Journal of Applied Sciences Research, 6(2): 97-106, 2010

Software Framework Design for Power Disitribution System Applications with Struts

[1]Thiruvenkadam, S, [2]Dr. Nirmalkumar, [3]Sathishkumar, M.,

[1]Department of Electrical and Electronics Engineering,
Dr. Mahalingam College of Engineering and Technology, Pollachi, Tamilnadu, India.
[2]Department of Electrical and Electronics, Bannari Amman Institute of Technology,
Sathy, Tamilnadu, India.
[3]Department of Electrical and Electronics, Vellalar College of Engineering & Technology, Erode,
Tamilnadu, India.

Abstract: This paper presents architectural design concepts of a struts framework for distribution feeder application. In this paper, software framework has been developed which proposes in finding a solution for distribution feeder reconfiguration problem. The new reconfigured network obtained through heuristic fuzzy based algorithm, embedded in the software framework developed, satisfies the objectives like line loss reduction, load balance at the feeder level, low computation cost suitable for online application and software reusability. The proposed struts software framework that works based on layered MVC (Model View Controller) architecture with strict top-down dependency reduces software couplings. The effectiveness of the proposed software framework is successfully demonstrated by employing the feeder switching operation scheme to a standard distribution system indicated from the results and at the same time the developed framework can have the software reusability.

Key words: struts framework, software reusability, MVC architecture, optimization, Feeder reconfiguration.

INTRODUCTION

In the past several decades many software packages have been developed for electric power systems. Object-oriented with pure top-down dependency has improved development of power system analysis[1-4]. All these works developed software approaches of software design and development for standard power system analysis algorithms. In the design area there were many repetitive works in software development. Design part provides software design reuse[1] and[4-6].

The work addressed[1] on many small, individual design modules since patterns usually target specific design problems. The developments of reusable software for custom applications are discussed in[7-8]. In[9], MVC design pattern has been implemented for one of the distribution system application effectively. The paper[10] targets on design extensibility and reusability of the framework rather than a specific algorithm for optimal DR placement based on minimal power losses. Moreover, the development methodology adopted was also tiresome for the simple objectives. For building an MVC oriented application along with libraries and utilities for making MVC development faster and easier, struts[11] provides the foundation or framework.

This paper addresses reusability of struts software framework at the domain level. The domain to be addressed here is distribution feeder reconfiguration algorithms[9] and[12-14], including component models, topological traverses, efficiency of algorithms, line losses and load currents before and after each stage of reconfiguration. The features of the framework are as follows,

The framework proposed serves as a platform for software engineers who work in power distribution feeder reconfiguration.

Clean separation of responsibilities and reducing coupling between application layers.

Standard design patterns are applied for the internal design of layers (Model, View and Controller) to achieve maintainability.

The MVC architecture of the Struts is layer driven and levelizable[7].

MVC Components are developed and their dependency is driven by interfaces.

Components are independent, thus any component can be modified without disturbing the other components.

Organization of the paper is as follows. In Section II, Software Framework and struts framework components preparation has been defined and Section

Corresponding Author: Thiruvenkadam, S,. Department of Electrical and Electronics Engineering, Dr. Mahalingam College of Engineering and Technology, Pollachi, Tamilnadu, India,
E-mail: thirutamil_s@yahoo.com

III describes the flow of execution to better understand about the data flow happens inside the software framework. A field experience of the distribution system feeder reconfiguration with software reusability and application of the proposed method is reported in Section IV. Section V presents the benefit of struts framework. Section VI presents the Conclusions and Future works.

Ii. Software Framework: The proposed method uses the MVC architecture based software framework and solves many of the inherent problems with the previous works presented[1-4]. This architecture has been built as three independent and virtually coupled layers which have been shown in the Fig. 1. The independent layers shown in figure are responsible for a specific task and they get the dependency by the interfaces.

The business layer keeps distribution feeder reconfiguration algorithm. As stated earlier, there were many algorithms developed for the reconfiguration of the distribution feeder aiming on optimization at the feeder level. The business logic has been developed for the same, which works along with the remaining layers of the software framework.

A. Concept of Struts Framework: Struts framework streamlines the building of applications based on the MVC design principles. It provides the foundation for building such a design along with libraries and utilities for making development faster and easier. Fig. 2 shows the architecture of building the struts framework based design pattern. Struts provide the basic structure and outline for building that application, freeing us to concentrate on building the business logic (reconfiguration algorithms) of the application and not the 'plumbing'.

B. Layer Components Preparation:

1. Model Layer Component: In an MVC application, this layer component is typically the largest and most important piece. The model layer is designed to house the business logic and data access code; in other words, the model consists of the core of the application in the form of java beans component and defines what application does. The view and controller interact with the model and provide a user interface to it. The MVC architecture dictates that the model layer should be self-contained and function independently from the view and controller layer.

