A Web-Based System to Gather and Distribute Specifications for Wind Tunnel Tests

Stephen B. Jenkins*
Institute for Aerospace Research, Ottawa, ON, K1A 0R6, Canada

A web-based system to gather and distribute specifications for wind tunnel tests has been created at the Aerodynamics Laboratory of the Institute for Aerospace Research, National Research Council of Canada. The test specification program has been written to provide a simple method of collecting and disseminating the information required by the technical support staff to properly install, configure, and conduct wind tunnel tests. Relying on Open Source Software (Linux, Apache, and Perl) for its implementation, this application takes advantage of three common technologies (the Web, email, and PDF files) to create a tool that is both powerful and easy to use.

I. Introduction

A web-based system to gather and distribute specifications for wind tunnel tests has been created at the Aerodynamics Laboratory (AL) of the Institute for Aerospace Research, National Research Council of Canada. This web-based questionnaire has been implemented as part of an on-going effort to web-enable all of the information processing activities associated with performing tests in the Laboratory facilities. Over the past several years, this effort has resulted in the creation of an ensemble of web- and net-based programs designed to work in a heterogeneous, distributed environment. Known collectively as the Internet Software Tools for Aerospace Research Installations, the ISTARI system currently consists of five primary applications (plotting, data file viewing, configuration management, event logging, and real-time data display), nine secondary applications (of which this test specification system is one), and a large number of web pages containing status information, documentation, and links to other on-line content.[1-3]

The test specification program, *testspeccgi.pl*, has been written to provide a simple method of collecting and disseminating the information required by the technical support staff at the facilities of the AL to properly install, configure, and conduct wind tunnel tests. Designed to be used by either the Laboratory test supervisor or a client designate, this software tool is accessible, via a web browser, from any networked computer that has been granted the appropriate access rights to the ISTARI system. Before the implementation of this application, test specification information had been collected and distributed in a rather ad-hoc manner, often leading to confusion and unnecessary delays during the setup and execution of wind tunnel tests.

II. Functional Description

The test specification form itself currently consists of 332 individual query elements divided into 21 separate sections with headings such as *Client Information*, *Model information*, *Data Reduction*, *Test Conditions*, *Shop Services*, *Strain-Gauge Measurements*, and *Flow Visualization*. After selecting the link for the test specification program from the ISTARI home page, the user is presented with an empty form, the top portion of which is shown in Fig. 1. The two buttons near the top-left corner have been provided to help the user by opening additional browser windows, one with general instructions describing how the form is to be used, and the other with a calculator to assist in the conversion of units.

Rather than overwhelm the user, and his computer and browser, with all 332 query elements, many parts of the form have been hidden, initially, with only a single checkbox element showing. The existence of each hidden area is indicated by the presence of a bright blue background around its controlling checkbox. When the user selects such a checkbox, in order to affirm the need to provide information related to its subject, the hidden area is automatically

* Senior Programmer/Analyst, Aerodynamics Laboratory.

