STRUCTURE OF MAPPING METHODS USED TO CONNECT ESP-r/HOT3000 TO THE WEB AND ITS INCREASED USABILITY THROUGH A NEW INTERFACE

Stephanie Mombourquette Paul Wyndham-Wheeler, P.Eng
CANMET Energy Technology Centre
Ottawa, Ontario
Stephanie.Mombourquette@nrcan.gc.ca Paul.Wheeler@nrcan.gc.ca

ABSTRACT

ESP-r/HOT3000 is a widely used building simulation engine and is the simulation engine used in this project. This engine is built upon the comprehensive and extensively validated ESP-r program developed at the University of Strathclyde (ESRU 2000). The CANMET Energy Technology Centre (CETC) has made some algorithmic additions to support the modelling of Canadian housing.

ESP-r/HOT3000 is a very complex stand-alone software programme and it's full potential is best realized and exploited by building engineers; for the average homeowner, the software is not especially easy or effective to use. It can be extremely taxing or near impossible for the homeowner to have to enter hundreds of inputs that are very technical—and possibly incomprehensible to them—but required by these more advanced software tools. What is necessary for the homeowner is a software tool that requires fewer and more understandable inputs, and runs on an accessible platform.

To address this challenge, CETC has created an on-line graphical user interface (GUI) for ESP-r/HOT3000 on the web for average homeowners to use. This paper will examine how the ESP-r/HOT3000 engine is connected with a web interface and how data is managed through an intelligent mapping and streaming process that then feeds results back to the client's interface. Next, it will consider the increased usability of the interface and the quality and features such as graphical prompts and other options, which make it easier for the client homeowner.

This site can be found at http://oee.nrcan.gc.ca/houses-maisons/english/homeowners/CHEC/index.cfm

INTRODUCTION

When designing the Home Energy Analyzer tool for the general public, it was important to take into consideration the audience that would be utilizing the tool. When CETC decided to add a web interface to ESPr/HOT3000, which could make this energy analysis software tool easier for the public to use, a lot had to be considered. ESP-r/HOT3000 is a very complex program with many inputs so it was crucial that enough information was collected from the user without bombarding them with difficult questions in order to run a valid simulation. Due to the fact that the thermal performance of residential buildings is being based on 30 inputs, choosing the right 30 questions to ask the user was critical.

The GUI was written in Hyper Text Markup Language (HTML) and scripting was done primarily in JavaScript. HTML is the basic and primary language for any web page and it was chosen because it is an industry standard and widely accepted. JavaScript was chosen as the scripting language for the GUI because it is a compact, object-based scripting language for developing client and server Internet applications and it is not browser dependent. See Figure 1 for a screen capture of the GUI.

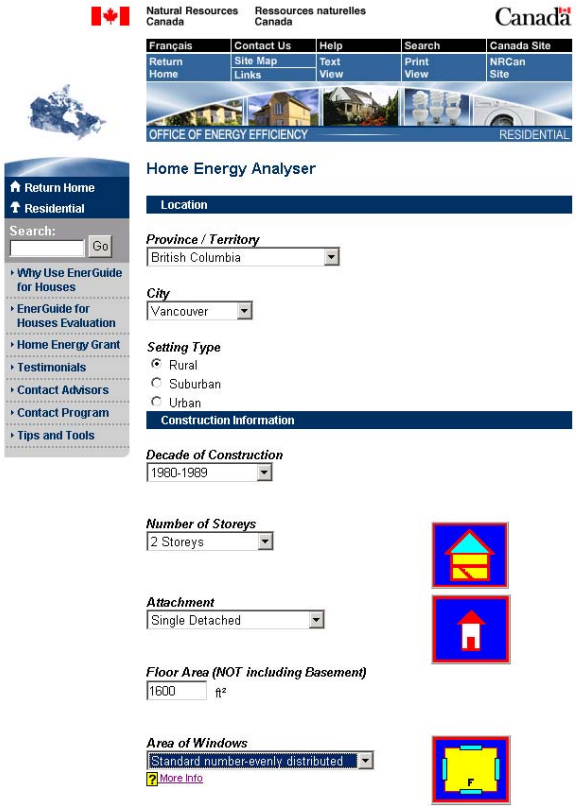

Figure 1: A screen capture of the GUI.

