

# Conversion of the BlueSky Framework into collaborative web service architecture and creation of a smoke modeling application



**FINAL REPORT TO THE JOINT FIRE SCIENCE PROGRAM**  
September 30, 2009

Project #: **08-S-07**  
Website: **<http://blueskyframework.org/ws>**

PI: **Dr. Narasimhan K. Larkin**  
**U.S. Forest Service AirFire Team**  
**Pacific Northwest Research Station**  
**400 N. 34<sup>th</sup> St Suite 201**  
**Seattle, Washington USA 98103**  
**206.732.7849 (W) 206-732.7801 (F) [larkin@fs.fed.us](mailto:larkin@fs.fed.us) (E)**

Co-PI(s): **Sean Raffuse, Sonoma Technology, Inc.**

Co-authors: **Daniel Pryden, Alan Healy, Kevin Unger of Sonoma Technology, Inc.; Dr. Tara Strand, Dr. Robert Solomon of the U.S. Forest Service AirFire Team.**



This project is was funded through grant 08-S-07 from the Joint Fire Sciences Program (<http://firescience.gov>).

## Abstract

This project addresses the need for a collaborative architecture for scientific modeling that allows various scientific models to easily interact. The need for such a system has been documented by recent studies such as the JFSP Smoke Roundtables and the JFSP review of tools done by the Software Engineering Institute. This project addresses these needs by modifying the BlueSky Modeling Framework so that it can better serve as a collaborative architecture, and then utilizing this architecture to create an advanced application that could not otherwise be created.

The BlueSky framework was modified for this purpose, and all changes integrated into all versions of BlueSky from 3.1.0 forward. BlueSky now contains a command line option that will automatically start it as a web-service provider, allowing it to be used by remote clients. When the web-service option is used, all models contained within BlueSky are automatically converted into web-service accessible modules, without need for a specialized web-service enabled version. Simple examples and documentation scripts designed to show a website or user interface creator how to access these models via web-service function calls were created. In addition, a more advanced website interface was created to show some of the advantages of web-service based scientific modeling. This tool, called BlueSky Playground, provides a single user interface into 10 models of fuels, consumption, emissions, plume rise, and smoke dispersion. A user can walk step-by-step through all of the model steps in the framework from fire information to smoke impact maps. At each step the user can choose the model they want to use and alter the modeled information before continuing on, allowing for a game-playing exploratory mode of interaction. Both the ability to access so many models through a single interface as well as the capability to obtain on-the-fly smoke dispersion calculations are novel to this tool. This application will be highlighted in 2010 through RX-410 classes as a way for users to learn about the various component models. It will also serve as a training tool for managers needing to run multiple scenarios and understand the implications of various choices.

The web-service oriented architecture utilized in the project offers many potential advantages to scientific research done with the goal of decision support. Separation of the scientific computing portion of such work from the user interface allows scientists to focus on creating the best models and web designers to focus on creating the best interfaces. Remote functioning of the models through the web means that local installation of the model is no longer required solving distribution issues, and allows an Internet user to run a model that requires resources not available to them locally (such as large datasets or fast processors). Modularity allows for “mash-ups” where models are combined in ways not originally foreseen to meet emerging needs, and provides choices to be made on exact modeling pathways at the user or institutional level.

## Background

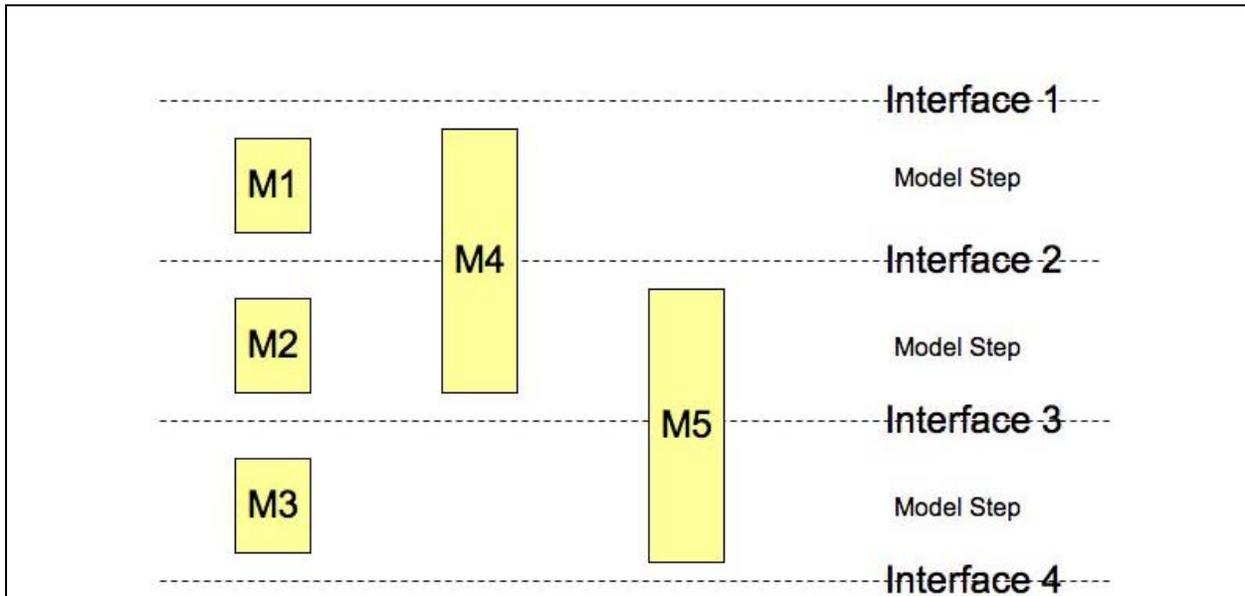
Several major efforts to organize fire and smoke research have been undertaken recently by the Joint Fire Science Program. The Smoke Roundtables (JFSP, 2007) have pointed out the large number of user interfaces developed in conjunction with various scientific models, and the

problems associated with the lack of compatibility between the application systems that has resulted. The Carnegie-Mellon based Software Engineering Institute's (SEI) review of JFSP tools has pointed out the need for a collaborative scientific modeling architecture to make currently diverse models inter-operable (Palmquist, 2008). Specific advantages of such a collaborative architecture are:

1. To separate the development of scientific models from user interfaces (UIs);
2. To allow for integrated UIs capable of driving multiple models;
3. To allow for faster development of models and UIs;
4. To allow for direct comparison between models; and
5. To allow for faster transition between developed models and operational applications.

This project has produced an example collaborative scientific modeling architecture, and highlighted the advantages of such a system through the creation of a unique game-playing application through modifications to the BlueSky Modeling Framework (Larkin et al, 2009).