The typical model layer of a correctly designed MVC application can be broken down into three conceptual sub-layers. Each sub-layer can be thought of as a component or responsibility of the model. Fig. 3, illustrates this breakdown.

Each sub-layer shown in Fig.3 does not necessarily represent a separate of classes, but rather the set of responsibilities. It may choose to house a specific function's code for all layers in one large class, or it may break down each sub-layer into fine-grained objects. In this paper, single large class Reconfiguration keeps the function code. The level of object granularity is up to the developer, and what's best and/or necessary really depends on the size and complexity of the application. The three sub-layers are as follows,

External Interface: Composed of code that provides an interface that external code uses to interact with the Model.

Business Logic: Encompasses the bulk of the Model code and provides the business functionality for an application.

Data Access: Composed of code for communicating with an application's data sources such as a database.

The responsibilities of this layer are classified into different groups as follows.

1. It is responsible for accessing and manipulating the data's of the Data Sources; by Get/Set methods.
2. Keeps the queries to retrieve the Power Distribution system data's (line, bus data and switch status) from the Relational Database Management System (RDBMS).
3. Produces the load flow related parameters such as voltage, current, power, etc.

1.1 Identifying System Necessary Parameters for Reconfiguration: The objective of the feeder reconfiguration lies in finding the best switching sequences so that the real power loss minimization and the load balance can be accomplished. The notations I_rfi_i and I_bfi are employed to indicate the relief and burden feeders individually. I_rfi_i is the current of relief feeder f_i after relieving load by switching and I_bfi_i is the current of burden feeder f_i after accepting load by switching.

The ideal load level for a distribution system feeder is defined by Equation 1, which is called ideal current of the feeder.

$$I\_ideal(i) = LR\_sys * RC\_feeder(i) \quad (1)$$

where,

$$LR\_sys = AC\_sys/RC\_sys$$
$$AC\_sys = \Sigma\ AC\_trans$$
$$RC\_sys = \Sigma\ \min(RC\_trans, \Sigma\ RC\_feeder(i))$$

A simple equation was designed by Civanlar *et al*[11] for estimating the amount of the loss change in the distribution system with perfect VAR compensation. LR of the system should be assumed as less than 1. The loss change results from transferring a group of

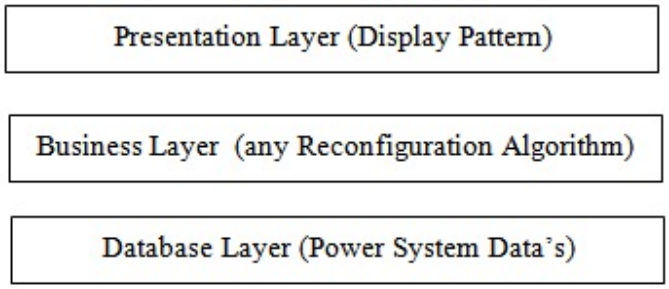

Fig. 1: MVC Design Architecture

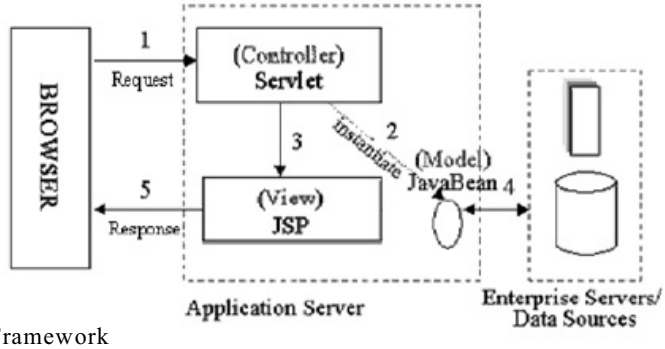

Fig. 2: Struts Software Framework

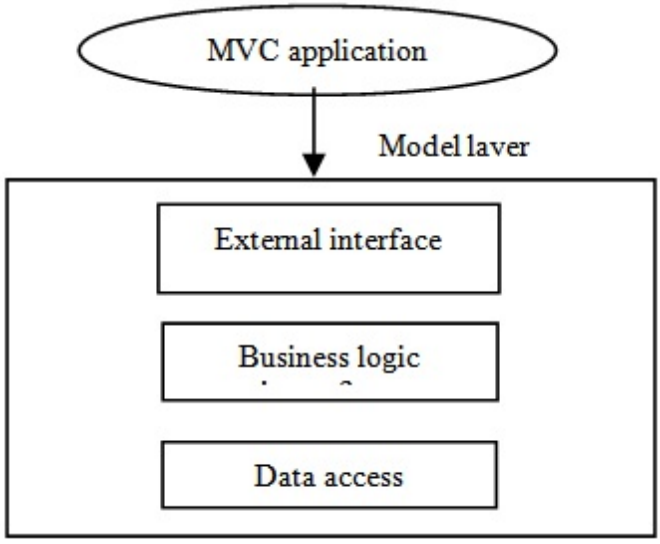

Fig. 3: Model layer Breakdown

loads from feeder-II to feeder-I, The relationship of the feeders is shown in figure below.

