Journal of ELECTRICAL ENGINEERING, VOL. 62, NO. 4, 2011, 227–232

SERVICE ORIENTED ARCHITECTURAL MODEL FOR LOAD FLOW ANALYSIS IN POWER SYSTEMS

Balasingh Moses MUTHU * — **Ramachandran VEILUMUTHU** ** — **Lakshmi PONNUSAMY** ***

The main objective of this paper is to develop the Service Oriented Architectural (SOA) Model for representation of power systems, especially of computing load flow analysis of large interconnected power systems. The proposed SOA model has three elements namely load flow service provider, power systems registry and client. The exchange of data using XML makes the power system services standardized and adaptable. The load flow service is provided by the service provider, which is published in power systems registry for enabling universal visibility and access to the service. The message oriented style of SOA using Simple Object Access Protocol (SOAP) makes the service provider and the power systems client to exist in a loosely coupled environment. This proposed model, portraits the load flow services as Web services in service oriented environment. To suit the power system industry needs, it easily integrates with the Web applications which enables faster power system operations.

K e y w o r d s: power systems, XML, WSDL, SOAP, SOA

1 INTRODUCTION

The large interconnected power systems require a common computational environment for interaction between the client and the service provider. The power system needs a communication among the service provider and client in a heterogeneous environment. Since there is a tremendous growth in power systems, it needs an architecture that allows plugging in new services or upgrading existing services in a granular fashion to address the new requirements. The power systems require the services which are available in the universal visibility to be accessed by the clients. At real time operations, an enormous quantum of data has to be exchanged among large interconnected power systems. The data must be exchanged in a reliable manner. The power flow services are essential at the time of planning and operations to carry out any type of power system applications by the power system engineer.

To monitor the power flow in the system, the Supervisory Control and Data Acquisition Systems (SCADA) are developed based on different platforms through Remote Terminal Units (RTU) linked by communication channel. Qui and Gooi developed a Web-Based SCADA display systems for monitoring load flow to solve the legacy issues using Java Native Interface (JNI), which is really a tedious process [1]. Chen and Liu demonstrated the potential advantages of the Web as the platform for developing and deploying complex power system simulations by using the concept of distributed technologies [2]. The RMI based load flow monitoring developed by Nithiyananthan and Ramachandran enables the neighbouring power systems to access the remote load flow server by the power system clients [3]. Molcolm et.al modelled a complete Web based and platform independent power system simulation package with various analyses in a distributed and clustered environment [4]. The technologies like SCADA, Common Object Request Broker Architecture (CORBA), Remote Procedure Calls (RPCs) and Remote Method Invocation (RMI) communication do not guarantee the exploitation of the full functionality among heterogeneous environment [6]. To overcome the complexity and proprietary nature of the above mentioned technologies, the SOA model has been developed for representation of power systems which provides the powerful set of features based on open standards like XML, SOAP, WSDL and UDDI. The model supports interoperability between services on different platforms in a reliable, scalable and adaptable manner.

2 XML REPRESENTATION OF LOAD FLOW DATA

As the size of power systems is large, data exchange is a major concern because of the complex nature of power system operations. New participants join the power industries as generation companies, transmission companies, and distributors. Because of the physical connectivity of power systems, all levels of industry like generation, transmission and distribution need proper operational data. The exchange of data needs to be reliable, error-free and adaptable to different types of software used for power system operations in the related power system industries. In power systems, data can be output of a process while being input for another. In real time

Department of Electrical and Electronics Engineering, Anna University, No 109, Academic Block, Tiruchirappalli 620 024, India, balasinghmoses@gmail.com