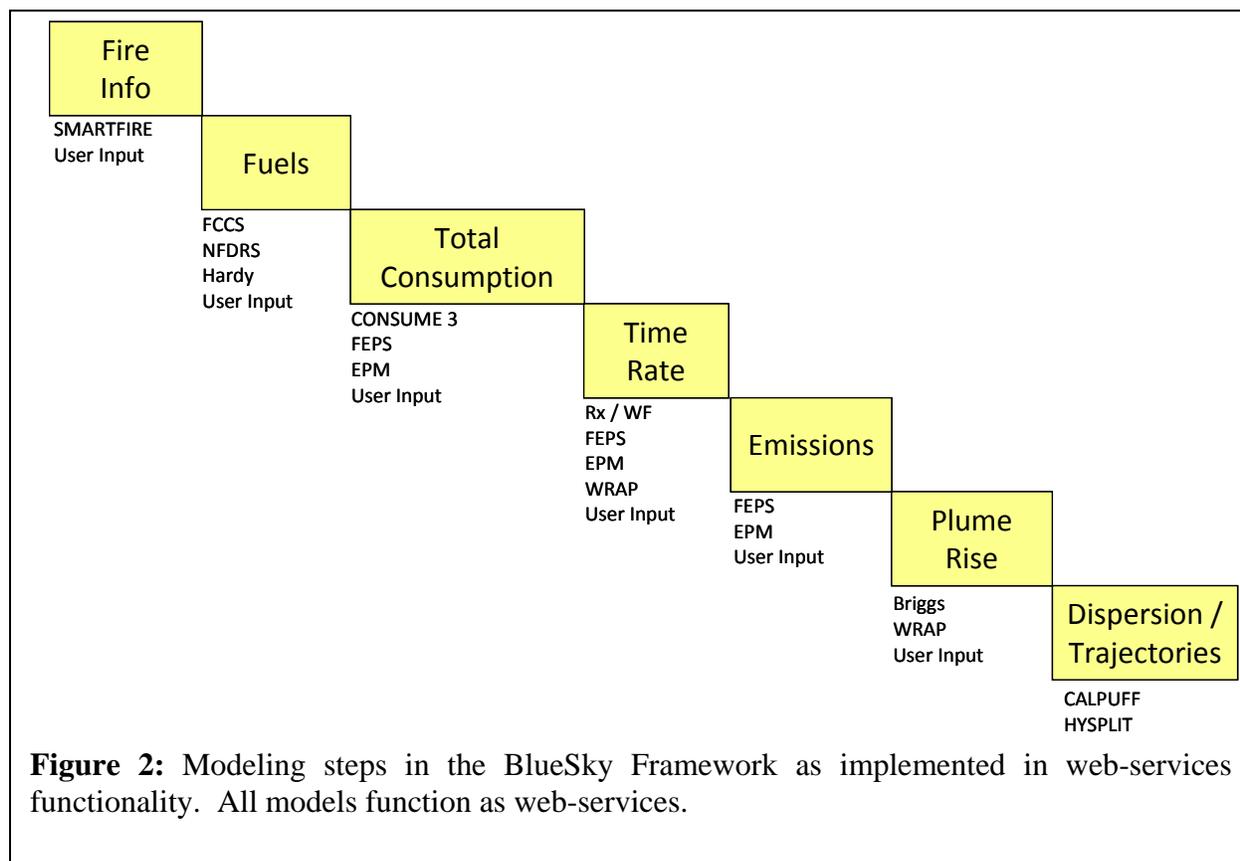
Recently completely rewritten under a grant from NASA, BlueSky version 3.0 is a modular modeling framework that connects fire information, fuel loading, fire consumption, fire emissions, plume rise, trajectory, and dispersion models, making the process of combining these models to produce a desired output (e.g. PM<sub>2.5</sub> emissions or smoke trajectories from a fire) easier. BlueSky has done this by creating a sequence of modeling steps defined by a set of standard input/output interfaces (see Figure 1). Individual models are then enabled to use the



**Figure 1:** Example of model steps and interfaces. Defining standard model steps also defines standard interfaces. Not all models need stop or produce output at each interface step (e.g. M4 and M5). Because of the standard interfaces, combinations of models (e.g. M1 + M2 + M3, M4+M3, M1+M5) are easy. Also any application that uses one combination (e.g. M1+M2) can be easily switched to use a different equivalent path (e.g. M4).

standard input/output interfaces by wrapping existing models (e.g. CONSUME 3) with lightweight software code. These wrapped models are referred to as modules and are run in the context of the BlueSky framework, which links the modules together and provides a set of standard utility routines (e.g. unit conversions) for use by all modules. The framework also provides for dispersed execution across a set of machines and other software enhancements designed to let it run faster and more efficiently.

At each modeling step, several different models have been implemented as BlueSky modules. Figure 2 shows the steps and models used in this project. The user of BlueSky can choose one of the implemented modules at each step, which allows for a number of different model combinations to reach a desired output level.



## Description

Location of work: Seattle, WA and Petaluma, CA  
 Geographic area of study: National

## Goals

This project's goals can be grouped into two categories: those focused on the development of a collaborative architecture, and those focused on the utility of the applications created.

With respect to the collaborative architecture:

1. Modify the BlueSky Framework to lightweight it to serve as a collaborative architecture;
2. Create individual stand-alone modules for each of the models in BlueSky;
3. Wrap these stand-alone modules so that they can function through standard web-service architecture Internet protocols;
4. Demonstrate the power of such an architecture; and
5. Document how to utilize such an architecture to create decision support applications.

With respect to the applications created for this proposal:

1. Create simple applications that can serve as examples;
2. Create an application that shows how both local and remote models can be combined;
3. Create a game-playing smoke modeling application useful in teaching RX-410; and
4. Demonstrate the full capabilities of this type of architecture.

## Methods and work summary

This project functionally required 3 pieces of work:

1. Revise the BlueSky Framework to serve as a lightweight collaborative architecture that uses web-services;
2. Create sample applications and documentation; and
3. Create an advanced game-playing smoke modeling application.

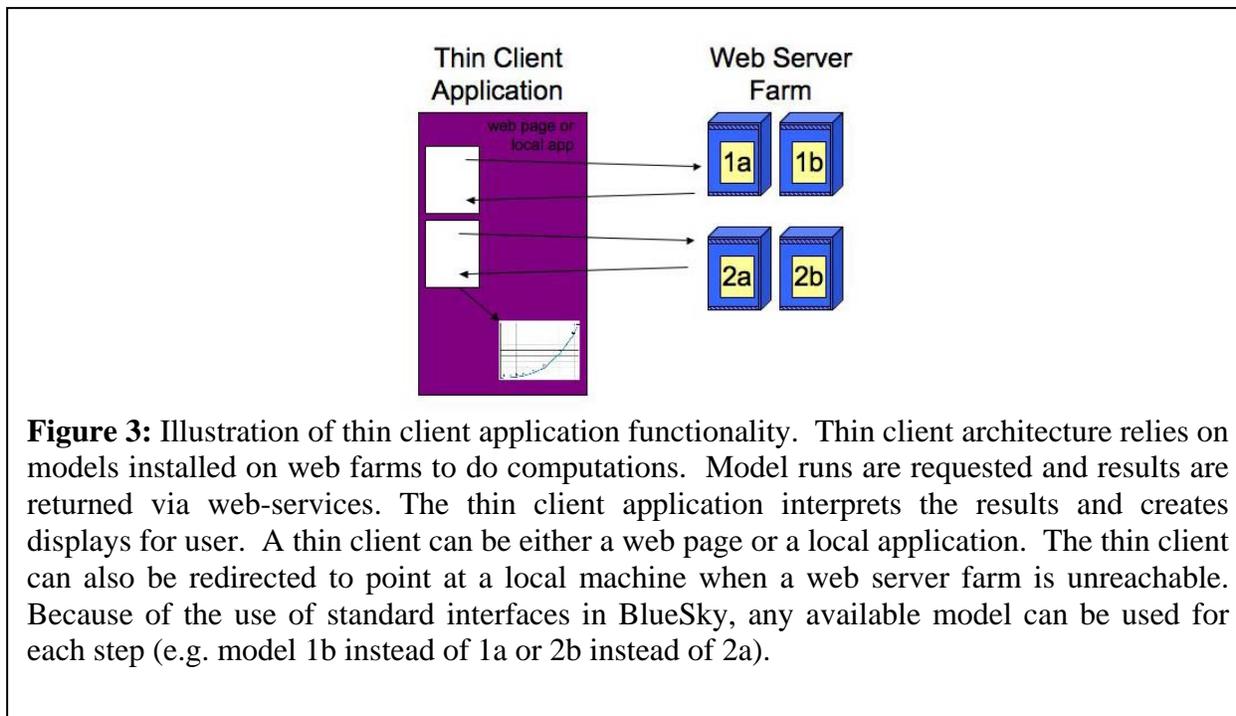
### **Revision of the BlueSky Framework**

The BlueSky Framework was significantly modified for this project, resulting in a new version, named BlueSky 3.1. Note that this work was done in line with other modifications to BlueSky, making the modifications done for this project part of the core framework and embedded in any future BlueSky development.

BlueSky 3.1 differs from past versions in that it:

- Is streamlined and restructured, eliminating some suboptimal legacy code unnecessary in the web-service architecture;
- Enables BlueSky to run as a standalone web-services server; and
- Allows the web-services functionality of BlueSky to utilize the web standard XML format in addition to the CSV formats previously favored by BlueSky.

BlueSky now has a command line switch (“bluesky -ws”) which, when invoked, results in BlueSky running as a web-services daemon that allows users to connect to it through a standard web-service call. The revisions to the BlueSky Framework allow the development of decision support applications built in any number of ways, including as thin clients (see Figure 3).



### Creation of simple applications and documentation

Alone, the web-service framework is not user friendly but requires a front-end to handle the data visualization. Both simple example front ends and a more advanced game-playing application were developed.