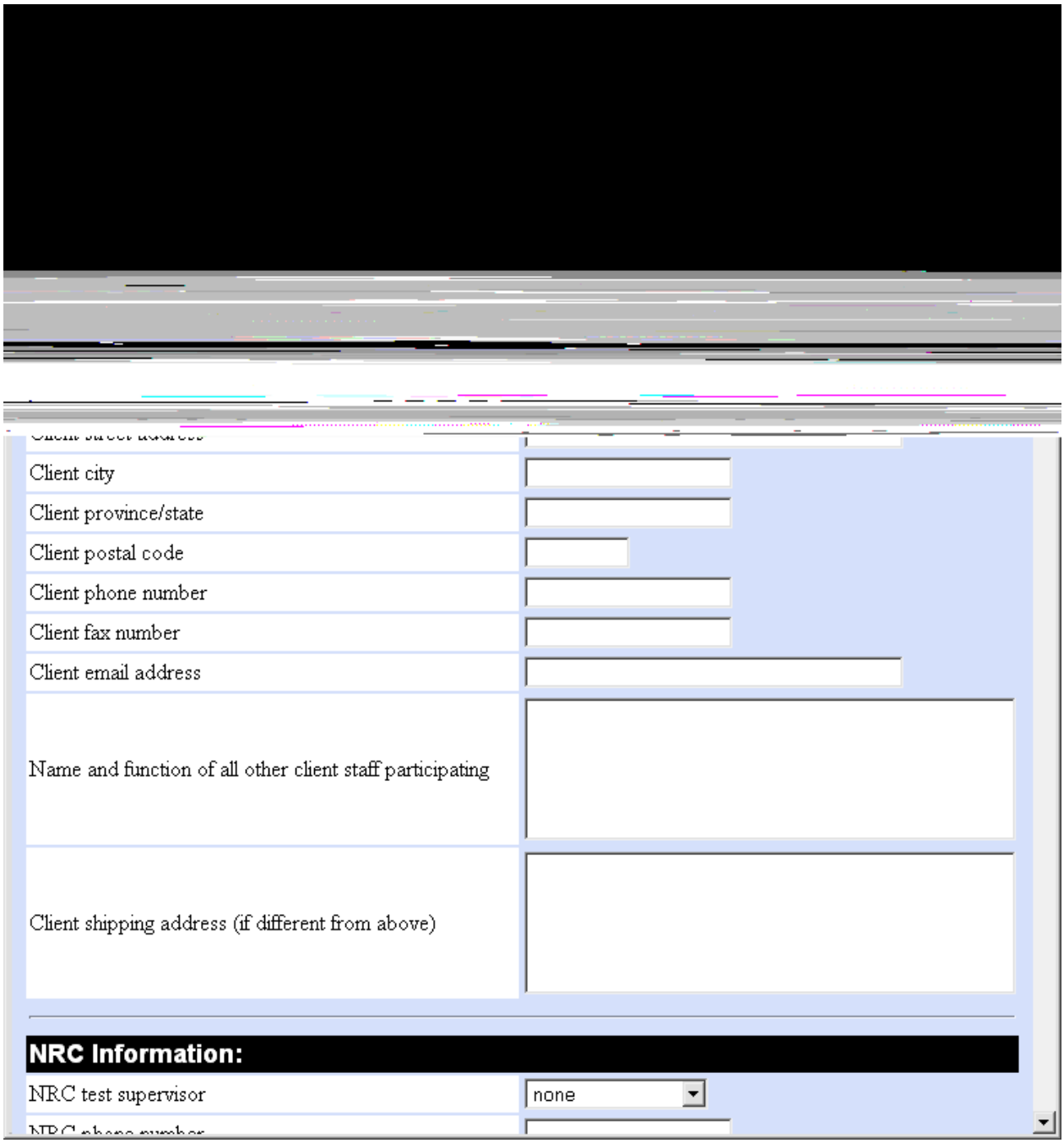
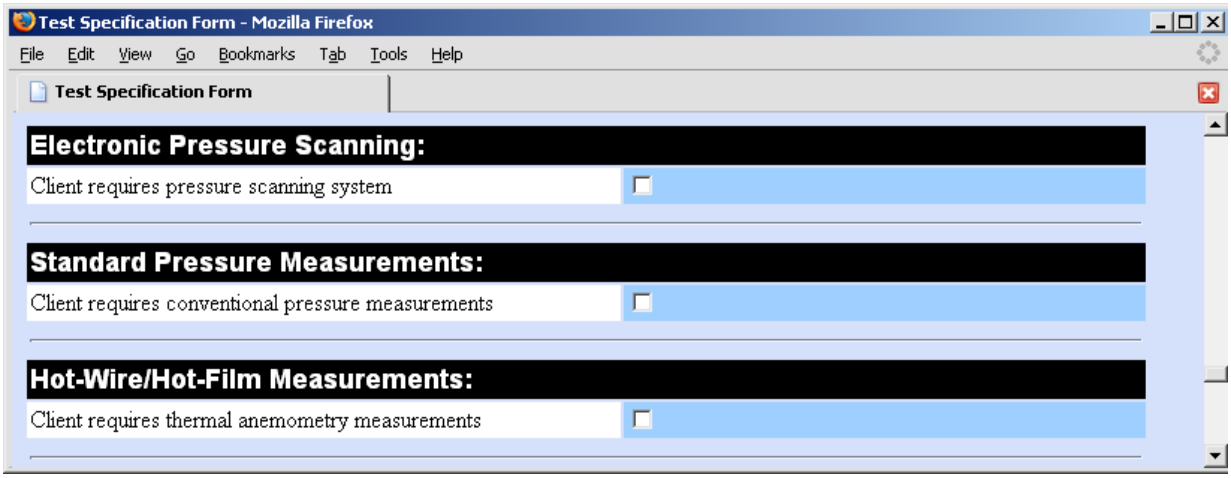