DOI: 10.2478/v10187-011-0036-9, ISSN 1335-3632

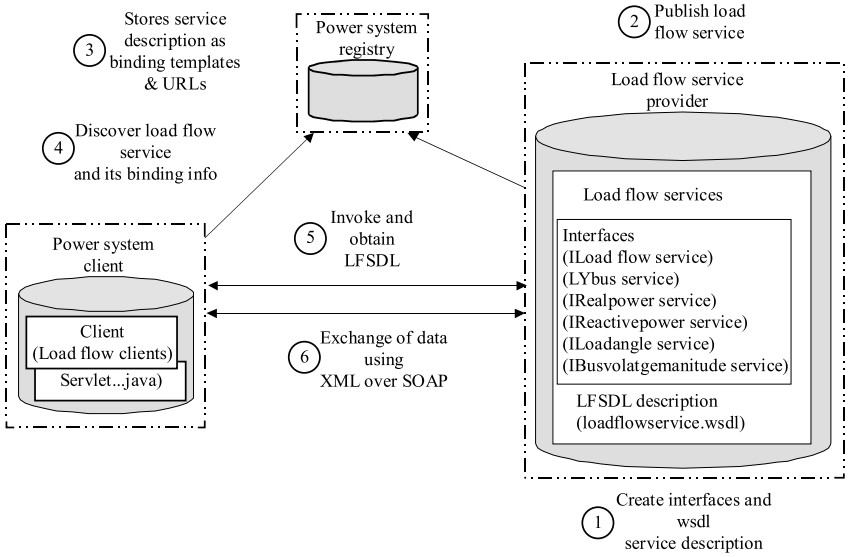

Fig. 1. Proposed SOA model for Load Flow Analysis

analysis, huge amount of data are being exchanged among interconnected systems. IEEE common format is utilized while representing the power system data in XML. The data exchange must have a protocol that makes the data meaningful for each power system operation. The exchange of power system data using eXtensible Markup Language (XML) offers trouble-free integration with the Web and Intranet / Internet applications.

The power systems bus data store all specified bus values and the violation of their operating limits. The power systems bus data required for load flow analysis is represented as follows

```
<?xml version="1.0" encoding="UTF-8"?>
<GENERAL>
    <Base MVA>Base value of the system</Base MVA>
   <NB> Number of Buses </NB>
   <TOLERANCE> Tolerance value</TOLERANCE>
    <IDENTIFICATION>Power Systems Bus Data
    </IDENTIFICATION>
 </GENERAL>
 <SYSTEM_ BUS_DATA>
      <SLACK_BUS>
        <VOLTAGE> voltage at the bus</VOLTAGE>
        <ANGLE>Load angle of the Bus </ANGLE>
      </SLACK_BUS>
      <GENERATOR_BUS>
          <MW> Real Power </MW>
          <VOLTAGE>Generating Voltage</VOLTAGE>
      </GENERATOR_BUS>
      <LOAD_BUS>
          <MW> Real Power </MW>
          <MVAR>Reactive Power </MVAR>
      </LOAD_BUS>
      <LIMITS unit='MVAR'>
          <MAX>Maximum Reactive Power Limit</MAX>
          <MIN>Minimum Reactive Power Limit </MIN>
    </LIMITS>
</SYSTEM_BUS_DATA>
```

SYSTEM_BUS_DATA is the root element. The child elements are Slack bus, Generator bus and Load bus. The power systems transmission line data store all the line values and the capability of the line to withstand the maximum loading. The power systems line data required for load flow analysis is represented as follows

```
<?xml version="1.0" encoding="UTF-8"?>
<GENERAL>
      <NL> Number of lines </NL>
      <IDENTIFICATION>Power Systems Line Data
       </IDENTIFICATION>
</GENERAL>
<SYSTEM_LINE_DATA >
    <SENDING_BUS> Starting line </SENDING_BUS>
    <RECEIVING_BUS>Ending line</RECEIVING_BUS>
      <R> Resistance of the line> </R>
      <X> Reactance of the line> </X>
      <Loading MVA> Maximum loading of the line
                                    </Loading MVA>
      <Transformer> Rating </Transformer>
      <Shunt_Capacitor>Rating </Shunt_Capacitor>
 </SYSTEM_LINE_DATA>
```

The XML document created can be edited online by the use of Document Object Model (DOM). With the help of XML style sheet language (XSL), XML document can be converted to any other form such as html, PDF, postscript, even XML document. But, often a name conflict may occur when two different documents use the same name describing two different types of elements. To overcome this difficulty, XML Namespaces are used. XML elements are used with prefixes, which are mapped to a URL (Uniform Resource Locator) that usually correspond to an Internet resource, usually the IP addresses.

3 PROPOSED SOA MODEL FOR LOAD FLOW ANALYSIS

Service Oriented Architectural model for representation of load flow services is shown in Figure1. The SOA model has three elements namely Load Flow Service Provider, Power Systems Registry and Power Systems Client. The power system load flow services are

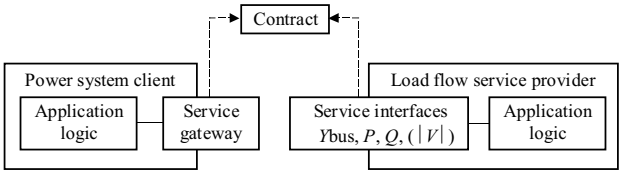

Fig. 2. Load Flow Service contract

categorized in to Bus admittance matrix (Y bus), Real power (P), Reactive power (Q), load angle (δ) and Bus voltage magnitude ($|V|$) services. A Load Flow Service provider offers the above services and describes the interface information of the services in interface description language called LFSDL (Load Flow Services Description Language) which is in the form of XML that makes the services available in the Power System Registry.