By placing the computational and database requirements on a remote server, applications can be developed that are extremely lightweight and focus only on the user interface (either as a web page or as a simple desktop program). We will create a sequence of extremely simple applications that show how this can be done, and show how an application can switch between utilizing a remote web-service model and a locally installed model (for use when not connected to the Internet). A simple web-based application example is shown in Figure 4 and available on the web through the project page: <http://blueskyframework.org/ws>. An FAQ and more technical documentation for anyone interested in building their own specialized application can also be found there.

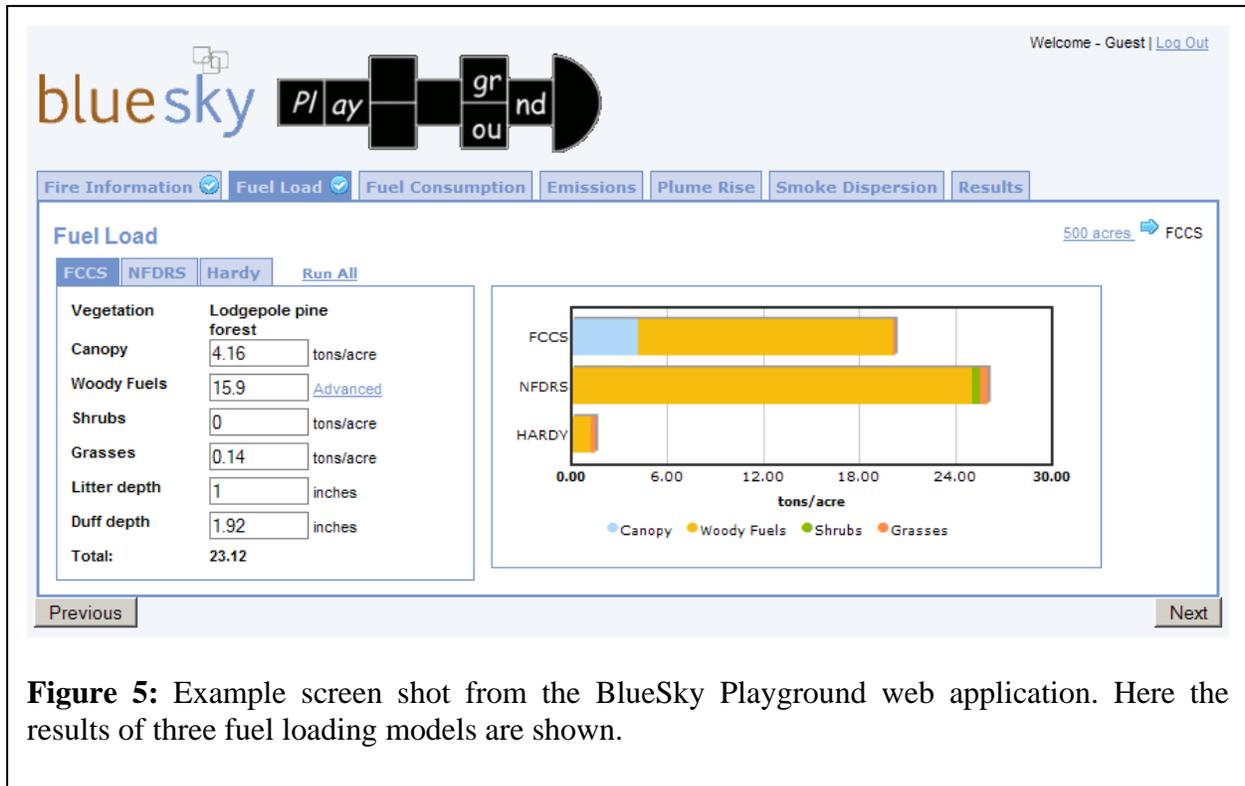
## BlueSky Framework webservice test

|   |  |
|---|--|
| <b>Run fuel loading model</b><br>Latitude: <input type="text" value="45.46"/><br>Longitude: <input type="text" value="-114.961"/><br>Model: <input type="text" value="FCCS"/><br><input type="button" value="Submit"/>  | <b>Result</b><br>Number of fires returned: 1<br>Fire #: UNKNOWN<br>Latitude: 45.46<br>Longitude: -114.961<br>Vegetation type: Lodgepole pine forest<br>1-hr fuels: 0.4<br>10-hr fuels: 2.2<br>100-hr fuels: 2.8<br>1,000-hr fuels: 8.3<br>10,000-hr fuels: 0.7<br>> 10,000-hr fuels: 0.5<br>Duff: 1.92<br>Grass: 0.14<br>Litter: 1<br>Rotten fuels: 1<br>Canopy: 4.16  |
| <b>Request</b> <pre>&lt;?xml version="1.0"?&gt; &lt;methodCall&gt; &lt;methodName&gt;FCCS&lt;/methodName&gt; &lt;params&gt; &lt;param&gt; &lt;struct&gt; &lt;member&gt; &lt;name&gt;latitude&lt;/name&gt; &lt;value&gt;&lt;double&gt;45.46&lt;/double&gt;&lt;/value&gt; &lt;/member&gt; &lt;member&gt; &lt;name&gt;longitude&lt;/name&gt; &lt;value&gt;&lt;double&gt;-114.961&lt;/double&gt;&lt;/value&gt; &lt;/member&gt; &lt;/struct&gt; &lt;/param&gt; &lt;/params&gt; &lt;/methodCall&gt;</pre> | <b>Response</b> <pre>&lt;?xml version="1.0"?&gt; &lt;methodResponse&gt; &lt;params&gt; &lt;param&gt; &lt;value&gt;&lt;struct&gt; &lt;member&gt; &lt;name&gt;fires&lt;/name&gt; &lt;value&gt;&lt;struct&gt; &lt;member&gt; &lt;name&gt;dispersion_offset&lt;/name&gt; &lt;value&gt;&lt;int&gt;0&lt;/int&gt;&lt;/value&gt; &lt;/member&gt; &lt;member&gt; &lt;name&gt;fire_events&lt;/name&gt; &lt;value&gt;&lt;array&gt;&lt;data&gt; &lt;/data&gt;&lt;/array&gt;&lt;/value&gt; &lt;/member&gt; &lt;member&gt; &lt;name&gt;start_date&lt;/name&gt; &lt;value&gt;&lt;string&gt;200908241719Z&lt;/string&gt;&lt;/value&gt; &lt;/member&gt; &lt;member&gt; &lt;name&gt;fire_locations&lt;/name&gt; &lt;value&gt;&lt;array&gt;&lt;data&gt; &lt;value&gt;&lt;struct&gt;</pre> |

**Figure 4:** Simple example showing how to access and use the web-services capabilities. The upper left box allows the user to input basic fire location information. The lower left box shows the request as sent to the web-services model. The formatted result is shown in the upper right and the actual result returned to the web-page in the lower right. The actual page can be accessed through <http://playground.blueskyframework.org/example/>. The full code for this page (including formatting) is listed in Appendix A.

### Creation of advanced game-playing smoke modeling application

In order to show the full capabilities of this type of collaborative modeling architecture, we created a unique, advanced game-playing application for smoke modeling that is initially targeted at RX-410 classes but is also useful for decision makers. A screenshot of the application front end, called BlueSky Playground, is shown in Figure 5. A detailed walkthrough showing how to use the application is given in Appendix B.



**Figure 5:** Example screen shot from the BlueSky Playground web application. Here the results of three fuel loading models are shown.

BlueSky provides several unique functions: it allows access to many models through a single interface, it allows for comparison of different model's output directly, and, since all the models are run in a few seconds while you wait, it also provides the first ever on-the-fly smoke dispersion calculation capabilities for the smoke management community.

BlueSky Playground allows users to enter a fire size and location and then leads the user through all of the steps in the BlueSky smoke modeling pathway up to and including the creation of smoke impact maps for the fire. At each step the user is shown the outputs from the available models (e.g. fuel loadings from FCCS, NFDRS, and Hardy) and is able to select one before continuing to the next step. Outputs include fuel loadings, total fire consumption, fire emissions, plume rise, and smoke impact maps, though users can stop at any point. By seeing the outputs from all of the model choices at each step, users can see the inter-model variability; by altering the model choices or information at any point, users are able to see the effects on the remaining modeling steps.