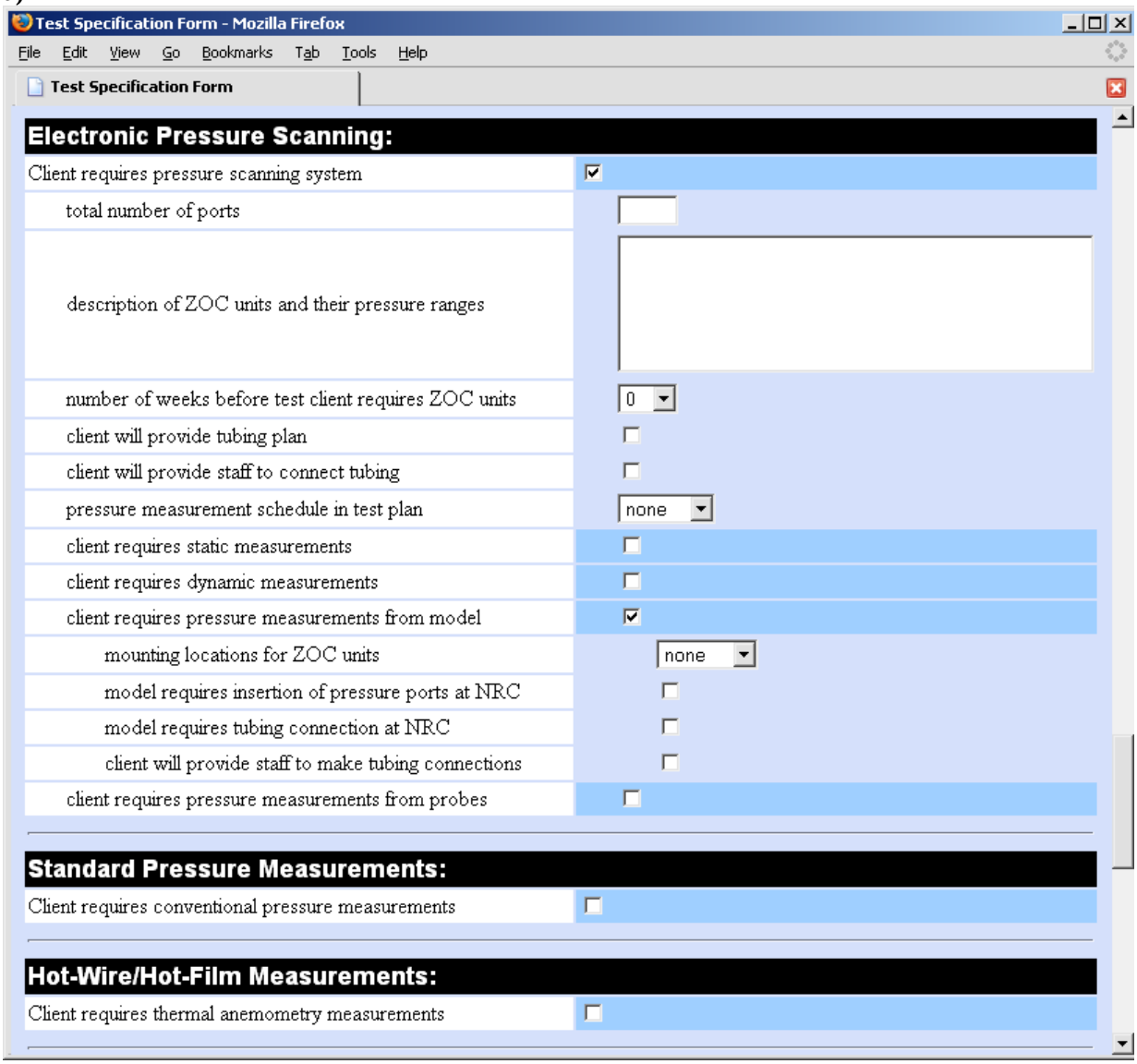
Figure 1. Top of form showing help buttons.

expanded. This may be seen in Fig. 2, where the *Electronic Pressure Scanning* section is shown before and after expansion. Note that hidden areas may contain further hidden areas, in a hierarchical fashion, and that that hierarchy is reflected in the indentation used when displaying the input elements.

