WWW based computation services: Transfer of power system applications to the WWW

Rainer Bacher Tina Orfanogianni
Rainer.Bacher@eeh.ee.ethz.ch, Tina.Orfanogianni@eeh.ee.ethz.ch
Swiss Federal Institute of Technology (ETH), CH 8092 Zürich, Switzerland

Abstract: This paper describes the necessary software engineering and development steps to transfer a "legacy" power system application, i.e. an ASCII input file and output file based program without user interactions, to the World Wide Web (WWW). An Optimal Power Flow (OPF) program is used as an example of such a "legacy" application. The paper describes development efforts on the WWW-client and on the WWW-server side. Also, a discussion of changes to an existing OPF code is given. The paper discusses the complexity of development efforts together with advantages and disadvantages of the presented approach both for the WWW-OPF developer and the WWW-OPF user.

Keywords - Internet power applications, WWW (World Wide Web), Legacy Applications, Software upgrade

1 Introduction

In the past many software applications have been developed based on the computer and software technology available at that time. Huge efforts have been put into sophisticated software packages such as optimal power flows, state estimation, load prediction software, etc. The programming language for these application is usually FORTRAN[-90], C or C++.

Many of these applications are based on simple input and output principles, see Fig. 1.

In Fig. 1, a set of ASCII-Files define the input (i.e. the values for the model parameters) of the application. The application produces during and at the end of the application run ASCII-output only. Usually no or only a few simple user interactions - usually at the beginning of the program execution - are possible.

Technology is advancing rapidly and especially the GUI (Graphical user interface), the access to databases has changed in recent years and will even change at a faster speed in the future. In addition the computer hardware is getting faster and cheaper. Perhaps the biggest innovation has been the internet which connects many millions of computers.

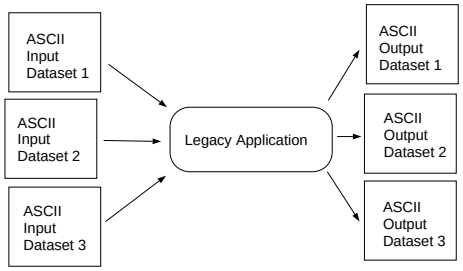

Figure 1: ASCII Input and output of legacy applications

These innovations influence the features of power applications and as a consequence their development processes. One cannot disregard the fact that new user interaction environments have been created, that computers are used in a much more interconnected way and that computing power is shared for CPU intensive applications.

Thus, the legacy power applications must be adapted to these environmental and technological changes. As a software developer of an application with complex algorithmic background, one is aware that often, much time and money has been invested in these applications in the past. As a consequence it is not possible to restart development from scratch, even when environments where the application is embedded, change drastically.

However, in order to have some continuous adaptation to software environment and technology changes, some changes must be done. If this adaptation does not happen, the software developer will - at some point in the near future - get stuck with his legacy software, due to incompatibility with the new environments. Very high investment will be needed for a late adaptation of these application to changed environments.

In order to avoid these future incompatibilities and in order to avoid huge future investments, we propose to adapt some of the "legacy" applications to the WWW-environment.

This paper describes the necessary software engineering steps to make a "legacy" power system application such as a conventional Optimal Power Flow program available

to any WWW (World Wide Web) user in the world who has access to the internet. The main motivations for such a project are:

- Continued usage and upgrade of "legacy" software in a modern environment.
- Offering computational services for complex classes of problems (possibly combined with a usage charge) over the internet.
- Exposing developed software prototypes to users in the world and getting "real-world" feedback.

We assume that such a "legacy" application is a FORTRAN or C (OPF) program which needs one or more ASCII data files as input. An example for an input is an IEEE format file for an AC power flow. In addition there are program control parameters (for example general limits for voltage magnitudes, optional control of area exports, choice of a rectangular/polar coordinate system) which the program user can set in order to manipulate certain execution options of the program. The "legacy" program is not interactive: It reads the user-defined input data and parameters from ASCII input files followed by the execution of complex algorithmic steps. Finally the server-side "legacy" application outputs result data in ASCII-format.

