////////////////////////////////////////////////////////////////////////////////////////  -->
<sequence name="main">
<!-- Receive input from requestor. (Note: This maps to operation defined in Web_Service.wsdl) -->
<!-- Generate reply to synchronous request -->
<receive name="Receive_Input" createInstance="no"
partnerLink="E_Governance_Portal" portType="client:Web_Service"
operation="process"/>
<assign name="Assign_Input"/>
<invoke name="Invoke_BPL" bpelx:invokeAsDetail="no" partnerLink="BPL"
portType="client:Web_Service" operation="process"/>
<assign name="Assign_BPL_Output"/>
<invoke name="Invoke_IAY" bpelx:invokeAsDetail="no" partnerLink="IAY"
portType="client:Web_Service" operation="process"/>
<assign name="Assign_IAY_Output"/>
<invoke name="Invoke_KJY" bpelx:invokeAsDetail="no" partnerLink="KJY"
portType="client:Web_Service" operation="process"/>
<assign name="Assign_KJY_Output"/>
<reply name="Reply_output" partnerLink="E_Governance_Portal"
portType="client:Web_Service" operation="process"/>
</sequence>
</process>
```

Table 5. Orchestration Logic of SWeP

The remaining codes for other orchestration processes also follow the same pattern. The corresponding BPEL process diagram for this code is shown in Figure -7.

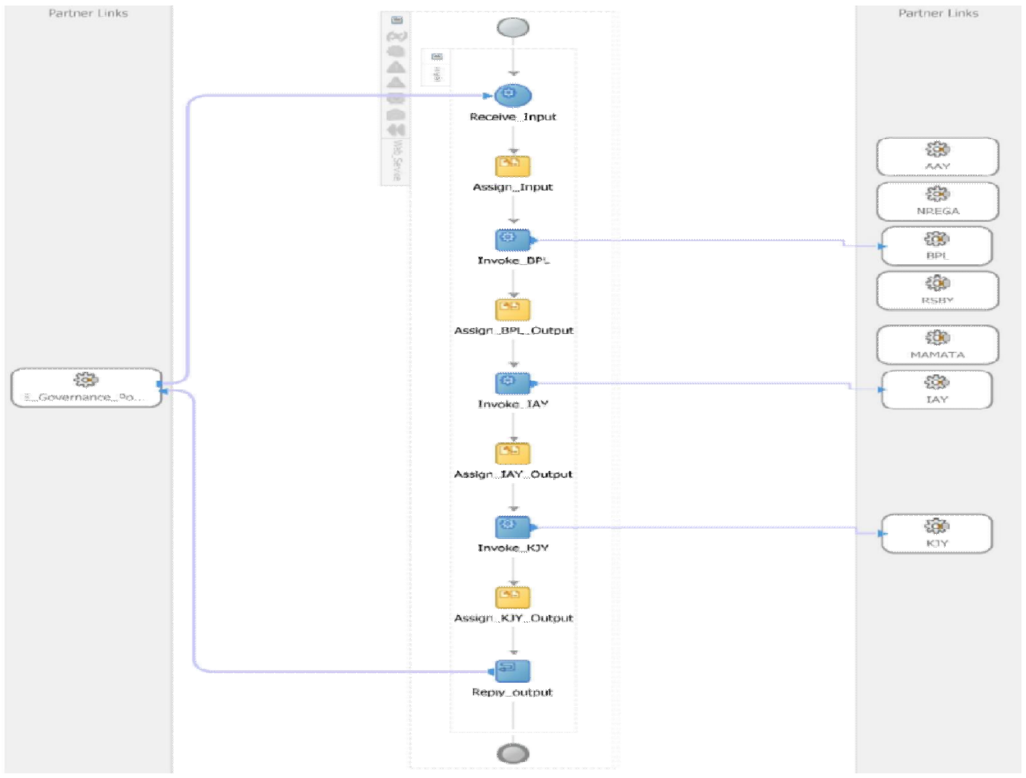


Figure 7. SWeP Service BPEL Diagram

BPEL Diagram Flow Details

The process starts when the client invokes E_Governance_Portal_ep interface. It is shown as partner link on the left side in the diagram. The remaining process proceeds as a sequence of steps which are described below.

1) First the input received is assigned to *assign input* and *invokes* BPL service
2) The *BPL* output is assigned to input variable and it invokes *IAY service*
3) The *IAY* output is assigned to output variable and it invokes *KJY service*
4) The KJY output is assigned to the reply output variable
5) The final output is assigned to a variable and is replied

8. SERVICE DEPLOYMENT