$$delP=2I(x)\ (E\_m - E\_n)+R\_loop[I(x)]^2 \quad (2)$$

Because of the quadratic nature of delP, the optimal distance current I_otc can be shown and calculated by differentiating Equation(2) with respect to I(x) and equate it to zero,

$$I\_otc=(E\_m - E\_n)/R\_loop \quad (3)$$

Therefore, the minimum power loss can be obtained by substituting the above equation x in equation 2 we get,

$$Del\_P\_min= (E\_n - E\_m)^2/R\_loop \quad (4)$$

The transferable current must be close to I(x_opt), for acquiring the near minimum power loss as revealed in Equation 4. Consequently, an optimal transferable current I_otc to evaluate the level of the loss reduction caused by transferring a group of loads from one feeder to another.

Therefore, it is of most important to receive the distribution system parameters such as LR, AC, RC, etc. from the user as request parameters for the reconfiguration to be prepared. It is observed that view layer component (user interface) should have the provision of receiving these parameters of the distribution system which is to be reconfigured.

There are many algorithms[9-14] has been introduced for quick decision-making process. Each technique follows its own method of finding the solution minding with minimum time consumption. This paper uses one of the successful demonstrated methodologies; a heuristic based fuzzy operation[9] is applied towards avoiding the numerical burden. Consequently, the search space of the feasible solutions can be confined by looking for a group of selected switching operations instead of exhaustive search.

1.2 Fuzzy Set Models Preparation: Two fuzzy-set models are developed one for loss reduction and another for load balance as discussed in[9]. The method for loss minimization has been developed,

```
public float jLoss(float I_otc,float I_sfi)
{
float f;
delI=I_sfi-I_otc;
f=-(I_otc);
if((delI>0) && (delI<I_otc))
return(1-(delI/I_otc));
else if((delI<0) && (delI>f))
return(1+(delI/I_otc));
else
return (0.0);
}
```

Similarly, another method for load balance has been developed after identifying the necessary parameters for load balance. The method for relief/burden feeder,

```
public float jrelief (float I_di,float I_rfi)
{
float Ib;
delI=I_rfi - I_di;
I_b=sqrt(((1.1*250)- I_di)*((1.1*250) – I_di));
if((delI>0) && (delI<I_b))
return(1 - (delI/I_b));
else if((delI<0) && (deli > - I_b))
return(1+(delI/I_b));
else
return (0.0);
}
```

The load balance level of the selected switch operation can further be defined when jburden and jrelief are determined. It can be expressed as follows,

$$jLoad(I\_di, I\_fi) = \min[jrelief, jburden] \quad (5)$$

Thus, an optimal decision fuzzy set D can be designated as follows,

$$M\_D(X,Y) = \max\{\min[jLoad, jLoss]\} \quad (6)$$

These methods are used to collect the necessary parameters from the request received through the controller layer and calculates distribution system line loss and distribution feeder loading at the each stage of reconfiguration A Java class, JReconfiguration has been developed which uses the reconfiguration algorithm shown in Fig. 4. to make the quick decision in finding the best switch pair between the feeders focusing on the objectives, jRadial method which keeps the radial load flow works along with jLoss and jLoad methods.

1.3 Preparation of Interfaces: There are many types of electric devices, or components, in power distribution systems such as cables, transformers, switches, loads, and so on. All these components hold some common parameters like voltage, current and power. Therefore it is most important in the preparation of the common parameters of the power distribution components.

Common parameters of the buses of distribution system are identified and the template/interface has been created for the entire domain as,

```
interface JBus_Interface
{
Complex getV(); // Voltage at Bus
Complex getS(); // Power at Bus
}
```

Common parameters of the lines of the distribution system are then identified and the template/interface has been created like,

```
interface JLine_Interface
{
Boolean getSwitch_status(); // get swich status
Complex getLine(); // get Line impedance
Complex getVs(); // get sending end voltage
Complex getVr(); // get receiving end voltage
Complex getI(); // get current through the line
float getP(); // get the real power flow on the line
}

Class JReconfiguration implements JBus_Interfaces,
JLine_Interfaces
{
//override all the methods of the interface
JLine_Interface
// override all the methods of the interface
JBus_Interface
//write java coding for the algorithm
//JReconfiguration (Fig.4)
}
```

Java Beans/Model component can be created and included for any other algorithms which outfit for the application and does not injure the rest of the layers.

2. View Layer Component: View layer should be designed to provide an interface to the application. JSP component has been created to accept the essential parameters of the system that is to be sent as a request to the Controller layer. All these data's must be made available before the restructuring have been done. In addition to this, request takes two more additional informations which are 'sys1_bus_dsn' and 'sys1_line_dsn', nothing but the Data Source Name specified for bus data and line data of any system.