The paper describes the various steps for a WWW developer who wants to port this "legacy" application to the WWW with minimal changes to the application code itself. Figure 2 shows the software environments in which these developments are done:

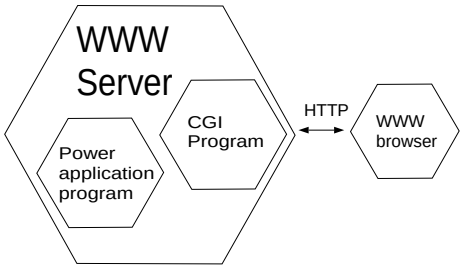

Figure 2: Main environments for the WWW-application developer and user

We assume that a WWW Server is available. In the project described in this paper the free Apache WWW server (http://www.apache.org/) Version 1.2.6 is used. All steps to install the WWW server software have been successfully executed. Also, we assume all default options of the Apache-Server software are activated and available.

Different ways exist to bring "legacy" software to the internet. The effort varies mainly depending on the features used when interacting with the internet and in the graphical appearance on the WWW browser. Today, Java could be used to develop and provide advanced graphical and WWW interaction features. In this paper, however, we concentrate on software development with minimal development efforts on all pieces of software. Also, we do not want to use any special, non-standard software pieces except the basic WWW server and browser features[1].

The three places where development needs to take place are (see Fig. 2) in the **CGI**[2]-program, in the **WWW-browser** and in the **"legacy" power application** program.

The three mentioned software pieces must interface to the WWW Server (Apache) in such a way that a modern, WWW based power application usage environment results with minimal coding efforts for the developers.

These steps include

- HTML development for the WWW pages on the client side, i.e. an input file for the WWW browser (section 2).
- PERL code development for parsing the uploaded file which arrives at the WWW server side. PERL code development for starting the "legacy" application (the OPF) on the server side. These steps are summarized as CGI program development (section 3).
- FORTRAN code (used in this paper as the main development language of the OPF legacy application) development for changing the "legacy" FORTRAN application (minimal changes) including generation of WWW-HTML-output and creation of other output files (possibly taken as input for a follow-up run) (section 4).

The paper ends with conclusions in section 5, a list of references and a short biography of the authors.

2 WWW browser developments: WWW client html-file

Fig. 3 shows a typical entry screen for accessing the WWW interface of an Optimal Power Flow application, accessible via the internet at http://www.eus.ee.ethz.ch/services/power_apps/Adopfform.html [3].

This WWW page is developed and stored on the WWW-server, but "downloaded" and used by the WWW-OPF user at the client-side WWW browser.

[1] One must be aware that this statement is based on mature technology available at the time of writing this paper.

[2] The Common Gateway Interface (CGI) allows the passing of a URL (Uniform Resource Locator) request to a service invoked by the WWW server. When a URL begins with "http:" the WWW client browser gets all information from the WWW server using a protocol named HTTP. A WWW server based service invoked via CGI writes data to its standard output and the web server passes that data directly back to the client WWW-browser.

[3] The string "Adopf" comes from the fact that for the OPF, a software engineering approach based on the ADIFOR - Automatic differentiation (Ad) of Fortran programs - has been taken ([1, 2]).

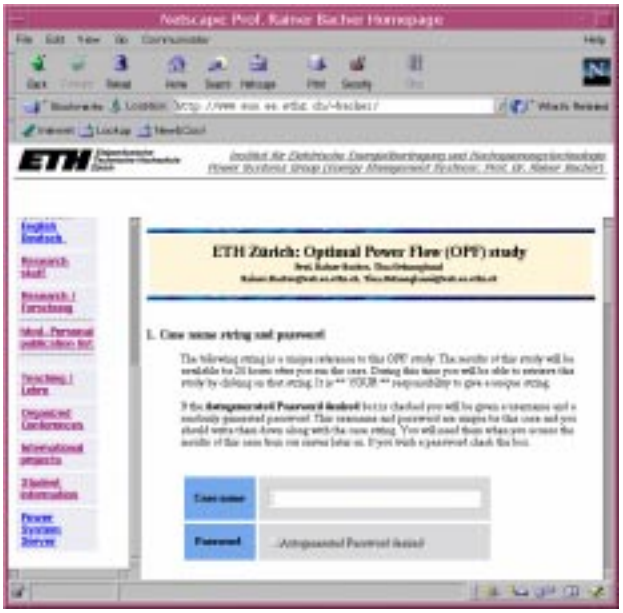

Figure 3: Entry Screen for WWW optimal power flow user parameters and application start: http://www.eus.ee.ethz.ch/services/power_apps/Adopfform.html. Lower part: "SUBMIT"-button to start WWW based OPF program on WWW-server