At the bottom of the form (shown in Fig. 3), four buttons have been provided to enable the user to manipulate the files containing the test information. By allowing the specification files to be saved and re-opened, details can be provided over multiple input sessions. Although the function of the *Save* button is rather self evident, the other three options require further explanation.



a)



b)

Figure 2. Electronic Pressure Scanning section shown (a) collapsed and (b) expanded.

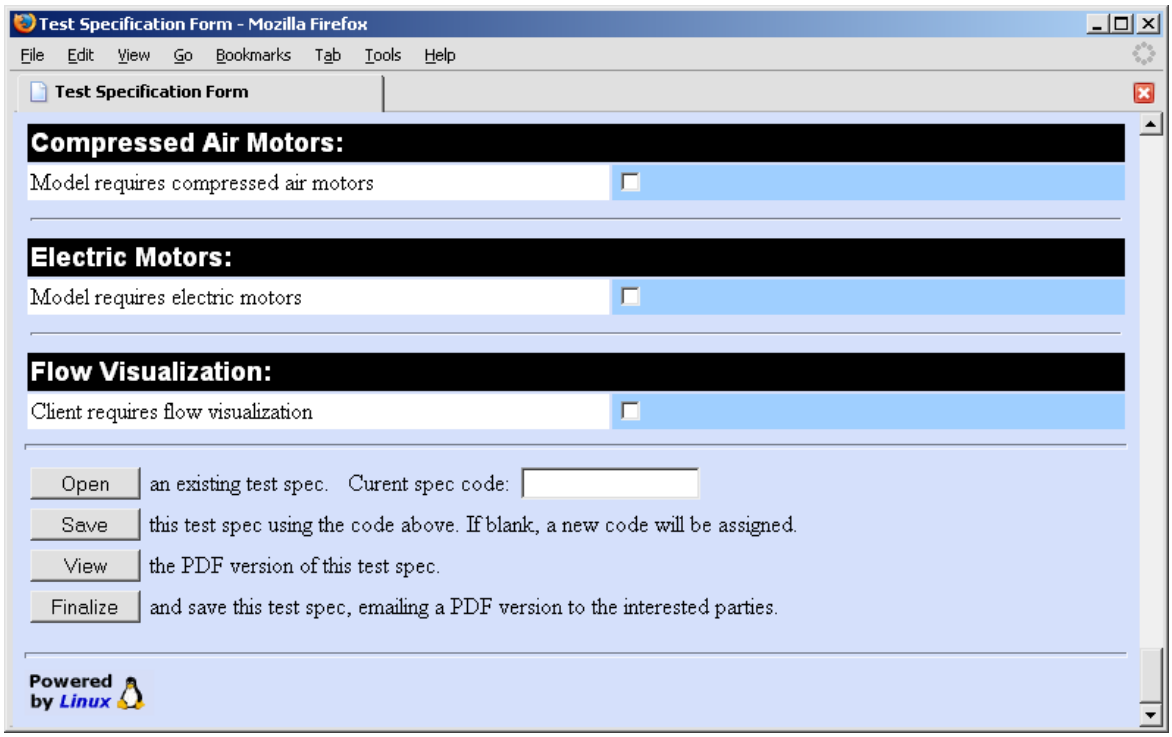


Figure 3. Bottom of form showing file handling elements.