Not only were there over 1000 lines of code that had to be written for the GUI, but there was even more "behind the scenes" coding that had to be written that the user never sees.

Key components of this project that will be discussed are:

- GUI development,
- streaming data to a valid ESP-r/HOT3000 data structure,
- running ESPr/HOT3000,
- extracting the info to be displayed to the user, from the output files, and
- displaying results in an understandable form.

GRAPHICAL USER INTERFACE

The target audience for this project is Canadian homeowners. Within this group, people with a wide range of technical expertise will be using the tool. It was therefore essential that the GUI be simple. This challenged the development team to ensure that the GUI was clear and concise, with unambiguous choices. Visually, the GUI had to conform to the Government of Canada's common look and feel standards. In short, it could not be confusing (see Figure 1 for a partial screen shot of the GUI).

It had to be taken into consideration that the typical homeowner does not know a lot about their house or the technical terms for the components of their house (i.e., insulation type in walls (R-value), what mechanical ventilation is, etc.). Therefore, most of the ESP-r/HOT3000 data was mapped based on certain user inputs (i.e., house location, age of home and age of heating equipment, etc.). These data inputs generated a house archetype otherwise known as a house model, which consequently becomes a representation of the user's home (Purdy 2004).

In addition to the archetype/house model, which compensates for the user's lack of technical knowledge required to run a detailed simulation, the GUI also allows the user to select a "Don't Know" option for certain questions. For example, if a user chooses "Don't Know" for the size of their hot water tank, that value is estimated depending on the size they have entered for their house. The larger the house, the larger the tank is assumed to be.

Extensive logic had to be incorporated into the GUI serving essentially two main purposes. The first was to make the GUI easier to use. Based on the users inputs, other form fields are greyed out or options are added or removed from selection lists. An example of this is when the user selects the "No Basement" option from the dropdown list. The choice for location of their hot water tank would then be limited to "Main Floor" (see Figure 2).

The second reason for incorporating logic in the GUI code is to validate the data before it is sent to the mapping application. For instance, if the user selects Ontario as their province, the drop down list for cities will change to only incorporate cities in Ontario. Not only does this eliminate the possibility for the user to make errors in their selections, but it also guides the user through the difficult process of answering questions relating to their home's thermal description. Another good example of logic that was added to the interface is the question pertaining to foundation type. If the user selects that they have no basement, the next question, "Is your basement heated" is then greyed out and not selectable and defaults to "No" (see Figure 2). Again, this guarantees the user cannot make illogical choices by selecting something that is not actually possible.

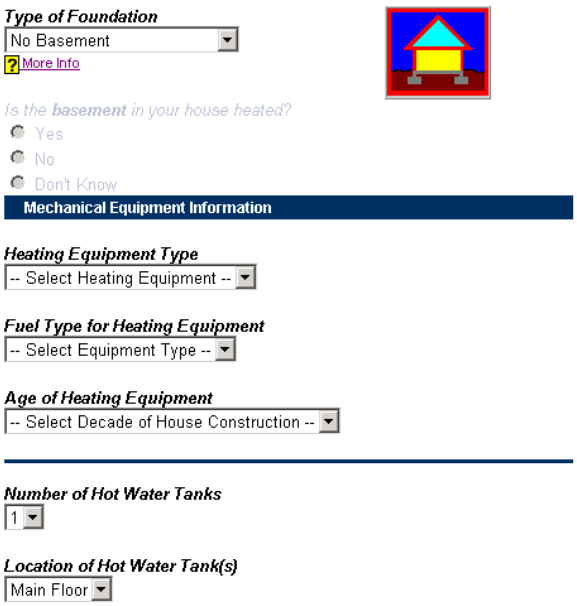

Figure 2: When Foundation type is set to "No Basement", notice that the question "Is the basement in your house heated?" (immediately below it) and all its possible selections are greyed out. As well, the question asking the location of the hot water tank(s) is defaulted to "Main Floor".

Overall, validation of the GUI form makes it easier for the user, and they are not able to submit bad data by accident. Not only is controlling the input data important during the filling-out-the-form stage, but when the user clicks the "RUN" button, a JavaScript runs through the form ensuring that all questions have been answered, and converts all inputs to SI. This is done because the user is allowed to enter the data in whatever units they are comfortable with – SI or Imperial. The JavaScript also checks to ensure that any data typed in by the user is in the proper format (i.e., integers vs. characters).