## Key Findings

### BlueSky Framework now web-service enabled

BlueSky now has a command line switch to run BlueSky as a web-services daemon that allows users to connect to it through as a standard web-service function. This functionality enables *all* models and modules in BlueSky to be run as web-services without need for altering the model or module code. When first envisioned and proposed, web-service

capability for every module was not promised as it was unclear whether it could be delivered without custom code development for every module. We are pleased to have been able to make BlueSky function through a standard web-service function because it means that:

- All existing modules in BlueSky can be used as web-services;
- All future modules in BlueSky will automatically function as web-services; and
- New or additional modeling steps developed in the future (e.g., radiative fire power emissions calculations) will be automatically enabled as web-services.

### **Models available on-line**

The BlueSky Framework is now exposed to developers wishing to produce applications that can make use of BlueSky services. Currently, users can access them through both example webpage scripts as well as the BlueSky Playground tool. A list of servers is maintained on the project page (<http://blueskyframework.org/ws>).

The multitude of models within BlueSky, coupled with its inherent flexibility, make multiple uses by multiple user communities possible. Each of these communities can benefit from focused applications that expose only the relevant functionality in BlueSky within custom designed user interfaces. The new web-service functionality enables this ability to provide multiple applications from a single framework. Additionally, as the framework is improved and upgraded, applications benefit automatically without need for re-installation.

### **Example applications developed**

To provide examples of the benefits of the new web-service architecture of the BlueSky Framework, example applications have been developed. One is a very simple application for looking up fuel loading by location that also serves as a code example for developers to follow when developing their own applications. The other example is a full-fledged, modern web-application that exposes most of the models in BlueSky and allows users to perform smoke modeling runs and examine how model choices affect results. This tool provides a way for any user to run BlueSky from a web browser, requiring no software installation or expert knowledge.

### **Initial framework work more complicated, application development less complicated than expected**

The exposure of the full functionality of BlueSky as web-services required substantial code development on the BlueSky core framework. The key benefit from this effort is that development of applications that take advantage of BlueSky functionality are now much easier and faster to develop. Developing the user interfaces for this project such as the example webpage and the advanced BlueSky Playground application was much simpler than it would have been without the new web-service functionality. This experience suggests that even advanced interfaces aimed at specific user communities can be built with only modest expenditures of time and money. In addition, as needs are discovered, new focused interfaces can be produced quickly using the BlueSky web-services, which provide a set of ready made building blocks.

## Management Implications

### The ability for managers to use one interface and access many different models

Much of the confusion over different models reflected in the JFSP Smoke Roundtables and other user feedback is due to the fact that each new model also comes with a new interface that the user must master. Use of web-services separates the scientific model calculations from the interface used to gain access to them. By creating standard input/output formats and modularizing different models to use them as has been done with BlueSky, it further enables the same user interface to function with different models. The capability of this is shown through the developed BlueSky Playground game-playing application. While this particular interface is not optimized for any particular user group, it shows how a simple web-interface design could be created and optimized to meet the needs of different user groups.

### The ability for managers to easily access to the latest science including high-performance computing capabilities

There are several related benefits that web-service enabled software provides in the management context. Federal agency employees may have difficulty installing new desktop software onto their agency computers. Any time a new revision is created, the developers must work to get users to install the new version, and there may be additional hurdles to doing so. With web-services, the software is centrally located and managed. New science can be distributed quickly, as it need only be installed on the web server. Finally, web-services can be accessed by modestly powered workstations, laptops, and smart phones with Internet access. This is particularly important for the models in the BlueSky framework. For example, the smoke trajectory and dispersion models are processor intensive and require many gigabytes of meteorological model data to run. The web-service enabled framework allows users easy access to these high performance models without locally stored data.

### A sample game-playing application for learning and scenario game-playing

In addition to the many different user interfaces provided by the various scientific models (discussed above), the models themselves produce very different scientific results. The understanding of the differences between the models and the uncertainties associated with modeling in general can be explored using the BlueSky Playground application. At most steps, the user is given the option of running multiple models and the results are immediately available for inter-comparison. These differences are instructive and can serve to educate smoke management personnel, such as in the context of an RX-410 class.

## Related Work

This project has highly leveraged on-going work from several different projects.

Primarily, this project has utilized the rewrite of the BlueSky code that was funded through a NASA ROSES grant. Because of overlap in these two projects, the web-services functionality

developed here was able to be directly placed into the finalization of the rewrite done for NASA. Additional work that will affect the web-services enabled BlueSky will be continuing through a new NASA ROSES grant, making the additional features developed for this new project also web-service enabled.

Additionally this work has been done in communication and collaboration with the new Interagency Fuels Treatment – Decision Support System (IFT-DSS) project underway through funding from the Joint Fire Science Program. The IFT-DSS project aims to create a scientific modeling architecture for fuels treatment management and modeling. Many of the lessons learned from BlueSky and from this project have been discussed with the management team of IFT-DSS. Additionally, when IFT-DSS becomes functional, the IFT-DSS system is planned to communicate directly with the web-service BlueSky system allowing for both to function as a meta-scientific software architecture.

The BlueSky Playground application developed here also is featured in a new prototype air quality functionality developed for the Wildland Fire Decision Support System (WFDSS). WFDSS is web-service function call enabled, making the web-service capability of BlueSky an easy fit for getting smoke modeling capabilities into WFDSS. It is expected that in the future more tailored user interfaces will help facilitate this integration. These interfaces will gain access to the BlueSky modeling capabilities through the web-services functionality developed here.

Work on making the BlueSky Playground more operational for land managers, including some interface improvements and connection to a large archive of meteorology which will allow it to be used by anyone wanting to examine fire retrospectively is underway through a grant from the America Recovery and Reinvestment Act (ARRA). This short-term project should make a second version of the BlueSky Playground available on the web before the 2010 fire season.

With the ARRA improvements, the BlueSky Playground will also be incorporated into the prototype Emergency Smoke Response System (ESRS) developed at the request of USFS managers. The prototype ESRS, which was deployed in 2007 and 2008 on the California fires, combines enhanced fire weather and smoke modeling with increased observational monitoring of smoke. The BlueSky Playground can add to this capability by providing real-time feedback on burnouts and other management possibilities being considered.

## **Future Work Needed**

The primary need is to place the web-service BlueSky onto operational (24x7 monitored and managed) servers. Currently the web-service BlueSky is running on research servers without an operational mandate or funding. An option for placement on dedicated operational servers is needed not only for BlueSky but for all future management oriented web-services based scientific modeling.

Additional short-term work is needed to connect the BlueSky Playground application with an archive of meteorology so that different days can be chosen. This work is underway through a grant from ARRA.

In the medium term, development of new interfaces that take advantage of the BlueSky web-service functions but are tailored to specific uses and user groups is needed. These can be stand-alone applications or websites, or integrated into existing applications and websites (e.g. adding “click here to view smoke impacts” button). Additionally the integration of BlueSky and the new IFT-DSS system should be expedited.

Longer-term, work is needed to adopt collaborative architectures that enable different models to communicate. Doing so would allow the development of scientific models and user interfaces to be separated and parceled out to different groups that excel in such matters. Scientists would be able to focus on making the best models, and website designers on making the best-unified interfaces. Utilizing web-service and other advanced Internet functionalities will allow delivery of the benefits of these scientific models to users without the need for many different interfaces and software installations. Placement of applications on dedicated operational servers will allow for guaranteed availability to the latest version of each model from any system with an Internet connection. Development of website applications that automatically switch to a local backup copy would enable the use of critical models even in remote areas without Internet connections.