HTML allows developers to create FORMS on the client computer. FORMS allow users to enter information using text boxes, check boxes, etc. Every HTML FORM specifies a WWW service in its URL, e.g. "/cgi-bin/services/power_apps/power_apps.pl". When the user pushes the SUBMIT-button for a form, the WWW browser sends the form data to the specified WWW service on the WWW server. The WWW service handles all data that the user has entered on the FORM.

The web browser delivers information to the WWW service as a collection of FORM variables, passed as a block of data. This block of data must be analyzed and separated again, see section 3 of this paper.

On the FORM of Fig. 3, the user can give all essential parameters of an optimal power flow. The parameters can be divided into two types: On the one hand, the user provides the name of an ASCII data file with data related to the network to be optimized. The format of this data file is "standardized", e.g. [3] can be used. In this datafile the user enters all parameters related to actual network topology N-port model parameters such as e.g. resistance for lines, transformers and desired voltage amplitudes controlled at generators or active / reactive powers at loads.

On the other hand, the user enters study related parameters such as a user defined name for the optimization study to be done next, the "standardized format" of the datafile mentioned before and some options related to the algorithm implemented in the OPF program which will be executed after pushing the "SUBMIT"-button at the end of the WWW page, see lower part of Fig. 3. These options include the type of mismatch equations (Current mismatch, power mismatch), the choice of complex type variables (rectangular or polar), the name of an optional file with data related to an efficient start of the optimization algorithm (the program routine MINOS [4] is used as part of the OPF algorithm) and some optimization inequality and objective function related values which are not part of the "standardized" network data mentioned before. Typical values include global upper and lower limits for voltage magnitudes, a choice between various objective functions (network or area losses; Future development: Bid based power exchange cost minimization/profit maximization, independent system operator bid based network security achievement, etc.).

A typical html-file (extracts from this file) for the WWW-CLIENT-PAGE looks as follows:

```
<html>
<head>
<TITLE>ETH Z&uuml;rich: Optimal Power Flow</TITLE>
</head>
<body bgcolor=white>
...
<form ENCTYPE="multipart/form-data" method=post
action="/cgi-bin/services/power_apps/power_apps.pl">
...
Case string: <input type="text" size=40  name="casestr">
...
<input type="checkbox" name="password_desired" >
Autogenerated Password desired
...
 2.  Give the input data file and the format:
Input File: <INPUT TYPE="file" SIZE=45 NAME="NetFilename1">
...
Format of input file:
    <SELECT NAME="Data format">
      <OPTION SELECTED>IEEE common format
      <OPTION> PTI format
    </SELECT>
...
3. Select the options for this OPF study
Mismatch
<input type="radio" name="mismatch" value="Current">Current
<input type="radio" name="mismatch" value="Power" checked>Power
Minimum LoadVoltage
<input type="text" size=10 value="0.9" name="Min Load Voltage">
...
<INPUT size=30 TYPE=submit value="Start OPF (warning: after
pushing, you will have to wait some time) " >
<INPUT TYPE=reset>
</form>
...
</body>
</html>
```

The developer of this client-side WWW page must understand the syntax and contents of the various fields. Hyperlinks can easily be added to this page to access WWW-help screens for usage of fields, buttons, network data and optimization theory of the provided WWW based OPF program.

3 CGI programming

3.1 User OPF data upload

Upon pushing the "SUBMIT"-button, the following steps occur[4]: First, all data given in the CLIENT-WWW-PAGE including the contents of the files whose names have been given, is collected by the browser executable into one file.