Another method of making the interface more user friendly and easier for the typical homeowner to understand, are the use of graphics, which relate the user inputs pictorially. For example, the question asking the user the number of storeys in their house, as they scroll through the possible choices, will display the corresponding image to be displayed. This was done for four of the more visual-type questions (see Figure 3).

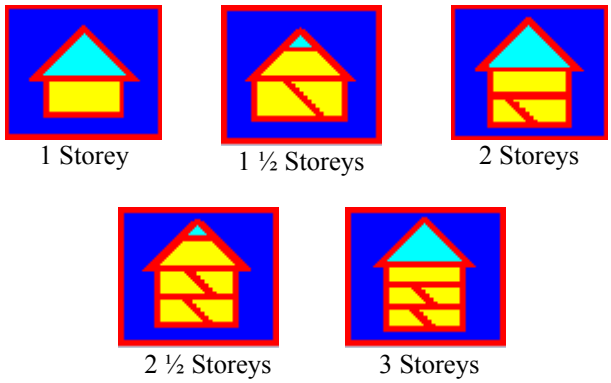

Figure 3: Graphics displayed to user based on their selection for the number of storeys in their home.

One last method of making the GUI easier for the user to understand is the use of help files. Each question that is less than obvious has a link called "More Info" that the user can click on it to read more information about the question and possible options. For example, "House Setting Type" has a detailed description of each of the possible options. The help files aid the user by clarifying terms that could otherwise be misinterpreted subjectively. For example:

> **Rural:** All four exterior walls are bordered 300 metres or more of open space.
> **Suburban:** All four exterior walls are bordered by 50 metres or more of open space.
> **Urban:** At least one exterior wall is bordered by less than 50 meters of open space.

By standardizing terms, it helps to ensure consistent results.

DATA STREAMING

Once the user has completed entering all of their house information, and the GUI has validated it, the data is then loaded into an HTML header file. A PHP Hypertext Pre-processor (PHP) script takes all of this data in the header file and writes it to a flat file, and names it with a unique name. The reason for the unique name is to avoid data collision if there are many users using this application at the same time. The PHP script then calls an application titled "streamtoespr" and passes it the name of the flat file as an argument.

PHP was chosen as the programming language for this project because it is a widely used general-purpose scripting language that is especially suited for Web

development and dynamic Web pages (a web page that interacts with the user).

In order for data in the flat file to be read without misunderstanding by the "streamtoespr" application, a simple format was agreed upon. The naming convention had to be logical; names represent the data fields in the GUI as well as the data names and are most importantly human readable. A few examples of these are:

- PROVINCE
- CITY
- HOUSE_ATTACHMENT_TYPE; and
- FOUNDATION_TYPE

ESP-r/HOT3000 STREAMING

The "streamtoespr" is written in ANSI standard C/C++. This language was chosen for its wide acceptance in the IT world and ease of portability. It can be easily moved to a number of different platforms. These platforms include Windows 2000, Linux and Unix.

The C++ classes used to build "streamtoespr" are grouped into five layers. These layers take care of five distinct functional parts of this application: 1) defining the project's ESP-r/HOT3000 directory structure; 2) parsing the input data (buildings thermal parameters); 3) project configuration; 4) mapping this input data to ESP-r/HOT3000 data structure; and 5) streaming this data to the ESP-r/HOT3000 data directory structure (see Figure 4). This layered approach gives us the freedom of reusing this code in other applications and system configurations. By simply replacing the mapping and data input layers this code can be used to couple the ESP-r/HOT3000 engine to another interface.

The purpose of the "steamtoespr" application is to take the 30 inputs collected by the GUI, translate them into a complete ESP-r/HOT3000 dataset then write the dataset to a valid ESP-r/HOT3000 directory structure, all in preparation for an ESP-r/HOT3000 simulation run. Some of the data mapping is accomplished through algorithms, some through extracting data from a table using the house age and location, which in turn produces the house model.