Initial requests by the View layer to the controller Servlet are sent via the Initial.JSP shown below,

```
<%@ taglib uri="http://struts.apache.org/tags-html"
prefix="html" % >
<html>
<body>
<html:form action="/search">
Rated Current <html:text property=" RC " />
Actual Loading <html:text property=" AC" />
Ideal Load Current <html:text property=" LC" />
System Loading Ration <html:text property="LR" />
Base MVA <html:text property="base_MVA" />
Base Voltage <html:text property=" base_voltage" />
Line Data's <html:text property="line_dsn">
Bus Data's <html:text property="bus_dsn">
<html:submit/>
<body>
<html>
```

View components are also responsible to generate the response to the browser. View component has been developed with the help of JSP technology, which is used to display the individual line loss and feeders loading under the each stage of reconfiguration of distribution system. The display of the response and the parameters want to be viewed can be modified based on the user needs. The changes can be done by the user requirements and any view technology can be used, which does not create any impact on any of the layers of this application. This layer is wholly responsible for generating the response.

JSP pages are at the center of the View components; they contain the HTML that is sent to browsers for users to see and they contain JSP library tags. The library tags are used to retrieve data from Form Beans. Form Beans provide the conduit for transferring data between the view and controller layers of Struts applications.

Form Beans are basic Java beans with getter and setter methods for each of their properties allowing their data to be set and retrieved easily. The view layer populates Form Beans with data coming from the HTML interface. The controller layer then takes the Form Beans and manages getting their data and putting it into the Model layer. Additionally, the controller layer takes data from the model layer and populates Form Beans so that the data can be presented in the view layer.

3. Controller Layer Component: Controller is typically a servlet that receives the requests for the application and manages the flow of data between the model layer and the view layer. It processes the request and validates the data. It controls the way that the model and view layers interact. This is responsible for receiving all incoming requests to the application. Upon receiving a request, its processing to the Struts requests processing engine. The request processing engine processes all aspects of the request. The Fig. 5 shows the components of the controller layer and their data control.

4. IDE's for Framework Development: Integrated Development Environments (IDEs) are providing good support in developing Struts like frameworks. There are several kinds of IDEs available in the marketplace. NetBeans is one of the most famous IDE which is well suitable for developing Struts Frameworks. The Framework, which proposed to create, is a group of different software packages (JavaBeans, JSP, Servlet, JDBC, HTML, XML, etc.). All needs different environments to develop and execute. NetBeans Provides an environment there we can create any software application very much easier and can be grouped into a single package (Struts Framework). Most importantly, the NetBeans IDE is user friendly to create the software packages and keeps the supporting servers (Tomcat) in it. In this paper, the struts software framework has been developed using NetBeans IDE.

III. Flow of Execution: Step 1: The browser (View) makes a request to the struts application that is processed by ActionServlet (Controller). Request has been made with the distribution system essential parameters such as bus data, line data, LR, RC, LC, AC, Base MVA and Voltage.

Step 2: Action Servlet (Controller) interfaces with the JReconfiguration object (Model) to perform the reconfiguration of the requested system.

(The Distribution system data's have been kept at the Data Sources. Data Sources are exactly the RDBMS)