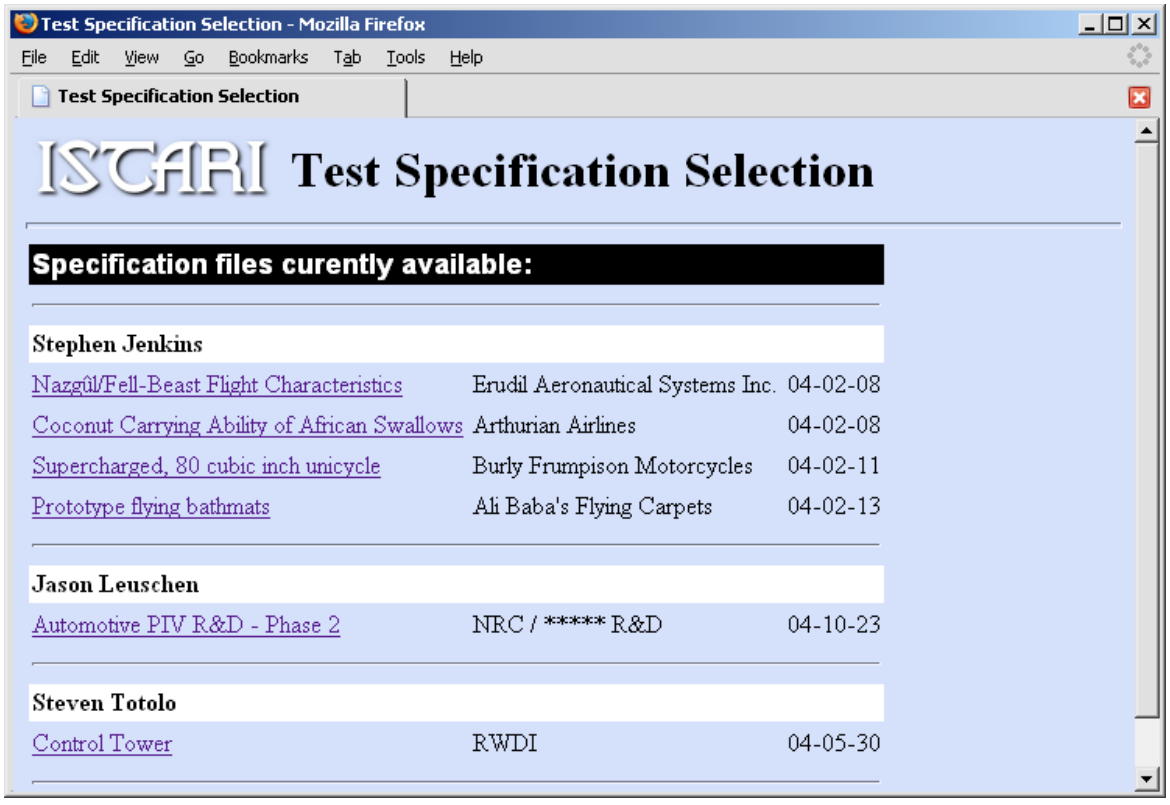


Figure 4. File open selection page.