## Deliverables Crosswalk Table

**Table 1. Deliverables Crosswalk Table**

| <b>Deliverable Type</b>   | <b>Description</b>  | <b>Promised Delivery Dates</b> | <b>Status (9/30/09)</b>       |
|---------------------------|---|--------------------------------|-------------------------------|
| Website                   | Web-service versions of models in BlueSky Framework                       | Initial: 10/08<br>Final: 2/09  | COMPLETE                      |
| Non-Refereed Publication  | Documentation for utilizing web-services to create applications           | Initial: 11/08<br>Final: 5/09  | COMPLETE<br>see below         |
| Website                   | Website showing simple example applications and documentation             | Initial: 11/08<br>Final: 5/09  | COMPLETE<br>see below         |
| Website                   | Game-playing smoke modeling application                                   | Initial: 1/09<br>Final: 5/09   | COMPLETE<br>see below         |
| Peer-reviewed publication | Journal article or peer-reviewed GTR detailing the system and application | 5/09 (extended)                | UPCOMING*<br>(see below)      |
| Non-Refereed Publication  | Final Report  | 5/09 (extended)                | COMPLETE                      |
| Presentation              | At National Air Quality Conferences                                       | Spring '09                     | COMPLETE                      |
| Presentation              | At BlueSky Stakeholders Meeting   | Spring '09                     | N/A**                         |
| Presentation(s)           | At national fire conferences  | Spring through Fall '09        | UPCOMING***<br>(10/09, 11/09) |
| Invited Presentation      | At National Weather Service Air Quality Workshop                          | Fall '09                       | UPCOMING***<br>(10/09)        |
| Training Sessions         | Incorporated into RX410 classes where BlueSky is taught                   | Winter '09/Spring '10          | UPCOMING***<br>Winter/Spring  |

### *Notes:*

\*A technical document on how to utilize the web-services and additional information on the BlueSky Framework, the BlueSky Playground and more are available online through the project website (<http://blueskyframework.org/ws>). A General Technical Report is in draft form and will be forthcoming.

\*\*BlueSky Stakeholders Meeting was not held this year (2009). This work will be presented at the next meeting (2010).

\*\*\*Several project deliverables listed in the original proposal were outside the timeline of the project due to conference and RX410 schedules. Abstracts on this work have been submitted to the Fire and Forest Meteorology, 4<sup>th</sup> Fire Conference, and American Geophysical Union meetings being held in Fall/Winter 2009.

## **Websites, Applications, and Documentation**

Additional information including access to the simple example application, the more advanced BlueSky Playground, and in-depth technical documentation can be found at <http://blueskyframework.org/ws>.

## **Presentations**

Chinkin L.R., Strand T.M., Brown T., Goodrick S., Larkin N.K., Raffuse S. M., Solomon R., Sullivan D.C., Lahm P. 2009. Development and applications of systems for modeling emissions and smoke from fires: the BlueSky smoke modeling framework, SMARTFIRE, and associated systems. National Air Quality Conferences, Dallas, Texas, March 2-5.

Craig K.J., Wheeler J.M.W., Raffuse S.M., Sullivan D.C., Reid S.B., Solomon R., Strand T., Larkin S. 2008. BlueSky Gateway: providing access to products from the BlueSky smoke modeling program. 7th Annual Community Modeling and Analysis System (CMAS) Conference, Chapel Hill, NC, October 6-8.

Larkin N.K., Strand T.M., Solomon R., Raffuse S., Sullivan D., Chinkin L., Lahm P., Acheson A., Brown T., Friedl L. 2008. BlueSky, SMARTFIRE, SEMIP and associated efforts. NASA Biomass Burning Coordination Meeting, University of Maryland, College Park, MD.

Larkin N.K., Strand T.M., Solomon R., Rorig M., Krull C., Sullivan D., Raffuse S., Pryden D., Ovard C., Chinkin L., O'Neill S., Friedl L., Knighton R. 2008. Prototyping the Emergency Smoke Response System (ESRS). WESTAR Fall Business Meeting. Seattle, Washington, October 1-3.

Larkin N.K., Strand T., Solomon R., Raffuse S., Sullivan D.C., Chinkin L., Brown T., O'Neill S., Friedl L., and Knighton R. 2008 The state of smoke tools: What we know now. International Association of Wildland Fire, The '88 Fires: Yellowstone & Beyond, Jackson Hole, WY, September 22-27.

- Larkin N.K., Strand T.M., Solomon R., Raffuse S., Sullivan D.C., Chinkin L., Brown T., O'Neill S., Friedl L., Knighton R. 2008. The state of smoke tools: What we know now. NIFC, Boise, Idaho, May 21.
- Strand T.M., Larkin N.K., Solomon R., Raffuse S., Sullivan D., Craig K., Pryden D., Wheeler N., Chinkin L., Brown T., Procter T. 2008. New tools for fire and smoke and their application to the 2008 California wildfires. Pacific Coast Fire Conference: Changing Fire Regimes, Goals and Ecosystems, San Diego, California, December 1-4.
- Strand T.M., Potter B.P., Larkin N.K., Solomon R., Rorig M., Krull C. 2008. AirFire Smoke Research. Hood River, Oregon, October 29-31.
- Strand T.M., Sullivan D.C., Larkin N.K. 2008. BlueSky Status. 2008 BlueSky Modeling Stakeholders Meeting, Boise, Idaho, May 20-22.

## References

- JFSP. (2007) Smoke and air quality roundtables, research needs and assessment. Joint Fire Science Program. 16pp. Available on the Internet at <<http://www.firescience.gov>>
- Larkin N.K., O'Neill S.M., Solomon R., Raffuse S., Strand T.M., Sullivan D.C., Krull C., Rorig M., Peterson J., and Ferguson S.A. (2009) The BlueSky smoke modeling framework. *Int. J. Wildland Fire* (in press).
- Palmquist M.S. (2008) Working summary of the SEI's engagement with the Joint Fire Science Program. Report prepared for the U.S. Department of Defense by the Acquisition Support Program, Software Engineering Institute, Carnegie Mellon University, April.

## Appendix A: Webpage Source for Web-services Example

See Figure 3 of main document for example output created using this webpage.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>BlueSky Framework webservice test</title>
<script type="text/javascript" src="xmlrpc_lib.js"></script>
<script type="text/javascript">

    function getRequest(methodName, latitude, longitude) {
        return '<' + '?xml version="1.0"?>\n' +
            '<methodCall>\n' +
            '<methodName>' + methodName + '</methodName>\n' +
            '<params>\n' +
            '<param>\n' +
            '<struct>\n' +
            '  <member>\n' +
            '    <name>latitude</name>\n' +
            '    <value><double>' + latitude.toString() + '</double></value>\n' +
            '  </member>\n' +
            '  <member>\n' +
            '    <name>longitude</name>\n' +
            '    <value><double>' + longitude.toString() + '</double></value>\n' +
            '  </member>\n' +
            '</struct>\n' +
            '</param>\n' +
            '</params>\n' +
            '</methodCall>\n';
    }

    function escapeHTML(s) {
        return s
            .replace(/&/g, '&amp;');
            .replace(/</g, '&lt;');
            .replace(/>/g, '&gt;');
    }

    function displayRequest(requestXML) {
        var div = document.getElementById("request");
        div.innerHTML = "<h3>Request</h3><pre>" + escapeHTML(requestXML) + "</pre>";
        div.style.visibility = "visible";
    }

    function displayResponse(responseXML) {
        var div = document.getElementById("response");
        div.innerHTML = "<h3>Response</h3><pre>" + escapeHTML(responseXML) + "</pre>";
        div.style.visibility = "visible";
    }

    function parseResponse(xml) {
        // OK, here we have the result of the XML-RPC method call, encoded as XML.
        // Rather than spending a lot of time messing around with XML parsing,
        // we're going to use the open-source xmlrpc_lib.js library to parse it
        // for us. (In fact, the library would have made the rest of the process
        // easier too, but we did it the hard way so we could see the actual XML
        // being sent back and forth.)