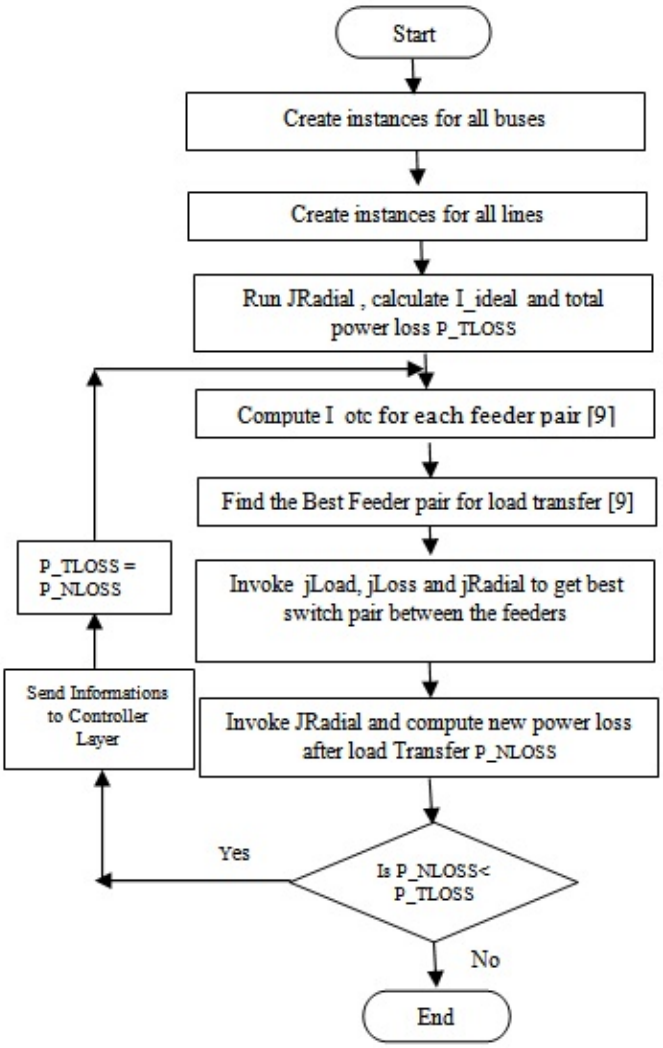

Fig. 4: JReconfiguration Algorithm

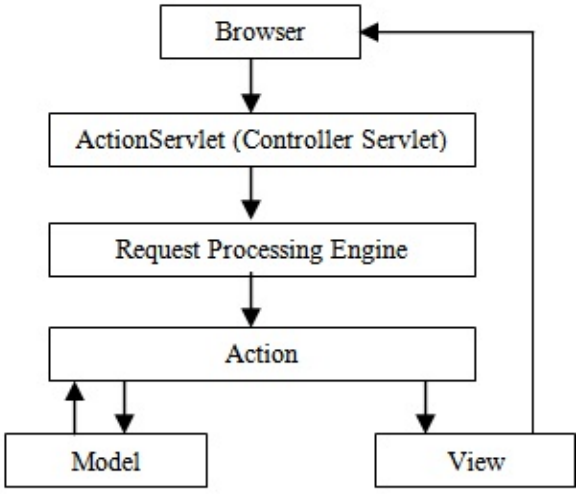

Fig. 5: The Controller layer lifecycle

Step 3: Structured Query Language (SQL) has been used to extract the requested (View) distribution system data's from the Data Source to the Model layer.

Step 4: Model layer then manipulates the data according to the business logic with retrieved data's from the Data Sources and sends the manipulated data's to the Controller layer.

Step 5: Controller layer then prepares the information requested by the user and forwards control to the JSP (View).

Step 6: JSP (View) send response to the browser, the way user expects the informations to be displayed.

IV. Case Study: The three feeder distribution system shown in Fig. 6 has been used to interpret the procedure of heuristic fuzzy search. This network has 16 buses and 16 lines. The initial status of the switches is shown for the network which produces the line loss of 102.6 kw., the line and bus data are prearranged separately for the given system as Sys1_line_data and Sys1_bus_data. The maximum current rating of the line in the system is 225A. The other parameters of the system are,

Rated current RC_{sys}= 675A; Actual loading AC_{sys}= 422.6A Ideal load current I_{di}=140.8

System loading ratio LR_{sys} =0.704 Base MVA = 100 MVA & Base KV = 23KV

All the three layers have been created for the current application (Distribution System Reconfiguration). Fig.7 demonstrates the view layer request page. The request has been attempted by the user are pre reconfiguration system parameters (LC, AC, RC, etc), line and bus Data Source Name (sys1_bus_dsn and sys1_line_dsn). Once the request made by clicking the Reconfigure button, which immediately takes those parameters to the controller layer. Controller layer acts like a bridge between view and model.

The request has been accepted and posted to the model layer. Model component exactly keeps the business logic. It processes/manipulates data's of the application which are received by the model layer. This is the only component interacts with the data sources. Model layer collects the requested system line and bus data's from the data source by referring their data source name. The instants of the lines and buses formed. Based on the business logic embedded in the component, it manipulates the system data's and produces the required informations.

The processed information is received by the view component (JSP) via the controller layer. JSP prepares the response and displays in the browser. Fig. 8 demonstrates the experience of response produced by the JSP component of the View layer. Fig. 9 and Fig. 10 represent graphical representation of the reduction in power loss and the load allocation for the each feeder at the each stage of reconfiguration.

Finally, this proposed architecture clearly shows the separation of business and presentation logic. Having the separation of business and presentation code accommodates multiple interfaces to the application such as web, wireless, or GUI (Swing). Additionally, this separation provides excellent reuse of code and anybody can modify any layer devoid of perturbing the existing layers of the system.

V. Benefit of Struts Framework: This struts based framework provides an understandable depiction for developing the reusable software for Distribution system reconfiguration problems considering the objectives line loss reduction and load balancing. With this comprehension, the framework users can build their own, high-level analysis based on the struts framework. In addition to the benefits (1-5) addressed in[10], this style of framework provides better perceptive, fewer number of layers, visible data flow, trouble-free to build the software frame work to the users and very well suitable for distributed computations.