In this section we present an implementation prototype of SWeP. This prototype is implemented using PHP, MySQL, WSDL and SOAP messaging. Different user interfaces are designed for the prototype. A brief discussion on each service code and related output screen shots are presented below. The SWeP prototype interacts with different departments of Government of Odisha like Panchayat Raj, Health, Finance, Women and Child Development and Energy. The portal receives information from these departments as per users' requirements, irrespective of their underlying technology. Using the web services technology, different autonomous web services are developed for each department and some composite web services depending on different departments and based on some orchestration rules. The portal provides different Government to Government(G2G) services and some Government to Citizen(G2C) services. A new service can also be developed as and when required, by invoking the available services to meet the user requirements. In the following sections we provide some screen shots and sample implementation codes for better understanding.Figure 8 shows the screen shot of administrativelogin ofthe SWeP portal.

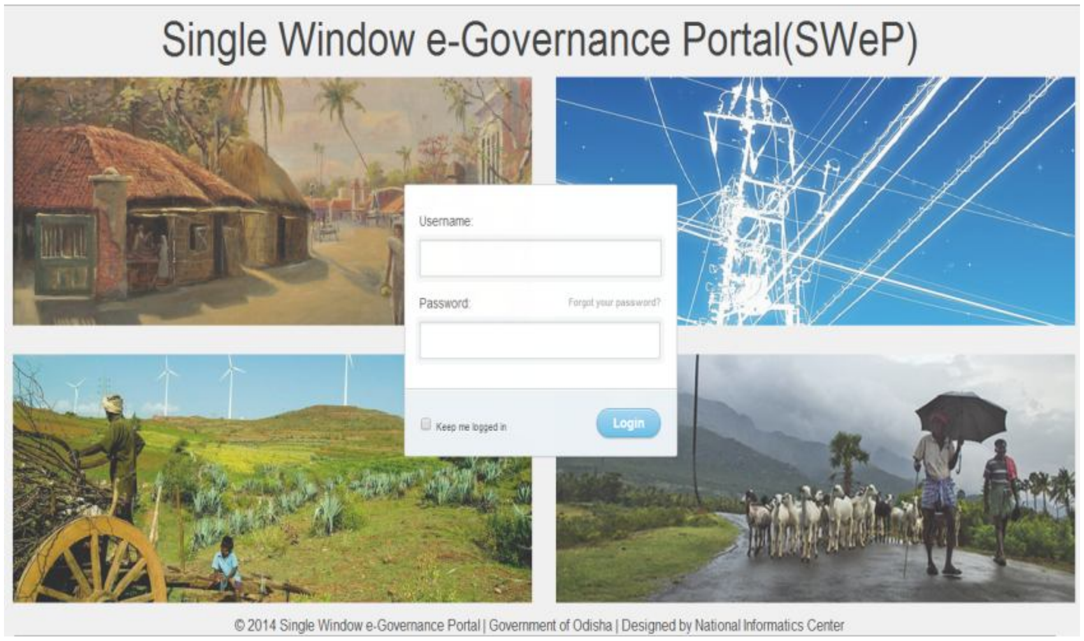
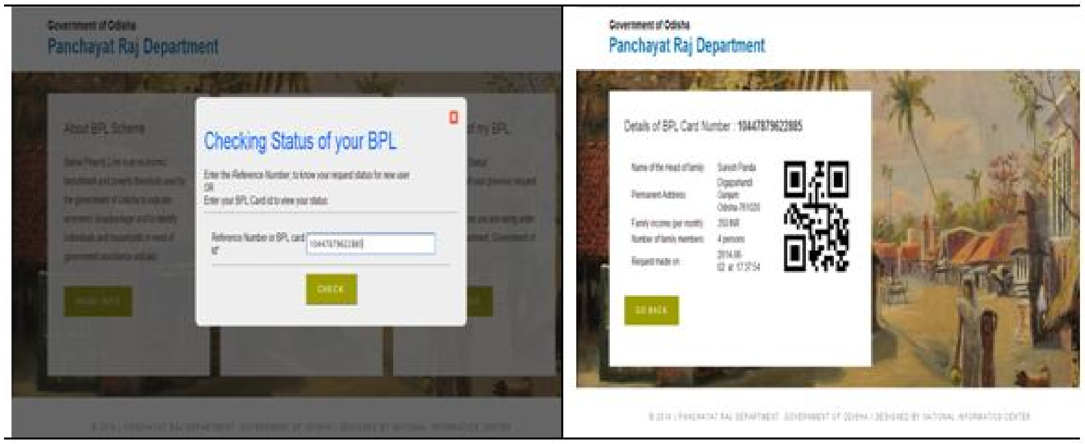

Single Window e-Governance Portal(SWeP)
Username:
Password:
Forgot your password?
Keep me logged in
Login
© 2014 Single Window e-Governance Portal | Government of Odisha | Designed by National Informatics Center

Figure 8. SWePUser Interface Snapshot

Figure-9 below shows the screen shot for retrieving the BPL status of any citizen through the BPL service.


Government of Odisha
Panchayat Raj Department
About BPL Scheme
Checking Status of your BPL
Enter the Reference Number, to know your request status for new user
OR
Enter your BPL Card id to view your status
Reference Number or BPL card Id
CHECK
Government of Odisha
Panchayat Raj Department
GO BACK

Details of BPL Card Number : 104478796223885

Name of the Head of family	Suresh Panda
Permanent Address	Digapahandi Ganjam Odisha-761020
Family income (per month)	350 INR
Number of family members	4 persons
Request made on	2014-06-02 at 17:37:54

Figure 9. Checking the BPL Service Status Snapshot

The following Table-6 shows the WSDL code for BPL Service.