Services are the key building blocks of SOA. A service is a reusable function that can be invoked by another component through a well-defined interface. Services are loosely coupled, that is, they hide their implementation details and only expose their interfaces. In this manner, power system client need not be aware of any underlying technology or programming language which the service is using. The loose coupling between services allows for a quicker response to changes than the existing conventional applications for power system operations. This results in a much faster adoption to the need of power system industry.

The power system clients discover the service available in the registry by service names and acquire the interface information by LFSDL of the load flow services. Based on this information, the clients have a binding with the load flow service provider and can invoke services using Simple Object Access Protocol (SOAP).

4 IMPLEMENTATION OF PROPOSED SOA MODEL

The services included in the load flow service provider are the formation of Bus admittance matrix (Ybus), estimation of Real power (P), Reactive power (Q), load angle (δ) and Bus voltage magnitude ($|V|$). In the large interconnected systems, the client can invoke any of the above services which are required at the time of power system operations through well-defined interfaces. The various stages involved in the implementation of proposed SOA model for Load Flow Analysis are service interfaces, service description, service configuration, service mapping, service publishing, service discovering and service invoking. The implementation details of these stages are explained in the following sections.

4.1 Load Flow Service interfaces

The service interface implements the contract between the power system client and load flow service provider as shown in Figure 2. This contract allows them to exchange information between clients and service providers. The service interface is responsible for all of the implementation details needed to perform the communication between the clients and service provider. In this proposed model, five interfaces are implemented for the formation of Bus admittance matrix, estimation of Real power, Reactive power, load angle and Bus voltage magnitude.

Decoupling the service interface code from the service implementation code enables the system to deploy the two code bases on separate tiers, potentially increasing the deployment flexibility.

The service interface for computing Y bus is as follows

```
package loadflow;
public interface ybus extends Remote
  {
  public String computeybus() throws Remote Exception;
  }
```

4.2 Describing the Load Flow Services

LFSDL is used as the metadata language for defining the load flow services. It describes how the service providers and client communicate with each other. LFSDL is capable of describing services that are implemented using any language and deployed on any platform. It represents information about the interface and semantics of how to invoke a service. It contains the information about the data type, binding and address information for invoking the services from the service provider.

The Ybus service is descried as follows

```
<definitions name="loadflowservice"
        targetNamespace="urn:LoadFlow" >
<types>
   <schema targetNamespace="urn:LoadFlow"
    xmlns:tns="urn:LoadFlow"xmlns: soap11-
  enc="http://schemas.xmlsoap.org/soap/encoding/"  >
  <complexType name="computeybus" >
</complexType>
<element name ="computeybus"
        type ="tns:computeybus"/>
<element name ="computeybusResponse"
        type ="string"/>
</types>
<message name ="ybus_computeybus" >
<portType name ="ybus" >
<operation name="computeybus" >
    <input message="tns:ybus_computeybus"/>
    <output message
          ="tns:ybus_computeybusResponse"/>
</operation>
</portType>
<binding name="ybusBinding" type="tns:ybus" >
    <soap:binding transport= "http://schemas.
     xmlsoap.org/soap/http" style="document"/>
    <operation name="computeybus" >
      <soap:operation soapAction=""/>
      <soap:body use="literal"/>
      </output>
    </operation>
</binding>
<service name="Loadflowservice" >
<port name="ybusPort" binding="tns:ybusBinding" >
  <soap:address
```

```
        location="REPLACE_WITH_ACTUAL_URL"/>
      <soap:address
        location="REPLACE_WITH_ACTUAL_URL"/>
    </port>
</service>
</definitions>
```

The LFSDL definition document consists of seven key structural elements for describing load flow service. The <definition> element defines the name of the service as 'loadflowservice' and declares the namespace as 'LoadFlow'. The <types> element defines the data types that would be used to describe the Bus and Line data. The <message> element represents a logical definition of the data being transmitted between the client and the service provider .The <portType> element defines the abstract definition of the operation (computeybus) of the service, request and response messages. The <binding> element specifies a concrete protocol (SOAP) used for representing messages. The <service> element represents the service to be invoked over multiple bindings. The <port> element specifies an address (ybusport) for binding to the service.