VI. Conclusion and Future Work: Frameworks are a promising technology to reuse design for an entire domain to reduce cost and improve software quality. This paper presents an architectural design for a framework aiming in distribution feeder reconfiguration problems. The MVC architecture of the struts framework is layer-driven with strict top-down dependencies to reduce software coupling and to gain reusability and extensibility. Standard interface patterns are applied to the layers to achieve maintainability, which also supports the idea of distributed computations. The framework proposed emancipation the user to think of interfacing the layers.

This work can be proposed by including the further constraints like phase balancing of the three phase unbalanced Distribution system under normal and contingencies. This may work with SCADA since the frame work which developed well suit for the online and remote applications.

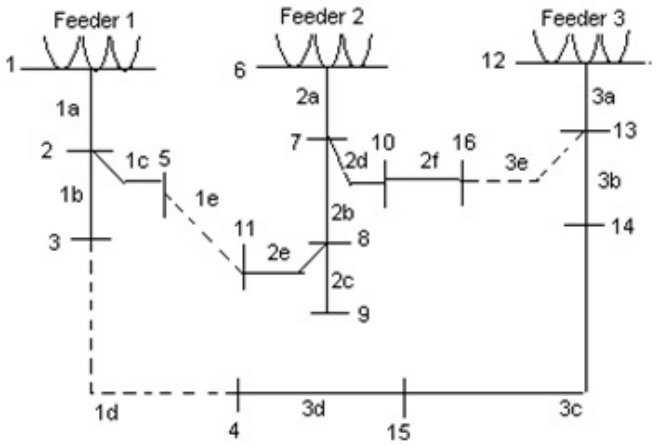

Fig. 6: Three feeder simple

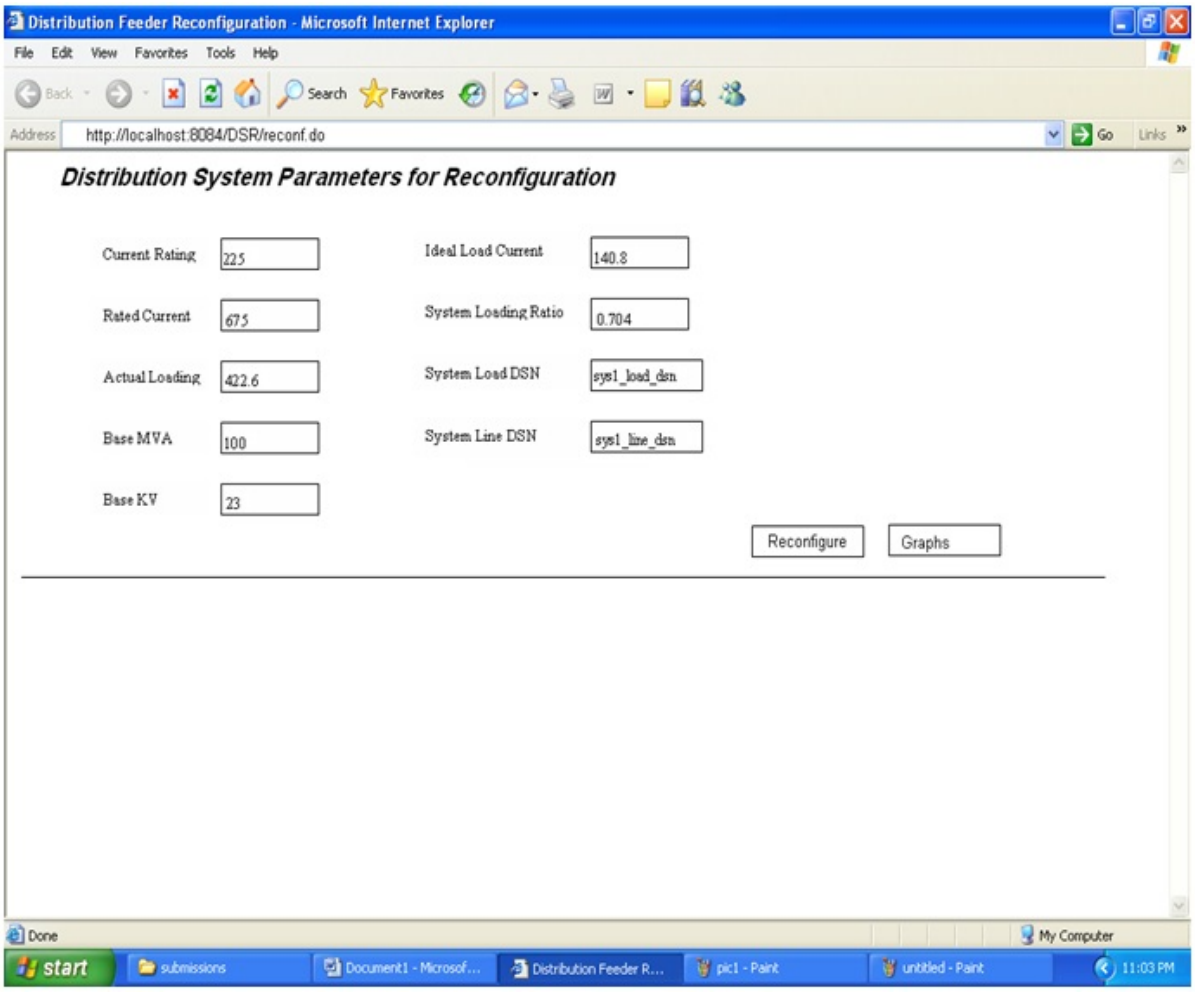

Fig. 7: Request/Response Screen (1)