        // There's no reason why this particular library is needed; there are plenty
        // of XML-RPC libraries out there, some of which may be nicer than this one.
        // I'm using this particular library because I was able to find it in less
        // than 30 seconds using Google.

        var div = document.getElementById("result");

        // Parse the result with xmlrpc_lib.js
        var msg = new xmlrpcmsg();
        var data = msg.parseResponse(xml, true, 'jsvars');

        // xmlrpc_lib.js likes to use lots of ".me" members (!)
    }

```

```

var fire_locations = data.val.me.fires.me.fire_locations.me;

div.innerHTML =
  "<h3>Result</h3>" +
  "<b>Number of fires returned:</b> " + fire_locations.length + "<br/><br/>";

// Loop through the returned fires (normally there will only be one,
// since we only provided input data for one)
for(var i = 0; i < fire_locations.length; i++) {
  var fireLoc = fire_locations[i].me;

  // Output some basic information on each fire
  div.innerHTML +=
    "<b>Fire #</b> " + (i + 1) + " :</b> " + fireLoc.id.me + "<br/>" +
    "<b>Latitude:</b> " + fireLoc["latitude"].me + "<br/>" +
    "<b>Longitude:</b> " + fireLoc["longitude"].me + "<br/>";

  var fuels = fireLoc.fuels.me;
  if(fuels) {
    // Output FuelsData information
    div.innerHTML +=
      "<blockquote>" +
      "<b>Vegetation type:</b> " + fuels["metadata"].me["VEG"].me + "<br/>" +
      "<b>1-hr fuels:</b> " + fuels["fuel_1hr"].me + "<br/>" +
      "<b>10-hr fuels:</b> " + fuels["fuel_10hr"].me + "<br/>" +
      "<b>100-hr fuels:</b> " + fuels["fuel_100hr"].me + "<br/>" +
      "<b>1,000-hr fuels:</b> " + fuels["fuel_1khr"].me + "<br/>" +
      "<b>10,000-hr fuels:</b> " + fuels["fuel_10khr"].me + "<br/>" +
      "<b>&gt; 10,000-hr fuels:</b> " + fuels["fuel_gt10khr"].me + "<br/>" +
      "<b>Duff:</b> " + fuels["duff"].me + "<br/>" +
      "<b>Grass:</b> " + fuels["grass"].me + "<br/>" +
      "<b>Litter:</b> " + fuels["litter"].me + "<br/>" +
      "<b>Rotten fuels:</b> " + fuels["rot"].me + "<br/>" +
      "<b>Canopy:</b> " + fuels["canopy"].me + "<br/>" +
      "</blockquote>" +
      "<br/>";
  } else {
    div.innerHTML +=
      "<b>fuels member of FireLocation object is null</b>";
  }
}

function runModel() {
  // Get the values of the form fields
  var latitude = document.getElementById("latitude").value;
  var longitude = document.getElementById("longitude").value;
  var methodName = document.getElementById("model").value;

  // Construct the request XML and display it in the box using displayRequest()
  var requestXML = getRequest(methodName, latitude, longitude);
  displayRequest(requestXML);

  // Basic XMLHttpRequest stuff. Send the request via HTTP POST to the server.
  var url = "http://" + window.location.host + "/xml-rpc.py";
  var req = new XMLHttpRequest();
  req.open("POST", url, true);
  req.onreadystatechange = function(e) {
    if(req.readyState == 4) {
      if(req.status == 200) {
        // OK, if we get here, then we have a valid response from the
        // server. Display the raw text of the response with
        // displayResponse(), and hand the text off to parseResponse()
        // to come up with a prettier result.
        displayResponse(req.responseText);
        parseResponse(req.responseText);
      } else {
        alert("ERROR: HTTP error code " + req.status);
      }
    }
  };
  req.send(requestXML);
}

window.onload = function() {
  document.getElementById("mainForm").onsubmit = function() { return false; };
  document.getElementById("btnSubmit").onclick = runModel;
};
</script>
<style type="text/css">

```

```

body {
  color: black;
  background: white;
  font: 14px Arial, sans-serif;
  width: 780px;
}

h1 {
  font: bold 24px Verdana, sans-serif;
  border-bottom: 1px solid black;
  margin: 0 0 10px 0;
  padding: 0;
}

h3 {
  font: bold 16px Verdana, sans-serif;
  border-bottom: 1px solid #999;
  margin: 0 0 10px 0;
  padding: 0;
}

form, #result, #request, #response {
  border: 1px solid #999;
  width: 355px;
  float: left;
  padding: 10px;
  margin: 5px;
}

form, #result {
  background: #def;
}

#request, #response {
  background: #dfe;
  visibility: hidden;
}

form, #request {
  clear: left;
}

label {
  width: 100px;
  text-align: right;
  clear: left;
  float: left;
  padding-right: 10px;
  line-height: 24px;
}

input, select {
  width: 235px;
  float: left;
}

input.button {
  width: auto;
  float: right;
  margin: 10px 10px 0 0;
}

</style>
</head>
<body>
  <h1>BlueSky Framework webservice test</h1>
  <form id="mainForm">
    <h3>Run fuel loading model</h3>
    <label for="latitude">Latitude:</label>
    <input name="latitude" id="latitude" value="45.46"/>

    <label for="longitude">Longitude:</label>
    <input name="longitude" id="longitude" value="-114.961"/>

    <label for="model">Model:</label>
    <select id="model" name="model">
      <option value="NoFuelLoading">NoFuelLoading</option>
      <option value="FCCS" selected="selected">FCCS</option>
      <option value="NFDRS">NFDRS</option>
    </select>
  </form>

```