4.3 Configuring the Load Flow Services

The services related to computing of load flow solutions are configured as follows

```
<service name="loadflowservice"
targetNamespace="urn:LoadFlow"
typeNamespace="urn:LoadFlow"
packageName="loadflow" >
¡interface name="loadflow.ybus"/>
¡interface name="loadflow.realpower"/>
¡interface name="loadflow.reactivepower"/>
¡interface name="loadflow.loadangle"/>
¡interface name="loadflow.busvoltagemagnitude"/>
</service>
```

This configuration file contains the information and details about the deployed load flow services (ybus, realpower, reactivpower, loadangle, busvoltagemagnitude) and metadata such as their service name (loadflowservice), namespace(LoadFlow) and description.

4.4 Mapping the Services

The mapping information is specified by an XML document. This document describes how the properties of the Ybus object have to be translated and mapped into XML. The mapping information for Ybus object is as follows

```
<package-mapping>
    <package-type>loadflow</package-type>
    <namespaceURI>urn:LoadFlow</namespaceURI>
</package-mapping>
<java-xml-type-mapping>
        <java-type>
          loadflow.ybus_computeybus_RequestStruct
        </java-type>
      <root-type-qname xmlns:typeNS="urn:LoadFlow">
            typeNS:computeybus
      </root-type-qname>
      <qname-scope>complexType</qname-scope>
</java-xml-type-mapping>
<port-mapping>
      <port-name>ybusPort</port-name>
      <java-port-name>ybusPort</java-port-name>
</port-mapping>
<service-endpoint-interface>loadflow.ybus
              </service-endpoint-interface>
  <wsdl-port-type xmlns:portTypeNS="urn:LoadFlow">
          portTypeNS:ybus</wsdl-port-type>
  <wsdl-binding xmlns:bindingNS="urn:LoadFlow">
        bindingNS:ybusBinding</wsdl-binding>
  <service-endpoint-method-mapping>
<java-method-name>computeloadangle
</java-method-name>
</service-endpoint-method-mapping>
```

The mapping file describes the elements like package, type, port, method, and endpoint and they have to be mapped into XML. The mapping information for the other services are described similar manner.

4.5 Publishing the Services in Registry

The proposed SOA model for Load Flow Analysis requires the common registry to deploy the services for easy integration, reuse and effective governance of services to meet the growing requirements. The registry allows the power system client to efficiently discover and communicate with the services. The main purpose of the registry is to allow fast and reliable communication and interoperability among diverse applications.

4.6 Discover the Load Flow Services

At the time of operation, the power system clients discover the required service which is available in the Registry. Once the service is discovered, the client will query the services by their names to get the binding information and the identification of the provider. Based on this information, the clients access the provider for invoking the services.

The procedure to discover the services is as follows

```
RegistryService rs = connection.getRegistryService();
BusinessQueryManager qm =
rs.getBusinessQueryManager();
BulkResponse response1 =
qm.getRegistryObjects("Service");
Collection objects = response1.getCollection();
Collection c=object.getServiceBindings();
ServiceBinding sb=(ServiceBinding)oi.next();
String uri=sb.getAccessURI();
```

The procedure involves the establishment of connection to the registry, querying the load flow service, getting the binding information and accessing URI to invoke a service.

4.7 Binding and Invoking the Services

The power system clients communicate with the load flow service provider using SOAP message as shown in Figure 3. The XML form of Bus and Line data are attached as input parameter in the SOAP body. The SOAP Body represents the mandatory processing information