```
<?xmlversion="1.0"encoding="UTF-8"?>
<wsdl:definitionsname="DeptPanchayat"xmlns:xsd="http://www.w3.org/2001/XMLSchema"targetNamespac
e="DeptPanchayat"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"xmlns:tns="DeptPanchayat"xmlns:wsdl="htt
p://schemas.xmlsoap.org/wsdl/">
          <xsd:documentation></xsd:documentation>
<!—types -->
          <wsdl:types>
<xsd:schemaxmlns:xsd="http://www.w3.org/2001/XMLSchema"targetNamespace="DeptPanchayat">
                              <xsd:complexTypename="AllBPL">

          <xsd:sequence><xsd:elementname="BPL_id"type="xsd:string"/><xsd:elementname="AllBPLStr"ty
pe="xsd:string"/></xsd:sequence></xsd:complexType>
                              <xsd:complexTypename="DetailsBPL">

          <xsd:sequence><xsd:elementname="BPL_id"type="xsd:string"/><xsd:elementname="BPLObj"type
="tns:BPLObj"/></xsd:sequence></xsd:complexType>
                    </xsd:schema>
          </wsdl:types>
<!-- message -->
          <wsdl:messagename="getAllBPLRequest"><wsdl:partname="BPL_id"type="xsd:string"/></wsdl:m
essage>
          <wsdl:messagename="getAllBPLResponse"><wsdl:partname="AllBPLStr"type="xsd:string"/></ws
dl:message>
          <wsdl:messagename="getDetailsBPLRequest"><wsdl:partname="BPL_id"type="xsd:string"/></ws
dl:message>
          <wsdl:messagename="getDetailsBPLResponse"><wsdl:partname="BPLObj"type="tns:BPLObj"/>
</wsdl:message>
<!--portType -->
          <wsdl:portTypename="DeptPanchayat">
          <wsdl:operationname="getAllBPL"><wsdl:inputmessage="xsd:getAllBPLRequest"/><wsdl:outputm
essage="xsd:getAllBPLResponse"/></wsdl:operation>

          <wsdl:operationname="getDetailsBPL"><wsdl:inputmessage="xsd:getDetailsBPLRequest"/><wsdl:
outputmessage="tns:getDetailsBPLResponse"/></wsdl:operation>  </wsdl:portType>
<!-- binding -->
          <wsdl:bindingname="DeptPanchayat"type="tns:DeptPanchayat">
                    <soap:bindingstyle="rpc"transport="http://schemas.xmlsoap.org/soap/http"/>
<!-- extensibility element for   operation details -->
          <wsdl:operationname="getAllBPL"><soap:operationsoapAction="http://localhost/dept_panchayat/b
pl_service.php"/>