The contents of this file have a structure which allows the separation of all fields (i.e. their names and types) defined on the WWW-client page together with their values given by the WWW-client user. A part of this file looks as follows:

```
-----------------------------295482227830232
Content-Disposition: form-data; name="casestr"

test
-----------------------------295482227830232
Content-Disposition: form-data;
name="NetFilename1"; filename="ieee14.dat"

 09/25/93 UW ARCHIVE           100.0  1962 W IEEE 14 Bus
BUS DATA FOLLOWS                            14 ITEMS
   1 Bus 1     HV  1  1  3 1.060    0.0      0.0      0.0
   2 Bus 2     HV  1  1  2 1.045  -4.98     21.7     12.7
...
  14 Bus 14    LV  1  1  0 1.036 -16.04     14.9      5.0
-999
BRANCH DATA FOLLOWS                         20 ITEMS
   1    2  1  1 1 0  0.01938   0.05917     0.0528     0
   1    5  1  1 1 0  0.05403   0.22304     0.0492     0
...
  13   14  1  1 1 0  0.17093   0.34802     0.0        0
-999
LOSS ZONES FOLLOWS                    1 ITEMS
  1 IEEE 14 BUS
-99
INTERCHANGE DATA FOLLOWS                 1 ITEMS
 1    2 Bus 2     HV    0.0  999.99  IEEE14  IEEE 14 Bus
-9
TIE LINES FOLLOWS                     0 ITEMS
-999
END OF DATA
-----------------------------295482227830232
Content-Disposition: form-data; name="Data format"
IEEE common format
-----------------------------295482227830232
Content-Disposition: form-data; name="mismatch"
Power
-----------------------------295482227830232
...
-----------------------------295482227830232
Content-Disposition: form-data; name="Min Gen Voltage"
0.9
-----------------------------295482227830232
Content-Disposition: form-data; name="System_losses_min"
At whole network
-----------------------------295482227830232
```

On the server, a "PERL-CGI-PROGRAM" must be started which takes this file apart. The main points are: Detection of the separator string "295482227830232", detection of name of each field (e.g. "Min Gen Voltage") and its contents ("0.9") for all fields, extraction of the contents of the file "name="NetFilename1"; filename="ieee14.dat"" to the server disk.

Often, due to the high flexibility in string manipulation, a perl script is doing this job. Note that the name of this perl script is given as an attribute of the FORM-tag on the CLIENT-WWW-PAGE (see section 2, the string "/cgi-bin/services/power_apps/power_apps.pl").

3.2 Internet USER data upload

One must be aware that upon pushing the "SUBMIT"-button, more data is uploaded than just the field and file data mentioned in the subsection before. In addition we always get the four following inforation pieces:

```
1) HTTP_USER_AGENT:
   Mozilla/2.0 (compatible/ MSIE 3.01/ Windows 95)
2) HTTP_ACCEPT:
   image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
3) REMOTE_HOST: gate.acompany.ch
4) HTTP_ADDR: 194.777.666.555
```

Especially information 4) is important: It identifies the computer on the internet where the access comes from, i.e. this is the address of the computer where the "SUBMIT"-button is pushed. This variable represents one piece of information to identify the WWW-OPF user. From 1) we can see what type of operating system the WWW-client is using (here: Windows 95), also, we see the type of internet browser (Microsoft Internet Explorer V3.01).

3.3 Summary of server side steps: CGI-Programming

To summarize, the CGI-perl script does the following jobs:

1. Split / parse user WWW-client-form data, 2. Create server directory based on client data, date and time, password 3. Save uploaded file information in a server-side file which can be read by the OPF application. 4. Save uploaded field information into files compatible[5] with the OPF application. 5. Start the OPF application (Following steps done within the OPF program): 5a. Read OPF input files, i.e. uploaded files (step 3) and saved uploaded field information (step 4). 5b. Execute the OPF algorithm. 5c. Produce WWW-compatible output: Create html-files and main html-page with links to the html-OPF output; The main page is sent back to the WWW-client.

One must mention that the CGI-PERL-PROGRAM can generate at any time html-output which can either be written to the disk of the server or directly to the "WWW-CLIENT-SCREEN". As a consequence, the CGI-PERL-Programmer must design the information which goes back to the WWW-OPF-user after the "SUBMIT"-button has been pushed and after the OPF has been computed. In the current OPF design we send back one screen with a hyperlink to subscreens, see Fig. 4.