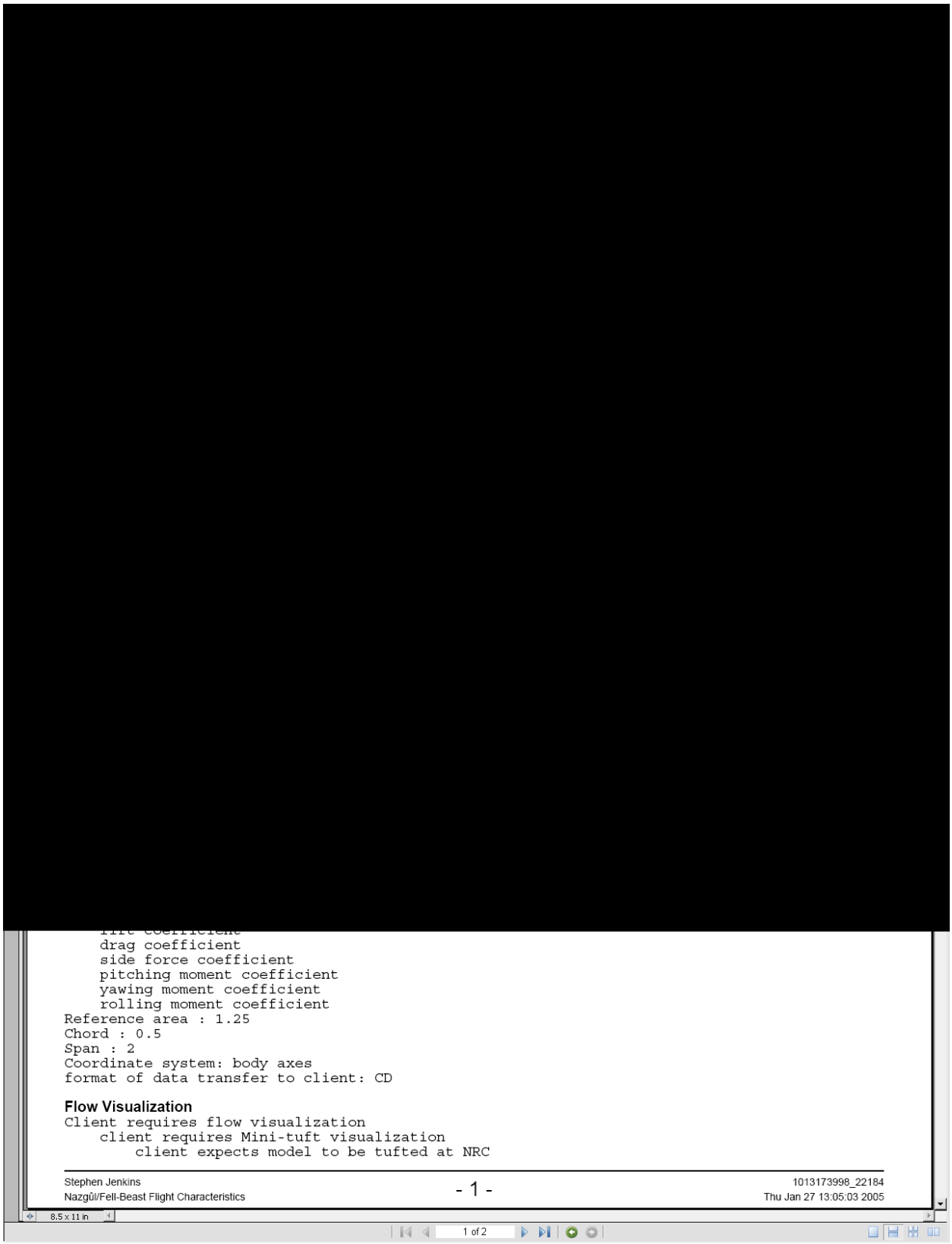


Figure 5. First page of PDF output file

Selecting the *Open* button causes a Test Specification Selection web page (similar to that shown in Fig. 4) to be loaded into the user's browser. The test name and client name strings for all available files are displayed, sorted by the test supervisor's name and the creation date. To open a file, the user has only to click on the test name string.

The *View* button submits the form back to the program, where its information is saved, then used to create the final output: a Portable Document Format (PDF) file. That file is then returned to the user's browser for viewing; Fig. 5 shows the first page of a typical example. Note that unused fields are not printed (resulting in the most concise document possible), and that fonts and white space have been used to increase readability. In addition, headers and footers have been added to each page to display a title, the page number, the test supervisor's name, the test name, and the creation time of the PDF file.

When the user is satisfied that all information for the test in question has been recorded, the *Finalize* button can be used. In addition to the form being saved and its contents converted to a PDF file, as happens with the *View* button, copies of the output are sent, via email, to a number of interested parties. The list of email address is contained within the program, and includes all potential test supervisors, the facility manager, and the heads of the software, instrumentation, and operations support groups. If, after the decision to finalize a test, changes are required to be made, the specification may be opened, updated and then re-finalized in order to inform everyone concerned.

III. Implementation

The following sections describe some of the implementation details of the test specification system. While written for software developers, and expected to be of interest primarily to that audience, the information may also be of value to others.

A. Design Decisions

In keeping with a software development policy that was adopted over eight years ago, the test specification application has been created with, and is dependent on Open Source Software.[4] The program has been written, primarily, using the Perl[5] programming language and makes use of several modules from the Comprehensive Perl Archive Network (CPAN). They include the web server interface module *CGI.pm*, a module to help produce PDF files, *PDF::Create*, and a module to dynamically generate and send email messages with attachments, *MIME::Lite.* Small portions of code that run within the user's browser have been written in JavaScript[6], and the final presentation of course relies on the Hypertext Markup Language (HTML) with Cascading Style Sheets (CSS).[7,8] The PDF was selected as the final output format because it has become the standard for sharing printable documents across all platforms and operating systems, and as such is both familiar to all and simple to use. The interaction with the user is handled by an Apache web server, and both that server and the test specification program itself execute on an Intel based PC running the Linux operating system.

In order to provide the simplest and most consistent interface for the user, it was decided to design the form with only one query element per line, and only allow four types of elements. Those types are, in increasing order of complexity for the user: checkboxes; popup menus; text boxes; and text areas. To reduce the possibility of input errors, the preferred input type for each query is, in general, the least complex one that allows for all input possibilities. For example, for a query with a small number of possible responses, a popup menu is to be preferred over a text box, since selecting an item from a list is far less prone to errors, and far quicker, than is typing a response.

In order to provide the simplest and most consistent information to the technical support staff responsible for the implementation of the test specifications, all input prompt strings have been phrased such that the default value will result in no action. In terms slightly more blunt: "If you don't fill it in, it won't be done." Of the 332 input elements, only three (the test supervisor's name, the client company's name, and the test name), are required to have values, and these only because they are necessary for the form to be saved and retrieved.