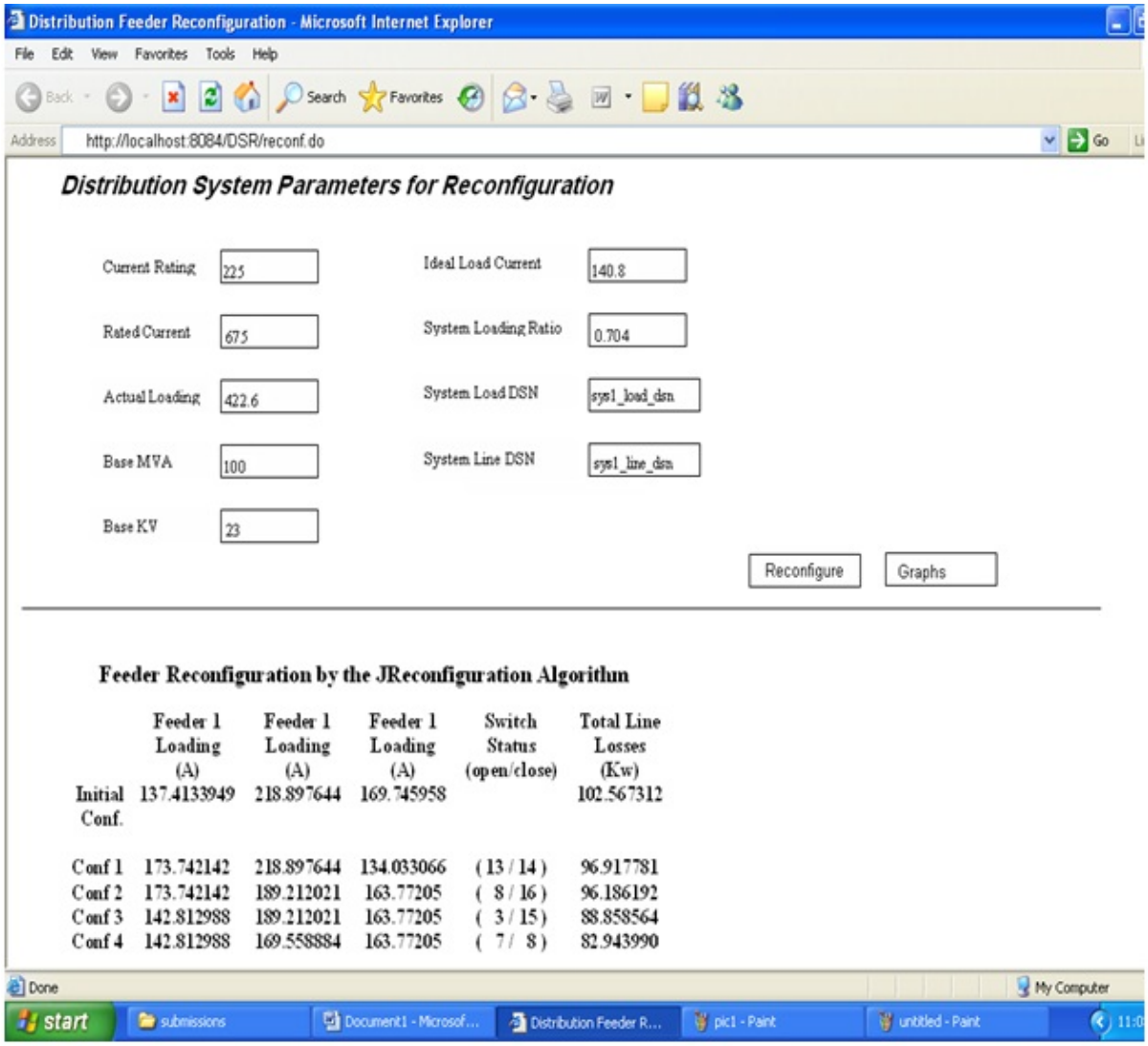

Fig. 8: Request/Response Screen (2)