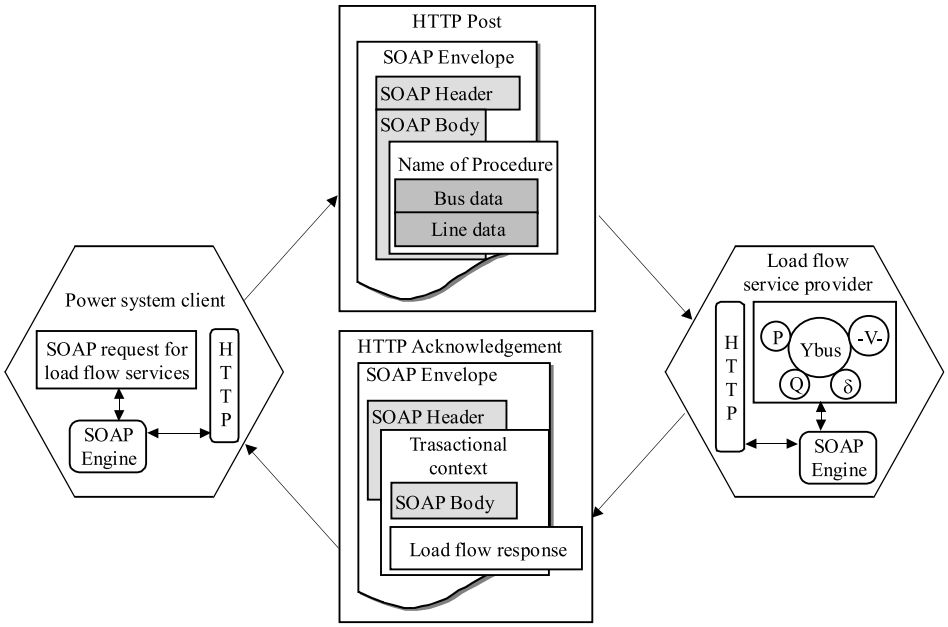

Fig. 3. SOAP communications (Client - Provider)

between the client and service provider. SOAP uses the HTTP POST for request and response in the form of messages.

SOAP messages are configured for load flow service invocation and response. The request message consists of IP address of the service provider and the required power system data in XML form. The following cod segment delineates how bus and line data have been attached to the SOAP message.

```
String urn = "urn:LoadFlow";
MessageFactory messageFactory=
                    MessageFactory.newInstance();
SOAPMessage message =
                  messageFactory.createMessage();
SOAPPart soapPart = message.getSOAPPart();
SOAPEnvelope envelope = soapPart.getEnvelope();
SOAPBody body = envelope.getBody();
SOAPElement bodyElement = body.addChildElement
          (envelope.createName(operation, "", urn));
FileInputStream fs;
fs=new leInputStream("\busdata.xml");
message.addAttachmentPart(attachment);
fs=new FileInputStream("c:\\linedata.xml");
message.addAttachmentPart(attachment1);
```

A binding is described how the SOAP request and response messages have been sent through the transport protocol. The following code segment delineates how the services are being invoked

```
SOAPConnection connection =
                     soapConnFactory.createConnection();
SOAPMessage reply = connection.call(message,destination);
reply.writeTo(output1);
SOAPPart soapPart1 = reply.getSOAPPart();
SOAPEnvelope envelope1 = soapPart1.getEnvelope();
SOAPBody body1 = envelope1.getBody();
Iterator iter = body1.getChildElements();
Node resultOuter = ((Node) iter.next()).getFirstChild();
Node result = resultOuter.getFirstChild();
connection.close();
```

Thus, establishing the SOAP connection to the destination as specified in the SOAP message, the power systems client is capable of invoking the required load flow service as given in the SOAP message at the specified port. The Bus and Line data content in the SOAP body are provided to the service. The provider sends an entire document rather than sending a set of parameters to the clients. The SOAP based communication model defines a loosely coupled and document-driven communication. Hence, the proposed SOA model for load flow analysis makes the service provider and the power systems client to exist in a loosely coupled environment.

5 RESULTS

The SOA model has been implemented in Windows NT based XP workstations connected in an Ethernet LAN. The implementation carried out in three different PC's and the specifications are shown in Table 1.

Table 1.

Name of the PC	CPU	Memory	IP Address
Load Flow Service Provider	Intel core 2 duo 2.6 GHz	2 GB	10.1.105.228
Power system Registry	Intel core 2 duo 2.0 GHz	2 GB	10.1.105.226
Power System client	Intel core 2 duo 2.0 GHz	2 GB	10.1.105.227

The SOAP request message contains the line and bus data for load flow analysis is shown below.

```
Part_0_13271847.1268718003027
Content-Type: text/xml; charset=utf-8
<SOAP-ENV:Envelope xmlns:—->
  <SOAP-ENV:Header/>
    <SOAP-ENV:Body>
      <compute xmlns="urn:LoadFlow"/>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
Part_0_13271847.1268718003027
Content-Type: text/plain Content-ID: data1
<linedata>
      <nb>8</nb>
      <nl>13</nl>
      <tol>0.0001</tol>
      <list> <sb>1</sb>
      <eb>2</eb>
      <lre>0.02</lre>
      <lim>0.06</lim> </list>
      ------
</linedata>
Part_0_13271847.1268718003027
Content-Type: text/plain
Content-ID: data2
<busdata>
      <vsp>1.06</vsp>
      <psp>0.7</psp>
      <qsp>-0.15</qsp>
      <qmin>0.0</qmin>
      <qmax>2.5</qmax>
      ------
</busdata>
```