The house model, including all the data required to perform an ESP-r/HOT3000 simulation, is the output of the "streamtoespr" application

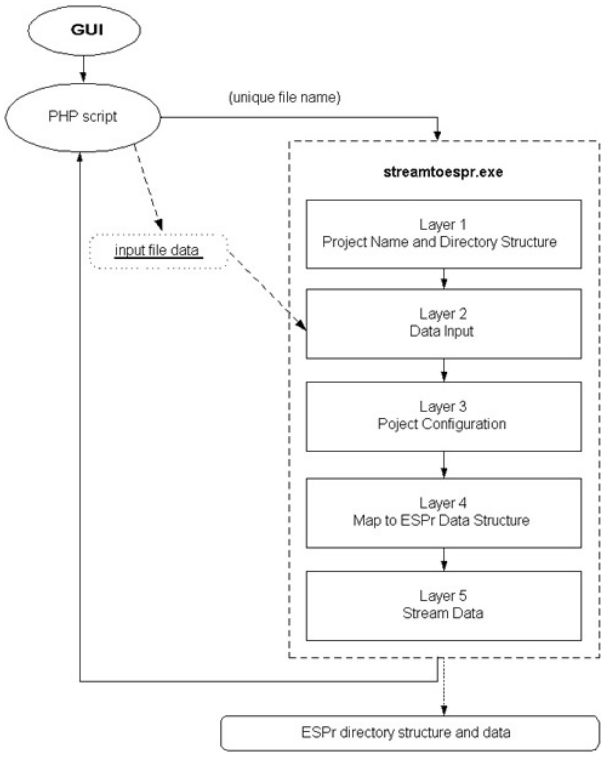

Figure 4: Graphical representation of the five layers.

Streamtoespr Structure:

As mentioned above, this application is structured in five distinct layers. They are the:

- project definition layer,
- data input layer,
- project configuration layer,
- mapping layer, and
- streaming layer.

The following sections detail these layers.

Project Name and Directory Structure Definition Layer (Start-up):

The project definition layer takes the unique file name passed as an argument from the PHP script, and passes it to the "streamtoespr" application. This file's name is used to construct the ESP-r/HOT3000 directory structure and all the project files. Where appropriate the project's name is used as the file name (i.e., ExampleHouse1.cfg). This .cfg file is located in the "user/E*xampleHouse1*/cfg/" folder. The files, which do not use "ExampleHouse1" as the file name, are *main*, *attic* and *basement* located in the "user/E*xampleHouse1*/zones/" folder. There is no conflict with having files with the same names because

each project is contained in the main project directory. For example:

- ExampleHouse1/zones/basement.con
- ExampleHouse2/zones/basement.con
- ExampleHouse3/zones/basement.con

Data Input Layer:

The "streamtoespr" application takes the project file name that is passed to it as an argument and opens the input data flat file, created by the PHP script. This file contains 30 data items, which describe this project's residential building's thermal performance and location.

The format of the data file is two items per line (token + data). This type of file format was chosen as it allows the inclusion or exclusion of data items if desired, without a large scale rewriting of the parsing code. This is because the "token + data" format means that order is not important when reading this data. All the data is read and parsed until no data items are left. The independent data items are placed in ESP-r/HOT3000 variables at this stage, such as climate data, domestic hot water tank volume and building floor area. The dependent data items are mapped only after all input data is read and parsed.

Project Configuration Definition Layer:

Almost all the data required for the configuration file is set in this layer except for the number of zones which are defined in the mapping layer. The first data items to be set are the configuration (.cfg) file's parameters describing which files are to be included in this project.

An ESP-r simulation run can take anywhere from thirty seconds to an hour. In order to improve, what could be a very long response time, it was decided to set the simulation periods to five seasonal periods of seven days each. ESP-r/HOT3000 analysing each location's weather data chose these seasonal dates, and it generated five average representative week periods.

For example, the peak-heating season in Calgary is different than that of Halifax. The following is an example of the simulation dates for a building located in Churchill, Manitoba:

- EARLY_WINTER (Mar 4 to Mar 10)
- SPRING (Apr 26 to May 2)
- SUMMER (Aug 31 to Sept 6)
- FALL (Nov 17 to Nov 23)
- LATE_WINTER (Dec 22 to Dec 28)

Paths are set for each ESP-r/HOT3000 file type and the ESP-r/HOT3000 files are named with the unique project name.