B. File Handling

The perspicacious reader will have noticed that there was no mention of filenames or paths in the "Functional Description" section. This is because all file handling details have been hidden, deliberately, from the user. A unique "spec code" is generated the first time each test specification is saved and it is used in the creation of the filename. All end-user file interaction is handled only via the information shown on the Test Specification Selection form (Fig. 4). By removing the need for users to provide and remember file names and locations, a significant area of potential error and confusion has been eliminated.

```
Electronic Pressure Scanning:
tag=EPSSREQ  type=checkbox  desc="Client requires pressure scanning system"
tag=EPSSNPRT type=textfield desc="total number of ports" size=3 max=4 showif=EPSSREQ
tag=EPSSZOCS type=textarea  desc="description of ZOC units ... " cols=40 rows=5 showif=EPSSREQ
tag=EPSSNWKS type=popup     desc="number of weeks before ... " opt="0,1,2,3,4,>4" showif=EPSSREQ
tag=EPSSPLAN type=checkbox  desc="client will provide tubing plan" showif=EPSSREQ
tag=EPSSSTFF type=checkbox  desc="client will provide staff to connect tubing" showif=EPSSREQ
tag=EPSSSCHD type=popup     desc="...schedule..." opt="none,start,middle,end,many" showif=EPSSREQ
tag=EPSSSTAT type=checkbox  desc="client requires static measurements" showif=EPSSREQ
tag=EPSSSNUM type=textfield desc="total number of channels" size=6 max=6 showif=EPSSREQ&EPSSSTAT
tag=EPSSSPER type=textfield desc="sample period (Sec.)" size=6 max=6 showif=EPSSREQ&EPSSSTAT
tag=EPSSSAVG type=textfield desc="average over (# of ... " size=6 max=6 showif=EPSSREQ&EPSSSTAT
tag=EPSSDYN  type=checkbox  desc="client requires dynamic measurements" showif=EPSSREQ
tag=EPSSDNUM type=textfield desc="total number of channels" size=6 max=6 showif=EPSSREQ&EPSSDYN
tag=EPSSDPER type=textfield desc="sample period (Sec.)" size=6 max=6 showif=EPSSREQ&EPSSDYN
tag=EPSSDFRQ type=textfield desc="frequency (Hz.)" size=6 max=6 showif=EPSSREQ&EPSSDYN
tag=EPSSDRES type=checkbox  desc="client requires NRC to measure ... " showif=EPSSREQ&EPSSDYN
tag=EPSSMOD  type=checkbox  desc="client requires pressure measurements from ..." showif=EPSSREQ
tag=EPSSMODZ type=popup     desc="mounting ..." opt="none,inside, ..." showif=EPSSREQ&EPSSMOD
tag=EPSSMODI type=checkbox  desc="model requires insertion of ... " showif=EPSSREQ&EPSSMOD
tag=EPSSMODT type=checkbox  desc="model requires tubing connection at NRC" showif=EPSSREQ&EPSSMOD
tag=EPSSMODS type=checkbox  desc="client will provide staff to make ..." showif=EPSSREQ&EPSSMOD
tag=EPSSPRB  type=checkbox  desc="client requires pressure measurements from ..." showif=EPSSREQ
#tag=EPSSNHDS type=textfield desc="total number of heads" size=5 max=80 showif=EPSSREQ
```

Listing 1. A small portion of the program's data segment.

C. Questionnaire Data

Although both XML format and the use of a database were considered, briefly, to store the questionnaire itself, it was decided to implement a traditional, single line per element format, containing a named parameter list. This way the question data could be appended directly to the end of the program file, making the system simpler and easier to maintain. Listing 1 contains a small portion of the program's data segment (edited to reduce line lengths) that corresponds to the *Electronic Pressure Scanning* section of the form shown in Fig. 2. This particular portion was chosen for presentation because the first four queries contain one of each of the four element types. Since the data segment is arguably the most important part of the test specification system – from the point of view of implementation at least – it will be the focus of the discussion for the next several paragraphs.