          <wsdl:input><soap:bodyuse="literal"namespace="DeptPanchayat"/></wsdl:input>

          <wsdl:output><soap:bodyuse="literal"namespace="DeptPanchayat"/></wsdl:output><!--
extensibility element providing body details -->
                    </wsdl:operation>
          <wsdl:operationname="getDetailsBPL"><soap:operationsoapAction="http://localhost/dept_panchay
at/bpl_service.php"/>
                    <wsdl:input><soap:bodyuse="literal"namespace="DeptPanchayat"/></wsdl:input>
                    <wsdl:output><soap:bodyuse="literal"namespace="DeptPanchayat"/></wsdl:output><!--
extensibility element providing body details -->
                    </wsdl:operation>   </wsdl:binding>
<!—service    →
          <wsdl:servicename="DeptPanchayat">
                    <wsdl:portbinding="tns:DeptPanchayat"name="Panchayat">
                              <soap:addresslocation="http://localhost/dept_panchayat/bpl_service.php"/>
                    </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Table 6. The BPL Service WSDL Code

The above code is used to define the BPL service and to access it. The BPL service is invoked through a remote procedure call (RPC) model. Two messages are described, the input/request message, which is sent from the client to the service named "getAllBPL" and the output/response message, which is sent back to the service "getAllBPLResponse". The BPL service is invoked using the statement "<soap:address location="http://localhost/dept_panchayat/bpl_service.php" />".

In the above Code

- targetNamespace:- is the logical namespace for information about this service. As WSDL documents imports other WSDL documents, hence setting targetNamespace to a unique value ensurs that the namespaces do not clash.[3]
- xmlns: is the default namespace of the WSDL document, and is always set tohttp://schemas.xmlsoap.org/wsdl/
- All the WSDL elements, such as <definitions>, <types> and <message> reside in this namespace.
- xmlns:xsd and xmlns:soap are standard namespace definitions which are used for specifying SOAP-specific information as well as data types.[3]
- xmlns:tns stands for this namespace.

The types section gives information about any complex data types used in the WSDL document. The document does not need to have a types sectionwhen simple data types are used.In thesetype of messages an abstract data definition is communicated.

In the example, the message contains two parts, request, response, which are of type string, where string is defined by the XML schema.In the prototype an abstract set of operations supported by one or more endpoints widely known as an interface; operations are defined by exchange of messages.

The above example defines a portTypenamed DeptPanchayat that consists of two operations: getAllBPL and getDetailsBPL.The WSDL binding element describes details of using a particular portType with a given protocol. The binding element contains many extensibility elements and a WSDL operation element for each operation in the portType. It describes how the operation isiniatiated by identifying concrete protocol and data format specifications for the operations and messages. The soap:operation element defines the SOAPAction HTTP header value for each operation and the soap:body element defines how the message parts appear inside of the SOAP body element (possible values include literal or encoded).

At services, a collection of related endpoints, where an endpoint is defined as a combination of a binding and an address (URI).One must give each port a name and assign it a binding. Then, within the port element, an extensibility element is used to define the address details specific to the binding. For example, the following sample defines a service called DeptPanchayat that exposes the tns:DeptPanchayat at the http://localhost/dept_panchayat/bpl_service.php URL.