Mapping Layer:

This layer takes the dependent input parameters and maps them to ESP-r/HOT3000 data, which is needed for all the remaining file types. Mapping this data begins with the building description. This process first starts with defining the building's origin, its dimensions, window sizes, the number of zones, HVAC system and temperature controls. Once the number of zones has been defined, the configuration data (.cfg file) is complete.

There are three different building configurations that are used to define the building geometry. These three are:

- single house
- row house - middle unit
- row house - end unit

Windows are defined according to the window parameter (large or small, evenly distributed or mostly front and back).

Using vertices created from the building geometry, all surfaces are defined for all thermal zones included in the project. The surface construction is mapped along with the surface definition. Temperature controls are also mapped and the heating/cooling systems are defined. This completes the building definition and the data is now ready for streaming.

For a detailed description of mapping algorithms, refer to Purdy (2004).

Streaming Layer:

The streaming layer is constructed with a set of C++ classes, defined with a one to one correspondence to each ESP-r/HOT3000 data file type. For example:

Class "CEspr**Cfg**Doc" → File "ExampleHouse1.**cfg**"
Class "CEspr**Geo**Doc" → File "ExampleHouse1.**geo**"

In the "streamtoespr" application, all ESP-r/HOT3000 file streaming is controlled through the configuration class "CEsprCfgDoc" and subsequently the function WriteCfgFile(). The logic for including and excluding files is contained in this function as discussed in the project configuration definition layer.

The standard ESP-r/HOT3000 file structure contains nine folders and at least eleven files contained in these folders. The files contain a detailed numerical

description of the building, from the control system to the materials that the building is constructed from. Once all the ESP-r/HOT3000 files are created, "streamtoespr" exits, returning control to the PHP script, in preparation of the simulation application (bpsH3K) launch.

GENERATING RESULTS

Once the "streamtoespr" application has returned control to the PHP script, PHP then launches ESP-r/HOT3000 (bpsh3k). Once bpsh3k has completed its simulation, the PHP script then checks to ensure the ESP-r/HOT3000 results file is created, proving that it ran successfully.

The PHP script then extracts the domestic hot water, heating and cooling loads from each of the five seasonal files. Then heating and cooling energy demands for the week are multiplied by heating and cooling multipliers to get the energy demand for the whole simulation period of interest. The heating multiplier is equal to the ratio of the number of degree-days for the whole period to the number of degree-days for the typical week. A similar approach is used to get the cooling multiplier.

These values are the "actual" energy consumptions for the house. The house "potential" is the energy consumption level the user could potentially obtain and is derived using the actual numbers. The potential is based on location of house, age of house, etc. Using these six numbers (Heating, Cooling, and Hot Water for the Actual house and Potential house), a graph is produced (see Figure 5) for the user so they can see how their house compares to other energy efficient houses in their area.

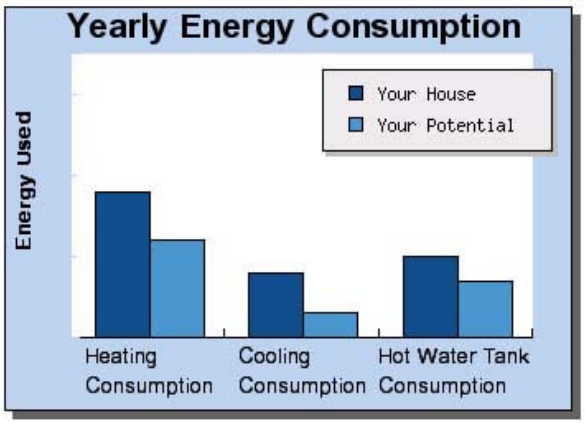

Figure 5: Example graph of what user will see for their heating, cooling and hot water tank consumption.

Once this is done, the PHP script removes the ESP-r/HOT3000 directory structure, and deletes all files associated with that particular ESP-r/HOT3000 simulation.

The PHP script then produces the HTML Results Page and based on the original inputs from the user, i.e., fuel type for their furnace, determines which information will be printed on the Results Page. For example, if the user heats their home with oil, the results page will refer them to on-line publications about heating with oil. If the user heats their home with wood, they would be referred to publications on maintenance of wood burning stoves and possibly publications on converting from wood to a more standard or energy efficient method of heating. The Office of Energy Efficiency chose which text to display to the user, based on publications they have. See Figure 6 for a partial screen capture of Results Page.