[4]The steps described are seen from a power systems engineer using software environments and not from a software specialist who sees details of all software layers.

[5]meaning: in such a way that the application can read it directly with known ASCII-formats

Figure 4: Overview output screen of WWW optimal power flow

Figure 5: Zoom into upper left part of Fig. 4

By clicking on one of the hyperlinks of the zoomed window of Fig. 5, one can access all uploaded and computed information, e.g. files which are stored on the server in a user-specific subdirectory. This directory place can optionally be password-protected. If no password is set, then in principle everybody can look at all user data. Due to the fact, however, that the server directory name (and also the WWW-address) is created using the WWW-client side computer name (e.g. the variable HTTP_ADDR) and the time when the job was created on the server, it is very difficult for non-authorized users to find e.g. the unique URL http://www.eus.ee.ethz.ch/upload/WWW_OPF_mycomputer_9811-02143153/Output/solution.html.

3.4 Interfaces: Summary

In principle the developer of all code pieces of a WWW based OPF must take care of the following interfaces: 1) Mapping the names and field contents to OPF-known strings and data formats. 2) Based on user parameters given in the WWW-CLIENT-PAGE, the OPF code developer must take care that the code actually considers the user defined parameters. 3) The CGI-PERL-PROGRAM must take care of passwords, user identifications, secure access to OPF generated output, setup of OPF-WWW compatible output screens including hyperlinks to server-side stored files.

4 Power application software development

From the steps given before one can see that only a few changes are necessary to make the original OPF code compatible to the WWW.

In principle, the input is not changed at all: This means that the PERL-CGI-PROGRAM must make the conversion from the uploaded field data (name and contents of field) to one or more ASCII files which the OPF program can read.

The only change is the output of the OPF program. Although in principle also pure ASCII-Files can directly be read by WWW browsers by just clicking on such an ASCII-file, it is advantageous to create some html-files within the OPF program in order to have a better appearance. One of the main advantage of the WWW is the use of hyperlinks even within a document. For example, the OPF output can be structured in such a way that first all buses (and branches connected to these buses) of an area are output. If the other bus of a branch is in another area one can easily create a hyperlink to this other area. Of course, this means that the OPF output code developer must modify the OPF program to produce html-compatible output strings.

In the case of this OPF application one main solution.html file is created which includes hyperlinks as described above. In addition, the users gets all information which the OPF application produces as if run on a local machine. Also, MINOS-related optimization data can be inspected by interested WWW-OPF users.

In addition, a hyperlink to a file of type .tar.gz is available: This file includes in compressed form the whole job (all uploaded data, all OPF output data). The WWW-OPF-user can download this file to the WWW-client-computer. There, all data can be inspected without network connection to the OPF-WWW-server.

5 Conclusions

The presented approach allows a power system engineer to transfer a “legacy” application such as an OPF to the WWW. The word “legacy” mainly refers to the type of input and output of these applications: They should be ASCII-based. Also, the “legacy” program just executes based on these ASCII files. No other input or command is needed to drive the OPF execution through possible subparts: All information must be given in ASCII-files before the OPF run is started.

A developer must learn to design and program html-pages: This includes learning about the use and programming of user definable inputs such as buttons, input fields, etc. This is a relatively easy task. In addition, one has to write

a [e.g. PERL based] CGI-PROGRAM. This job, however, can be supported by downloading example files from the internet and extending these files as needed. The main software problem consists of splitting several uploaded files and field information in the correct way. One must be aware that WWW-CLIENTS can be UNIX-, WINDOWS and MAC-based systems. As a consequence, the uploaded information type is not always exactly the same: For example, sometimes, a carriage-return is included, special characters appear in various forms, etc. Also, one must be aware that not all versions of WWW-browsers have the same capability. For example, some older versions do not allow file uploading. The CGI-PERL-PROGRAM should theoretically take into account all types of WWW-clients: Doing a perfect job here represents an enormous work and is almost impossible.