The SOAP response for the load flow solutions using proposed SOA model is given below.

```
Part_0_13271847.1268718003027- -
<env:Envelope xmlns:env="http://schemas.xmlsoap.org
/soap/envelope/" xmlns:enc="http://schemas.xmlsoap.
org/soap/encoding/"xmlns:ns0="urn:LoadFlow"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<env:Body>
<ns0:computeResponse>
<result>
<iterations>13</iterations>
<vlist><v>1.0600</v><v>1.1000</v>——</vlist>
<deltalist><delta>-0.9554</delta><delta>-
          1.0054</delta>——</deltalist>
<ploss>0.0454</ploss><qloss>-0.4811</qloss>
</result>
</ns0:computeResponse>
</env:Body>
</env:Envelope>
```

The above SOAP response for load flow solutions can be parsed partially or totally to provide required data and the SOAP request will be generated dynamically to initiate other power system operations.

6 CONCLUSION

An effective Service Oriented Architectural model has been developed for representing load flow analysis of a power system and tested for a sample of 8, 14 and 30 bus systems. This model has the ability to analyze the load flow that may arise in the power system network by increasing the generation, transmission and distribution. Any number of power system clients can be served with out any limitation in this service oriented environment. The various power system services can be plugged into this model and the services are made available any time and any where for the power system operations.

References

[1] QUI, B.—GOOI, H. B.: Web Based SCADA Display System for Access via Internet, IEEE Transactions on Power System **15** No. 2 (2000).

[2] CHEN, S.—LIU, F. Y.: Web Based Simulations of Power Systems, IEEE Transactions on Computer Applications in Power **15** No. 1 (2002), 35–40.

[3] NITHIYANATAN, K.—RAMACHANDRAN, V.: RMI Based Multi-Area Power System Load Flow Monitoring, IJECE **3** No. 1 (2004), 28–30.

[4] IRVING, M,—TAYLOR, GARETH—HOBSON, P.: Plug in to Grid Computing, IEEE Trans. on Power and Energy Magazine **2** No. 2 (2004), 40–44.

[5] HASAN DAG—UMAT UTKAN: An XML based Data Exchange Model for power System Studies, ARI **54** No. 2 (2004).

[6] SHEKHAR, M.—KELAPURE, S. S. K.—SASTRY, A.—RAO, J. G.: Application of Web Services in SCADA Systems, International Journal of Emerging Electric Power Systems **6** No. 1 (2006).

[7] COMER, E.: Understanding Web Services: XML, WSDL, SOAP, and UDDI, Addison-Wesley, 2002.

[8] SKOCZYLAS, R.—NAGAPPAN, R.: Developing Java Web Services, Willey, 2003.

[9] DOUGLAS, K. BARRY: Web Service and Service Oriented Architecture, Morgan Kaufmann Publisher, 2003.

[10] http://www.service-architecture.com.

Received 5 May 2010

Balasingh Moses Muthu is currently working as a Lecturer in the Department of Electrical and Electronics Engineering, Anna University Tiruchirappalli, INDIA. He had received his Bachelor of Engineering (Electrical and Electronics Engineering) from Bharathidasan University, Tiruchirappalli and Master of Engineering (Power System Engineering) from Anna University, Chennai, INDIA in 1997 and 2004 respectively. He is currently doing Ph.D in Anna University Chennai, INDIA. His research interests include Power System Studies, Distributed Computing and Web Services.

Ramachandran Veilumuthu is currently working as a Professor of Computer Science and Engineering in College of Engineering, Guindy, Anna University, Chenai, INDIA. He has received his Masters of Engineering and PhD in Electrical Engineering from College of Engineering, Guindy, Anna University, Chennai, INDIA. His research interests include Power Systems Reliability Engineering, Network Security, Component Technologies, Soft Computing and Web Services.

Lakshmi Ponnusamy is currently working as a Assistant Professor of Electrical and Electronics Engineering in College of Engineering, Guindy, Anna University, Chenai, INDIA. She has received her Masters of Engineering and PhD in Electrical Engineering from College of Engineering, Guindy, Anna University, Chennai, INDIA. Her research interests include Power Systems Stability, Power Quality and Intelligent Controllers.