```
    <input type="submit" class="button" id="btnSubmit" value="Submit"/>
  </form>
  <div id="result">
    <h3>Result</h3>
  </div>
  <div id="request"></div>
  <div id="response"></div>
</body>
</html>
```

## **Appendix B: BlueSky Playground Walkthrough**

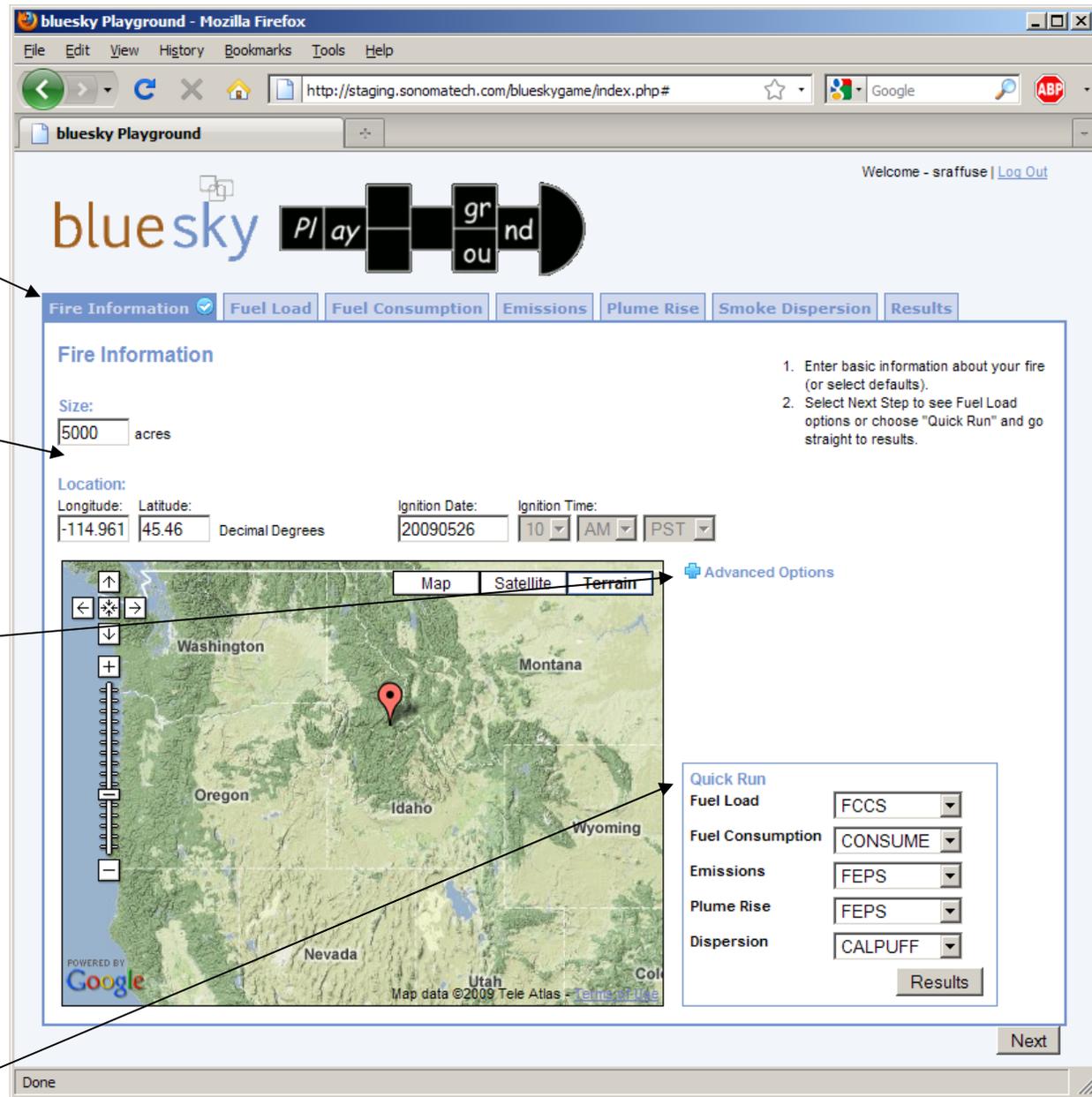
The following pages contain an example walkthrough of the BlueSky Playground web application.

The BlueSky Playground walks users step-by-step through the steps required to model smoke emissions and dispersion from fires. Tabs along the top of the interface allow navigation back and forth and show what steps have been completed.

Fire Information is the first step. The basic information BlueSky needs includes fire size, location, and time of ignition. Location can be specified by clicking on the interactive map.

Advanced options can also be selected, allowing the user to provide additional information about the fire.

The BlueSky Playground can be run either step-by-step or all at once. To run step-by-step, the user clicks on the "Next" button or selects the next step (Fuel Load) from the tabs above. Alternately, if the user is not interested in the results of individual models, they can preselect all of the steps and go straight to results using the Quick Run feature.



As soon as the Fuel Load tab is clicked, a fuel loading model in the BlueSky Framework is run. Notice that the Fire Information tab has a checkmark, indicating that it is complete.

This section indicates the modeling pathway that the user has followed to this point. In this example, a 5000 acre fire followed by FCCS fuel loading.

The Fuel Load through Plume Rise tabs are all structured similarly. On the left are the model choices, selectable by tabs. Editable model outputs are shown in text boxes. Outputs from one step serve as inputs to the next step. The user can change these values before the next model is run.

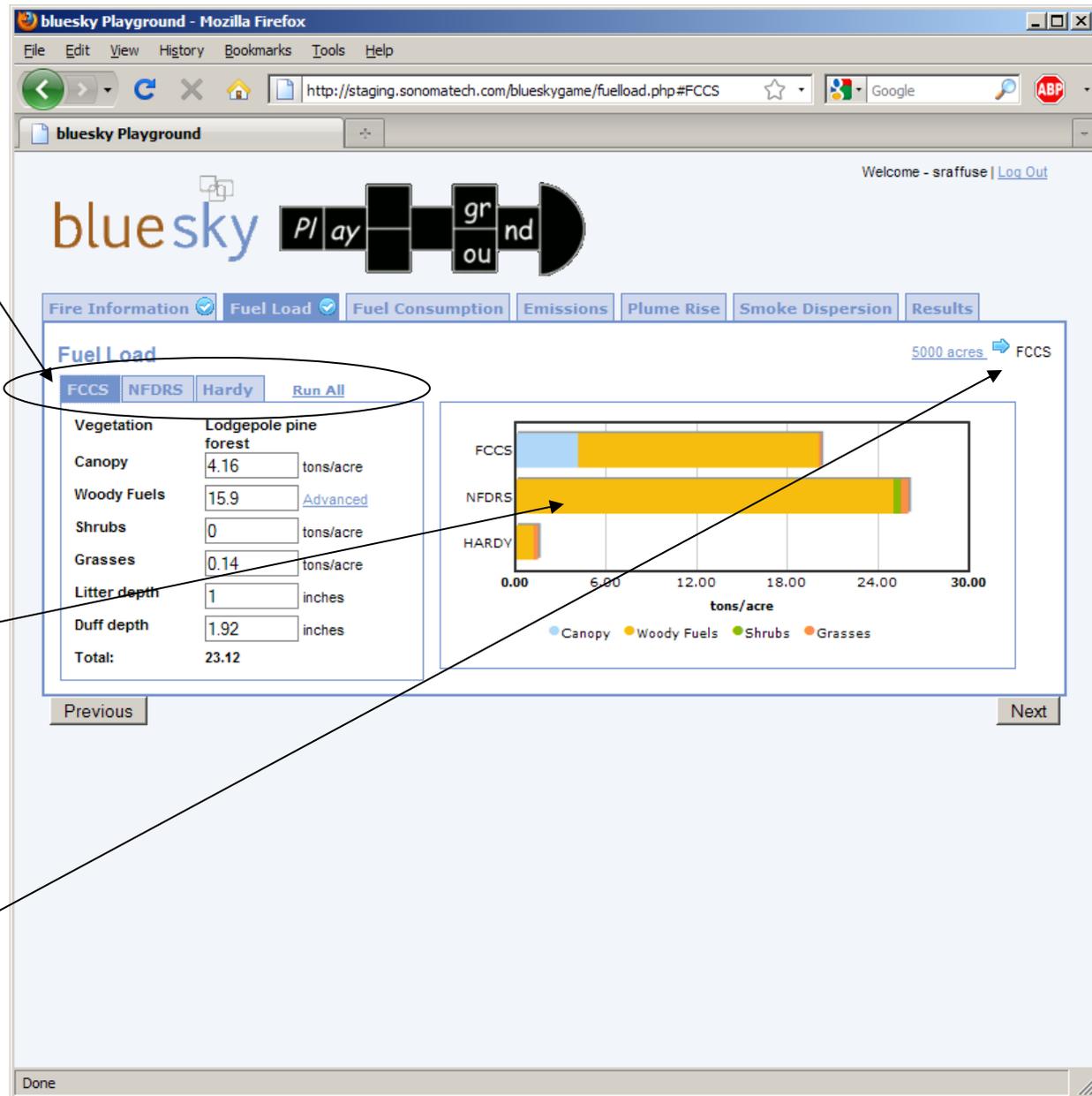
On the right side is a graphical representation of the output for this step.

The screenshot shows the 'bluesky Playground' web application in a Mozilla Firefox browser. The URL is <http://staging.sonomatech.com/blueskygame/fuelload.php#>. The page features a navigation bar with tabs: Fire Information (checked), Fuel Load (checked), Fuel Consumption, Emissions, Plume Rise, Smoke Dispersion, and Results. The 'Fuel Load' tab is active, displaying a table of input parameters and a bar chart of the resulting fuel load.

| Vegetation    | Value        | Unit      |
|---------------|--------------|-----------|
| Canopy        | 4.16         | tons/acre |
| Woody Fuels   | 15.9         | Advanced  |
| Shrubs        | 0            | tons/acre |
| Grasses       | 0.14         | tons/acre |