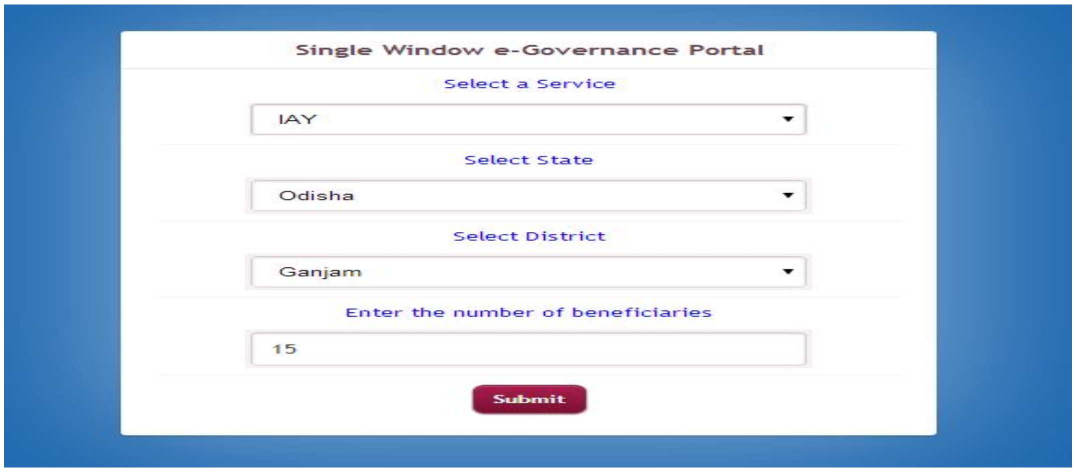
The following Table-7 shows the PHP code for BPL Service. The bpl_services.php code has two functions defined as "getdetailsBPL" and "getAllBPL". "getdetailsBPL" is used to retrieve the detail information about a single BPL family.The "getALLBPL" is used to retrieve the BPL holder's information under a District/Block/Panchayat/Village as required in G2G and G2C services.

```
<?php
classBPLClass{
public$HeadName;
public$AddrVill;
public$AddrDist;
public$AddrState;
public$AddrPIN;
public$FamilyIncome;
public$NumMember;
function
__construct($HeadName,$AddrVill,$AddrDist,$AddrState,$AddrPIN,$FamilyIncome,$NumMember
){
$this->HeadName=$HeadName;
$this->AddrVill=$AddrVill;
$this->AddrDist=$AddrDist;
$this->AddrState=$AddrState;
$this->AddrPIN=$AddrPIN;
$this->FamilyIncome=$FamilyIncome;
$this->NumMember=$NumMember;
}
}
include("functions.php");
$db=ConnectDB();
functiongetAllBPL(){
$sql="SELECT `BPL_id` FROM `bplholders` WHERE `BPL_id` NOT LIKE
'Ref_%'";$result=mysql_query($sql);
if(mysql_affected_rows()<=0)returnNULL;
$num_rows=mysql_affected_rows();
for($i=0,$BPL_list="";$i<$num_rows;$i++){
$details=mysql_fetch_array($result);$BPL_list=$BPL_list.", ".$details["BPL_id"];
}
return(trim($BPL_list,", "));
}
functiongetDetailsBPL($BPL_id){if(preg_match('/Ref_/',$BPL_id))returnNULL;
$sql="SELECT * FROM `BPLHolders` WHERE `BPL_id` =
'".$BPL_id."'";$result=mysql_query($sql);
if(mysql_affected_rows()<=0)returnNULL;
$details=mysql_fetch_array($result);
$BPLObj=newBPLClass($details["HeadName"],$details["AddrVill"],$details["AddrDist"],$details["
AddrState"],$details["AddrPIN"],$details["FamilyIncome"],$details["NumMember"]);
return$BPLObj;
}

$server=newSoapServer("dept_service.WSDL",array());
$server->addFunction("getDetailsBPL");
$server->addFunction("getAllBPL");
$server->handle();
CloseDB($db);
?>
```

Table -7 PHP Code for BPL Service

Figure-10 is a screen shot for selecting a particular number of IAY families from a desired district, which is generally used as a G2G service for providing IAY benefits to the new citizen as per the suggested list provided by the web service by selecting the most appropriate citizens based on different services already availed by them. The suggested output list of the query given in Figure-10 is shown below in Figure-11