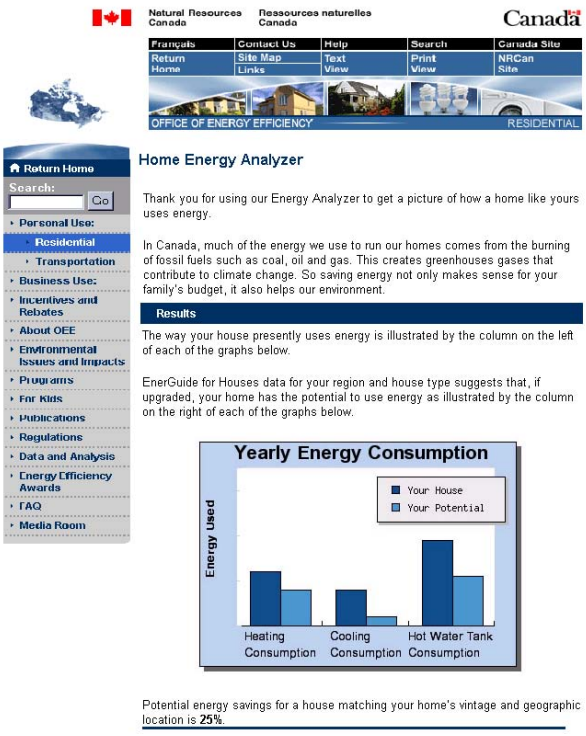

Figure 6: Partial screen shot of Results Page.

The purpose of the Results Page is to provide the user with information regarding their energy use. Suggestions are also made as to how they can modify their energy use to be more energy efficient.

The results displayed are in a comparative bar graph format rather than a numerical format. By simplifying the inputs it is to be expected that the results will also be simplified. Users are asked to contact EnerGuide for Houses, a Government program for helping homeowners retrofit their home, and they will perform a more detailed simulation for them.

The overall approach (web based GUI) may seem small compared to what ESP-r is really capable of, however a detailed simulation is run, and only a limited amount of results are displayed to the user. If in the future CETC were to expand this project, more inputs could easily be requested, more results could be shown and perhaps more detailed building types could be modeled. All the stepping-stones are in place for this project to grow. At this time it is a marketing tool for the EnerGuide for Houses Project, and not meant to be overly detailed.

FUTURE ENHANCMENTS

Some future enhancements that CETC intends to include in the Home Energy Analyser are: 1) simulating other residential building types, such as duplexes, triplexes, walk-ups, etc; 2) allowing the user to run a second simulation so they can compare the energy consumption of their house with and without upgrades (i.e., upgrading windows); and 3) adding an option to the GUI where the user can enter the number of appliances and/or their frequency of use (i.e., 1 fridge vs. 3 fridges).

Additional future enhancements will also take into consideration the suggestions and future requirements of users.

CONCLUSIONS

Through the use of a user-friendly web interface, coupled with a detailed program for mapping the data to the ESP-r/HOT3000 structure, CETC made available to all Canadians, one of the premiere and most advanced energy analysis tools on the market. CETC helps the common Canadian homeowner in a simplified approach to help them analyse their energy usage. This means you don't have to be knowledgeable in the building technologies field to analyse your home's energy consumption.

ACKNOWLEDGEMENTS

We would like to acknowledge Natural Resource's Office of Energy Efficiently for funding this project and providing us with the server space for the web pages.

REFERENCES

Purdy, Julia, *The Development Of A Dynamic On-Line Whole-Building Energy Analysis Tool For Homeowners*, eSim 2004.

ESRU, ESRU publications, ESRU website, January 2004, http://www.esru.strath.ac.uk/publications.htm

ESRU (2000), *The ESP-r building system for Building Energy Simulations*: User Guide Version 9 Series, ESRU Manual U00/1, University of Strathclyde, Glasgow UK.

Clarke J. A. (2001a), Energy Simulation in Building Design, (2nd ed.), Butterworth Heinemann.

Clarke J. A. (2001b), "Domain Integration in Building Simulation", Energy and Buildings, 33 303-308