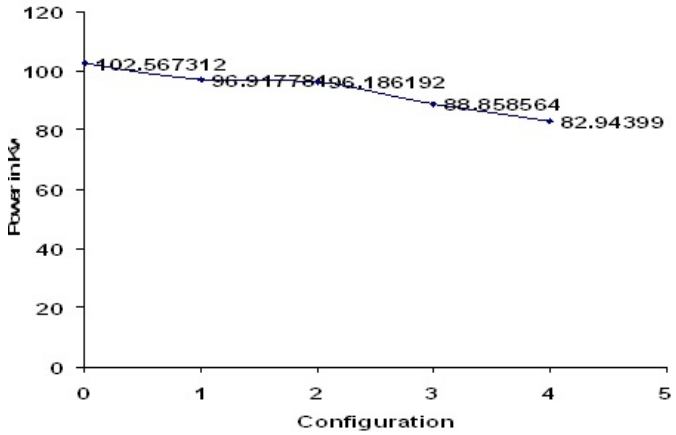

Fig. 9: Configuration Vs Power Loss

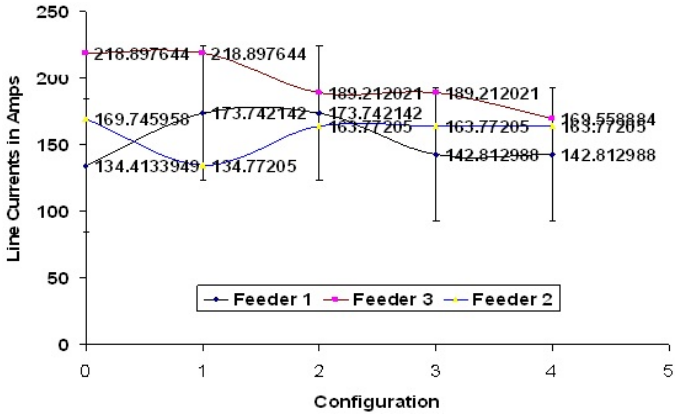

Fig. 10: Configuration Vs Feeder Currents

Appendix A. Nomenclature

AC_sys	Actual current of System
AC_trans	Actual Current of Transformer
API	Application Programming Interface
E_n	Voltage at bus n
E_m	Voltage at bus m
I_ideal(i)	Ideal Current of Feeder i
I_otc	Optimum Transferable Current
I_rfi	Relief Feeder i current
I_bfi	Burden Feeder I current
LR_sys	Loading Ratio of System
MVC	Model View Controller
RC_sys	Rated Current of System
RC_feeder(i)	Rated Current of Feeder i
RC_trans	Rated Current of Transformer
RC_feeder(i)	Rated Current of Feeder i
R_loop	Series resistance of the path connecting two feeders i and j

REFERENCES

1. Zhu, J. and P. Jossman, 1997. "Application of design patterns for object-oriented modeling of power systems", IEEE Trans. Power syst., 14: 532-537.
2. Zhu, J. and D. Lubkeman, 1997. "Object-oriented development of software for power system simulations", IEEE Trans. Power Syst., 12: 1002-1007.
3. Pandit, S., S. Soman and S.A. Khaparde, 2000. "Object-orinted design for power system applications", IEEE Computer Applicat. Power, 13: 43-47.
4. Losi, A., M. Russo, 2003. "Object–oriented load flow for radial and weakly meshed distribution networks", IEEE Trans. Power Syst., 18(4): 1265-1274.
5. Lakes, J., 1996. Large-Scale C++ Software Design", Reading, MA: Addison-Wesley.
6. "Building Application Frameworks: Object-oriented Foundations of Framework design", M.E. Fayad *et al*., Eds., Wiley, New York, 1999.
7. Broadwater, R.P., M. Dilek and J. Thompson, 2001. Centralized, distributed responsibility, and decoupled object-oriented software designs for power systems," in Proc. IEEE Power Eng. Soc. Summer Meeting, 2: 1025-1028.
8. Gamma, E., *et al.*, 1997. Design Patterns-Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley.
9. Thiruvenkadam, S., A. Nirmalkumar, A. Sakthivel, 2008. "MVC Architecture based Neuro-Fuzzy approach for distribution feeder reconfiguration for loss reduction and load balancing," Proc. Int. Conf. IEEE-PES Transm., and Distrib., Chicago, USA.
10. Fangxing Li, Robert P. Broadwater, 2004. "Software frame work concepts for Power Distribution System Analysis", IEEE Trans. Power Syst., 19(2): 948-956.
11. James Holmes, 2007. Building Struts Software Framework: Struts The Complete Reference.
12. Civanlar, S., J.J. Grainger, H. Yin, S.S.H. Lee, 1988. Distribution feeder reconfiguration for loss reduction, IEEE Trans. Power Del., 3(3): 1217-23.
13. Kim, B.H. and R. Baldick, 2000. A comparison of distributed optimal power flow algorithms, IEEE Trans. Power Syst., 15: 599-604.
14. Venkatesh, B., R. Ranjan and H.B. Gooi, 2004. Optimal reconfiguration of radial distribution systems to maximize loadability, IEEE Trans. Power Syst., 19(1): 260-266.