Single Window e-Governance Portal

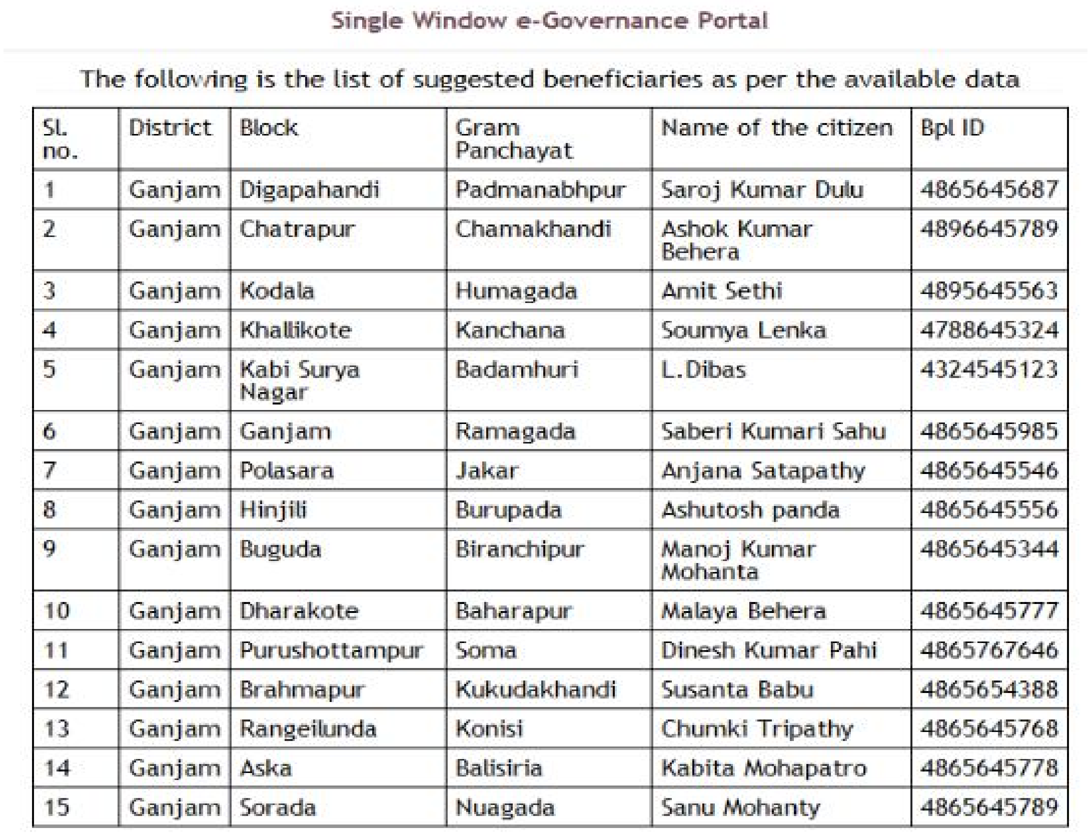
Select a Service: IAY

Select State: Odisha

Select District: Ganjam

Enter the number of beneficiaries: 15

Figure 10. Checking the BPL Service Status Snapshot

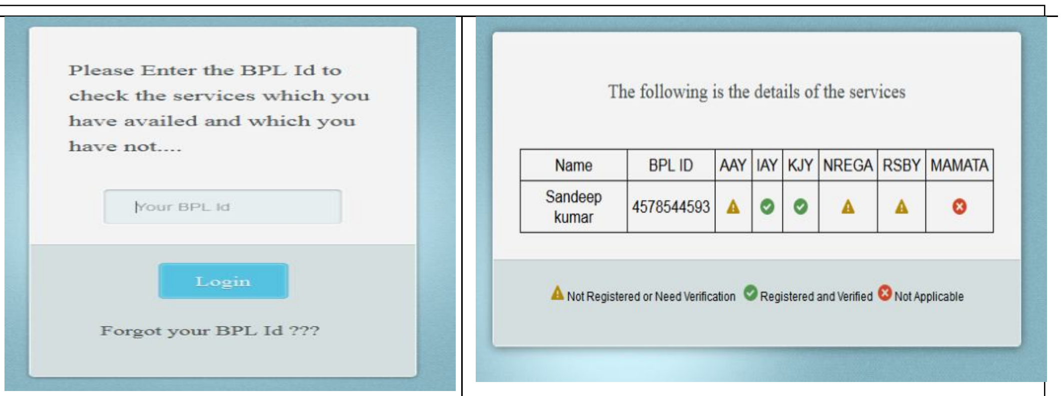

Single Window e-Governance Portal

The following is the list of suggested beneficiaries as per the available data

Sl. no.	District	Block	Gram Panchayat	Name of the citizen	Bpl ID
1	Ganjam	Digapahandi	Padmanabhpur	Saroj Kumar Dulu	4865645687
2	Ganjam	Chatrapur	Chamakhandi	Ashok Kumar Behera	4896645789
3	Ganjam	Kodala	Humagada	Amit Sethi	4895645563
4	Ganjam	Khallikote	Kanchana	Soumya Lenka	4788645324
5	Ganjam	Kabi Surya Nagar	Badamhuri	L.Dibas	4324545123
6	Ganjam	Ganjam	Ramagada	Saberi Kumari Sahu	4865645985
7	Ganjam	Polasara	Jakar	Anjana Satapathy	4865645546
8	Ganjam	Hinjili	Burupada	Ashutosh panda	4865645556
9	Ganjam	Buguda	Biranchipur	Manoj Kumar Mohanta	4865645344
10	Ganjam	Dharakote	Baharapur	Malaya Behera	4865645777
11	Ganjam	Purushottampur	Soma	Dinesh Kumar Pahi	4865767646
12	Ganjam	Brahmapur	Kukudakhandi	Susanta Babu	4865654388
13	Ganjam	Rangeilunda	Konisi	Chumki Tripathy	4865645768
14	Ganjam	Aska	Balisiria	Kabita Mohapatro	4865645778
15	Ganjam	Sorada	Nuagada	Sanu Mohanty	4865645789

Figure 11. Suggested List of Beneficiaries for IAY

Figure-12is a screen shot for finding different services already availed by a citizen, given his BPL_ID. This is a Government to Citizen(G2C) service which crawls all the six department databases through the corresponding web services to find out the status of all the services availed by the BPL citizen. This helps the citizen to apply for the rest of the eligible services which are marked as not-registered.