The main advantages of a WWW based application such as an OPF are as follows:

Advantages for the OPF software developer: 1. Offering computational services for complex classes of problems (possibly combined with a usage charge). 2. Exposing developed prototype and commercial software to the world (for free or for a charge) and getting "real-world" feedback. 3. Continued usage of "legacy" software in a modern environment. 4. Easy possibility to offer different versions of an OPF-software, i.e. only one software installation for each version for an undetermined number of users. 5. After an initial high effort, easy, fast, automatic and individualized "account" creation: Each pushing of the "SUBMIT"-button creates an individual account. The client-side user (and not the server) is responsible for managing different version of network input and output data sets. 6. Possibility to send e.g. warning-messages (E-mail) of type "User at 129.234.56.78 has just been running a successfully converging OPF" (or opposite) if somebody is using the WWW-OPF-server.

Advantages for the OPF software user: 7. No installation of any OPF-related executable software, i.e. the user always has the latest software available without individual installation. 8. Usage based charge, i.e. no use means no cost.

The main disadvantages are:

Disadvantages for the OPF software developer: 9. Restriction to WWW based interactions, file and field data uploads. 10. No real interaction of the WWW-CLIENT-browser to the internal state of a running SERVER-side OPF program. Each pushing of the "SUBMIT"-button creates an individual job. Successive jobs cannot share memory data automatically.

Disadvantages for the WWW-OPF user: 11. The WWW-OPF user gives away his/her data: The user must rely on the WWW-server-administrator that data secrecy is given and that the server-side setup works. 12. The WWW-OPF user shows some of his/her capabilities of handling OPF complexity: The OPF-server could always observe all steps which the client-side user executes. Again, the user must trust the WWW-OPF-service organization that user behavior and data is not publicized.

The WWW, associated communication services, software standards and environments are rapidly changing. These changes will certainly lead very soon to changes to the concepts as presented in this paper. The main goal of this paper is to show, that already today (1998), only a few steps and little additional knowhow about a few software technologies are needed to transfer a "legacy" application with minimal changes to existing code and minimal new code development to the WWW. Offering such a service leads to a dynamic process: If the service does not satisfy the requirements of the users, it will just not be used any more. However, if the service is on a high professional level, if it does not cost much, if the user trusts the OPF-WWW-service provider that uploaded data is handled with enough care and secrecy, and if the internet access is fast, there is no reason that such a service could not be used by professionals outside of the academic world.

References

[1] Tina Orfanogianni and Rainer Bacher. Using Automatic Code Differentiation in Power Flow Algorithms. IEEE Transactions on Power Systems, (PE-828-PWRS-0-2-1998), 1998.

[2] Tina Orfanogianni and Rainer Bacher. Increased OPF code development efficiency by integration of general purpose optimization and derivative computation tools. S*ubmitted to IEEE PICA'99, Santa Clara, CA.*

[3] IEEE Commitee Report. Common format for exchange of solved power flow data. *IEEE Trans. Power Apparatus and Systems*, PAS-92(6):1916–1925, Nov./Dec. 1973.

[4] Bruce A. Murtagh and Michel A. Saunders. MINOS 5.4 User's Guide (preliminary). Technical report, Departement of Operations Research Stanford University, 1983. Revised Jan. 1987, Mar. 1993.

Rainer Bacher received the Dipl.El.-Ing. degree in electrical engineering in 1982 and the Dr.sc.techn. degree in 1986, both from the Swiss Federal Institute of Technology (ETH) in Zürich, Switzerland. After his doctorate he joined the Energy Management Systems Division of Control Data Corporation in Minneapolis, U.S.A. In 1989 he joined Colenco Ltd., a Swiss consulting company. Also in 1989 he started as a lecturer and part-time senior researcher at the power system group at the ETH. In 1993 he was appointed assistant professor of Energy Management Systems at the department of electrical engineering at the ETH Zürich. He is a member of technical committees for PSCC 93, 96, 99 and PICA 93, 95, 97, 99 (Vice Technical Chairman). He can be reached at Internet e-mail: Rainer.Bacher@eeh.ee.ethz.ch

Tina Orfanogianni received the Dipl.El.-Eng. degree in electrical engineering in 1995 from the National Technical University of Athens (NTUA), Greece. She is currently doing her PhD in the area of power system optimization using FACTS devices. She can be reached at Internet e-mail: Tina.Orfanogianni@eeh.ee.ethz.ch