The first line of the listing shows a section heading, identified by its trailing full colon. The program displays this line as white text on a black background. The remaining lines in Listing 1 represent test specification query elements, and are composed of several *key=value* pairs. The three keys that are required to be in every line for it to be well formed, are *tag*, *type*, and *desc*. The value for the *tag* key is expected to be a unique alphanumeric string that will be used to identify the query element (essentially, its "name"). The *type* key is expected to be set to one of the four element types: *checkbox, popup, textfield*, or *textarea*. The value for the *desc* key will be used as the prompt string that describes to the user what information is expected to be provided.

In addition to the three required keys, there are several additional *key=value* pairs that are available to the data segment maintainer, depending on the element type. For *popup* elements, an *opt* key is required to provide the list of options that will be shown in the popup menu. For *textfield* input elements, there are two non-compulsory keys, *size* and *max*, that specify, respectively, the width of the input field in characters, and the maximum number of characters allowed. For *textarea* elements, there are also two optional keys, *rows* and *cols*, which may be used to specify the size of the text box entry field.

Any input element may also have an optional *showif* key-value pair, to make that element's visibility dependent on the state of one or more checkboxes. The program will "show" the element in question if all of the checkboxes named in the value portion, are "on." The ability to specify multiple checkbox dependencies allows for the hierarchical system mentioned in the *Functional Description* section. The values of the *showif* keys are also used by the program to determine which checkbox elements require the modified background colour that informs the user that there are hidden elements. The last line of Listing 1 shows an element that has been commented out, through the use of a leading pound character; the program will thus ignore that line.

The portion of the code that reads and parses the data segment is also responsible for ensuring that it is valid and well formed. If data lines cannot be parsed properly, if any of the required *key=value* pairs are missing, or if *tag* values are found to be duplicated, that information is appended to the questionnaire web page, alerting the author/maintainer of the data segment to the existence of the error.

IV. Concluding Remarks

Because it employs the web for its user interface, the test specification system is not only easy to use and familiar, (nearly everyone has filled in a web-form at some time or other), but it also automatically inherits all the benefits of web-based applications: it is easy to maintain; easy to develop; and, by definition, easy to distribute.

Because it is written in Perl and uses a data segment for the query information, the test specification system is easy to modify: there is no compilation required. Questions and options may be added, removed, or revised, with little knowledge of the program's internals, and without causing compatibility problems with previously saved specification files.

Because it gathers all of the information necessary to install, configure, and conduct a wind tunnel test, and sends all interested parties an identical PDF file containing that information, the test specification system ensures a comprehensive and consistent transfer of knowledge about the test requirements.

While the system has been discussed purely in terms of a test specification tool, the software is easily modified for any information gathering task that lends itself to a form-based approach. Merely by changing the data segment information, this system could be used for things such as client satisfaction surveys, resource request forms, and project status reports. The three standard technologies used by this system – the Web, email, and PDF files – combine in a synergistic way that enables a relatively simple application to become an extremely powerful information gathering and distribution software tool, that is also easy to use.

References

[1]Jenkins, S. B., "Using Perl to Create Web-Based Software Tools for Wind Tunnel Testing," AIAA 2001-0905, 39th Aerospace Sciences Meeting & Exhibit, Reno, NV, Jan. 2001, URL: http://www.erudil.com/pdf/reno2001.pdf [cited 2005-09-15]

[2]Jenkins, S. B., "A Web-Based Environment to Support Aerodynamic Testing," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 19, No. 1, Jan. 2004, pp. 3-11, URL: http://www.erudil.com/pdf/aess2004.pdf [cited 2005-09-15]

[3]Jenkins, S. B., "An Overview of the ISTARI System," Institute for Aerospace Research, LM-AL-2004-0072, National Research Council, Sept. 2004.

[4]Jenkins, S. B., "Open Source Software at the Aerodynamics Laboratory," *Linux Journal*, Issue #90, Oct. 2001, pp. 70-74, URL: http://www.linuxjournal.com/article/4835 [cited 2005-09-15]

[5]Wall, L., Christiansen, T., and Orwant, J., *Programming Perl*, 3rd ed., O'Reilly & Associates, 2000.

[6]Flanagan, D., *JavaScript: The Definitive Guide*, 4th ed., O'Reilly & Associates, 2002.

[7]Musciano, C. and Kennedy, B., *HTML & XHTML: The Definitive Guide*, 4th ed., O'Reilly & Associates, 2000.

[8]Meyer, E., *Cascading Style Sheets: The Definitive Guide*, 2nd ed., O'Reilly & Associates, 2004.