Please Enter the BPL Id to check the services which you have availed and which you have not....

Your BPL Id

Login

Forgot your BPL Id ???

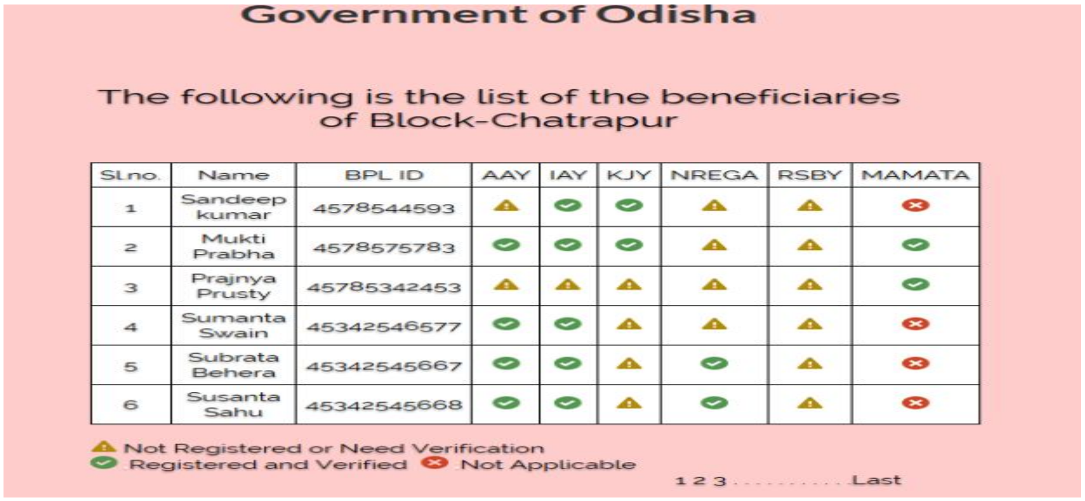
The following is the details of the services

Name	BPL ID	AAY	IAY	KJY	NREGA	RSBY	MAMATA
Sandeep kumar	4578544593	⚠	✓	✓	⚠	⚠	✗

⚠ Not Registered or Need Verification ✓ Registered and Verified ✗ Not Applicable

Figure 12. Snapshot for Checking the individual services availed status

Figure-13 is a screen shot for getting the list of beneficiaries of a desired district/block/panchayat with the list of different services already availed by them. This is a Government to Government(G2G) service, which helps the administration to provide new services to the most eligible person of that area, as per the suggested list generated by the web service. To get this list, it crawls all the six department databases through the corresponding web services to find out the status of all the services availed by the BPL citizens of that area with name and BPL_ID.


Government of Odisha

The following is the list of the beneficiaries of Block-Chatrapur

Sl.no.	Name	BPL ID	AAY	IAY	KJY	NREGA	RSBY	MAMATA
1	Sandeep kumar	4578544593	⚠	✓	✓	⚠	⚠	✗
2	Mukti Prabha	4578575783	✓	✓	✓	⚠	⚠	✓
3	Prajnya Prusty	45785342453	⚠	⚠	⚠	⚠	⚠	✓
4	Sumanta Swain	45342546577	✓	✓	⚠	⚠	⚠	✗
5	Subrata Behera	45342545667	✓	✓	⚠	✓	⚠	✗
6	Susanta Sahu	45342545668	✓	✓	⚠	✓	⚠	✗

⚠ Not Registered or Need Verification
✓ Registered and Verified ✗ Not Applicable

1 2 3 . . . . Last

Figure 13. List of beneficiaries with status of different services

Due to space constraint we have not included details of other services like IAY, AAY, Mamata, NREGA, RSBY and KJYin this paper.

9. SERVICE MONITORING

The ability to set up a Service Based System (SBS) monitoring framework has been increasingly recognized as one of the essential preconditions for deployment of an e-Governancesystem. This is because the government organizations involved in the e-Governanceservice are running independently. Such organization's services are developed and managed autonomously and can change without notification leading to run-time problems. As and when such deviations are detected those must be captured and analysed so as to take appropriate action. For instance a citizen to get KJY Service, the process involves Panchayat Raj Department, Energy Department and the designated bank for payment. The citizen may want to know why the process got delayed. The occurrence of significant delay of this process has to be reported as soon as possible so that the responsible authority can take prompt action.A monitoring framework has been developed [2] which is independent of any business logic and service composition platform. The monitoring engine works in parallel with the SBS and allows for easy adjustment of the business process. The SBS sends interesting events from the business layer, service layer as well as from the infrastructure layer and feeds those into an event bus at run-time. The service monitor observes the events from the event bus and accordingly monitors the functional and non-functional service composition assumptions and requirements of the SBS. We have also designed a monitor generator which automatically generates a program for the monitor which is deployed at run-time, thus reducing the design and implementation efforts.

10. CONCLUSIONS

Developing e-Governance applications has been a daunting task because of the inherent complexity, dynamic requirements and need for interoperability. The service oriented approach has emerged as a viable alternative to address the intriguing issues in the implementation of e-Governance systems. In this work, we have defined a Service Oriented e-Governance (SOeGov) approach and discussed each step in detail. We have also described the functionality of each of the service layers, namely, Citizen Access layer, Service Interface Layer, Application Layer, Shared Services Layer and Data Layer.The proposed layered approach can help in building an effective service delivery infrastructure for any e-Governance system.

REFERENCES

[1] Das, R. K. and Patra, M. R. (2013). "A Service Oriented Design Approach for e-Governance Systems". In International Journal of Information Technology Convergence and Services (IJITCS), Vol.3, No. 3, pp. 1-11.

[2] Parta, M. R. and Tripathy, A. K., Das, R. K. (2011). "Monitoring of service based e-governance systems", 5th International Conference on Theory and Practice of Electronic Governance ICEGOV 2011: September 26–28, 2011, Tallinn, Estonia. ACM Publication, pp. 353-354. doi: 10.1145/2072069.2072138

[3] Oracle *JDeveloper* - Official Home Page, available at http://www.oracle.com/ technetwork/developer-tools/jdev/overview/index.html, last accessed on 14/05/2014.

[4] Palvia, S. C. J. and Sharma, S. S. (2007). "E-Government and E-Governance: Definitions/Domain Framework and Status around the World". Foundations of E-Government. ICEG 5th International Conference on E-Governance.

[5] The Oracle SOA Suite Downloads links, available at http://www.oracle.com/ technetwork/middleware/soasuite/downloads/index.html?ssSourceSiteId=opn, last accessed on 14/05/2014.

Authors

Rama Krushna Das is Technical Director (Scientist-E) working with National Informatics Centre (NIC), Department of Electronics and Information Technology, Government of India. His research and professional career spans over twenty six years of coding, research and capacity building in computing, e-governance and related subjects. His expertise is primarily in the domains of Electronic Governance, Implementation Architectures and Strategy, and Software Technology. He is presently involved in development and implementation of different E-Governance projects of NIC. He has published several peer-reviewed papers as journal articles, book chapters, and contributions to conference proceedings. His research interests are e-governance, software engineering, Service Oriented Architecture and Cloud computing. He is a life member of Computer Society of India (CSI) and a professional member of the Association for Computing Machinery (ACM).

Dr. Manas Ranjan Patraholds a Ph.D.degree in Computer Science from the Central University of Hyderabad and hasbeen teaching in the Department of Computer Science, Berhampur University for more than 25 years. Currently he is also the Director of the University Computer Centre. He has worked in the International Institute for Software Technology, Macau as a United Nations visiting Fellow in the year 2000. He is actively engaged in teaching and research in different areas of computer science. His research interests include Intelligent agents, Service Oriented and Cloud based Computing, applications of data mining and Electronic Governance**.** He has more than 100 publications in International journals and conferences to his credit. He has extensively travelled abroad for presenting research papers in international conferences held in U.S.A., Australia, Europe, China, Korea, Bangkok and Egypt. He has served as a member in the Review and Programme committees of various International journals and conferences.

SujataPatnaikis continuing her Master of Computer Applicationsin Berhampur University, India. She is a gold medallistin B.Sc. Computer Science(Honours) in the year 2012 from Berhampur University. Her research interests include Service Oriented Architecture and Cloud Computing.