| Litter depth  | 1            | inches    |
| Duff depth    | 1.92         | inches    |
| <b>Total:</b> | <b>23.12</b> |           |

The bar chart on the right shows the fuel load components in tons/acre. The x-axis ranges from 0.00 to 30.00. The components are: FCCS (blue, ~4.16), Woody Fuels (yellow, ~15.9), Shrubs (green, 0), and Grasses (orange, ~0.14). A '5000 acres' button is circled in the top right corner of the page.

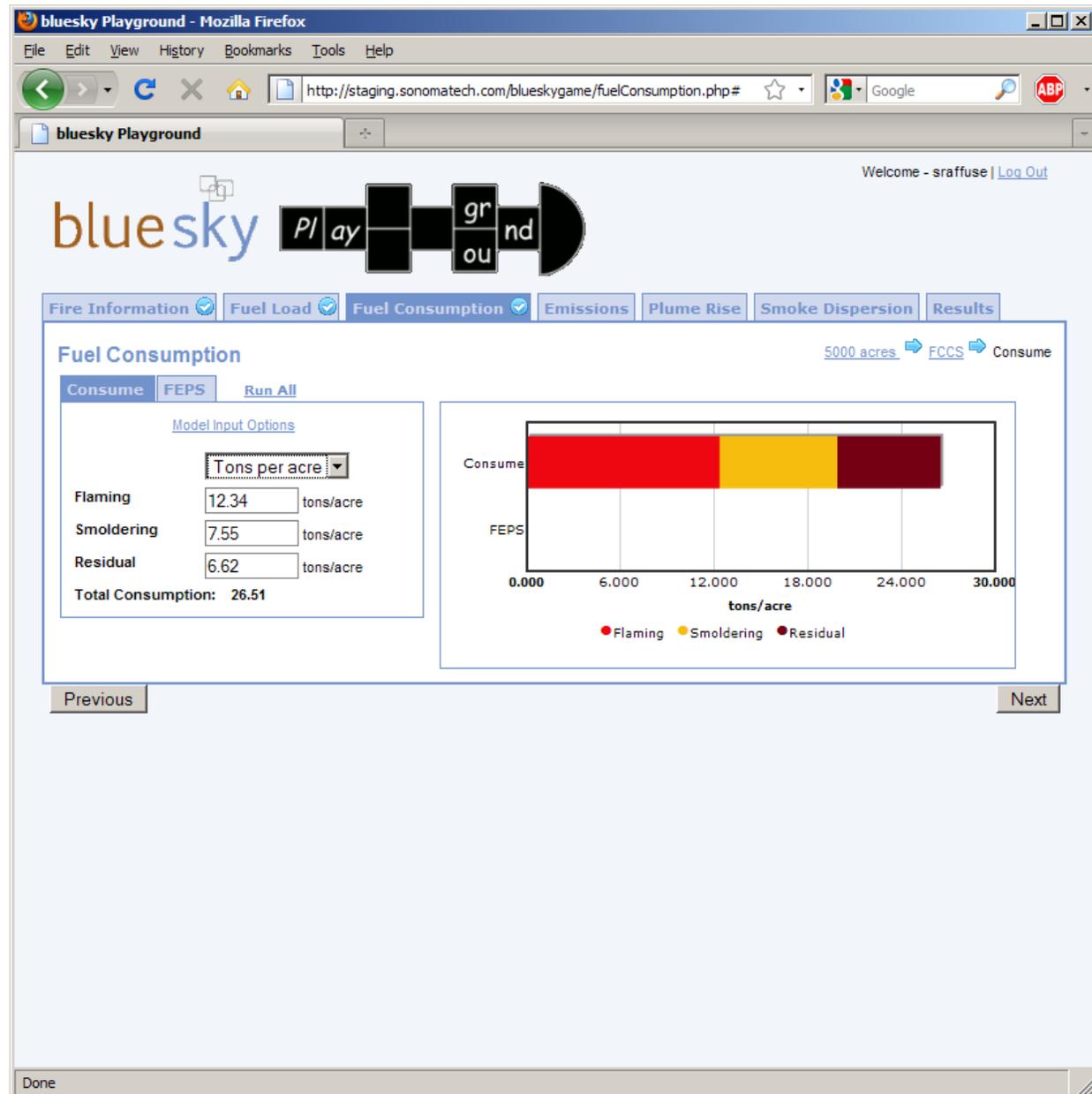
The user can run alternate models for the current step by either clicking on the model tabs or clicking Run All.



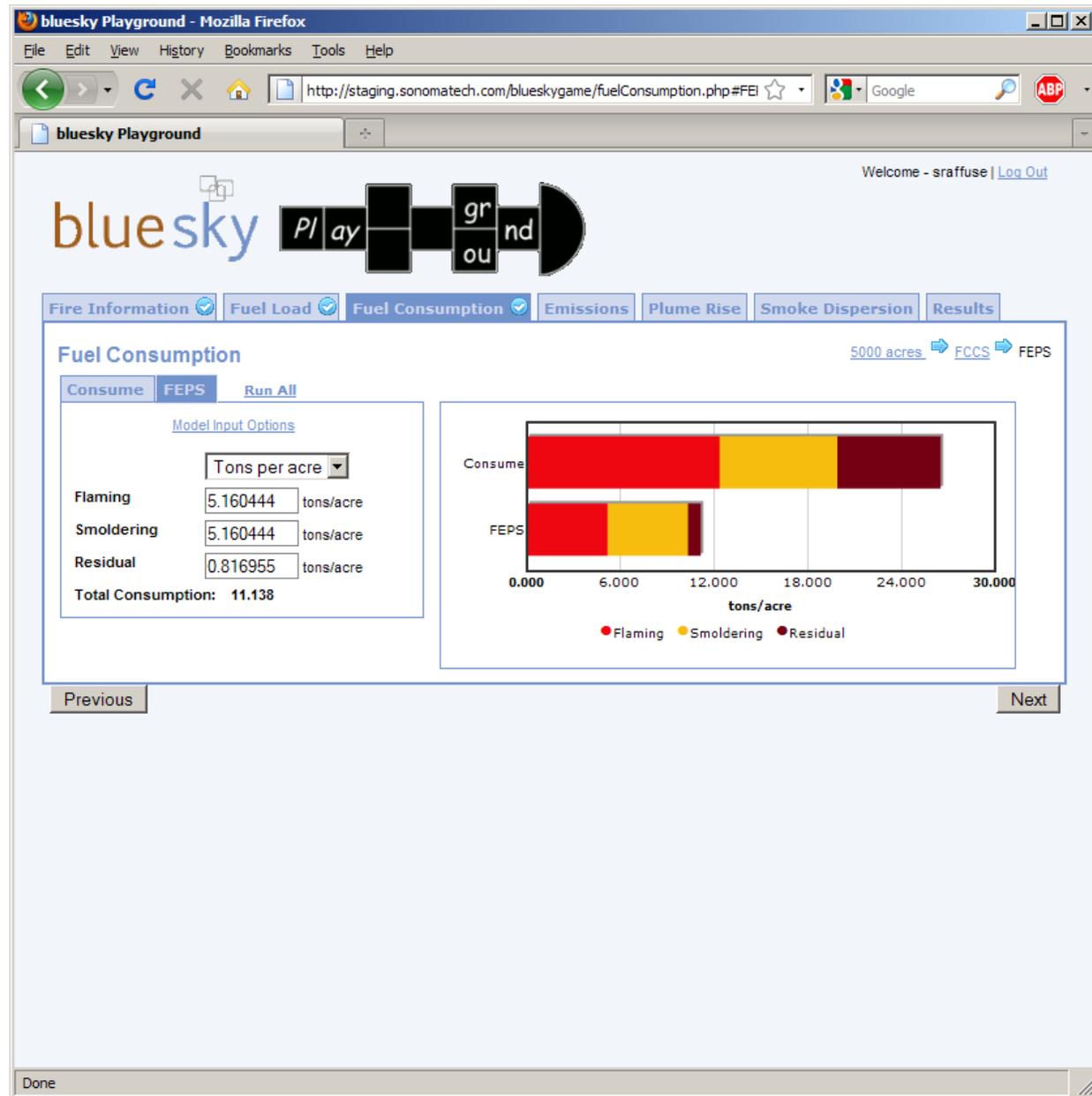
Model results are dynamically added to the output graph as new models are run. Changing the output values manually will also update the graphs.

The currently selected model, which will be used as the input to the next step, is highlighted in the model tabs and also shown in the pathway indicator.

The Fuel Consumption screen is similar to the Fuel Load screen. The output graph shows consumption per acre broken out by combustion phase.



In this example, the user has selected an alternate Fuel Consumption model (FEPS).



The Emissions screen is similar to the Fuel Consumption and Fuel Load screens. Two graphical outputs are available. This screen shows total emissions broken out by pollutant.

bluesky Playground

Welcome - sraffuse | [Log Out](#)

Fire Information Fuel Load Fuel Consumption **Emissions** Plume Rise Smoke Dispersion Results

5000 acres FCCS FEPSConsumption FEPS

**Emissions**

FEPS EPM Run All

Model Input Options

| Parameter         | Value    | Unit         | Display                             |
|-------------------|----------|--------------|-------------------------------------|
| Heat              | 6591808  | Million BTUs | <input type="checkbox"/>            |
| PM <sub>2.5</sub> | 684.78   | tons         | <input checked="" type="checkbox"/> |
| PM <sub>10</sub>  | 808.04   | tons         | <input checked="" type="checkbox"/> |
| CO                | 8130.83  | tons         | <input type="checkbox"/>            |
| CO <sub>2</sub>   | 84184.06 | tons         | <input type="checkbox"/>            |
| CH <sub>4</sub>   | 393.41   | tons         | <input checked="" type="checkbox"/> |
| NO <sub>x</sub>   | 89.56    | tons         | <input checked="" type="checkbox"/> |
| NH <sub>3</sub>   | 133.03   | tons         | <input checked="" type="checkbox"/> |
| SO <sub>2</sub>   | 54.56    | tons         | <input checked="" type="checkbox"/> |
| VOC               | 1912.34  | tons         | <input type="checkbox"/>            |

tons

900.00  
720.00  
540.00  
360.00  
180.00  
0.00

PM<sub>2.5</sub> PM<sub>10</sub> CH<sub>4</sub> NO<sub>x</sub> NH<sub>3</sub> SO<sub>2</sub>

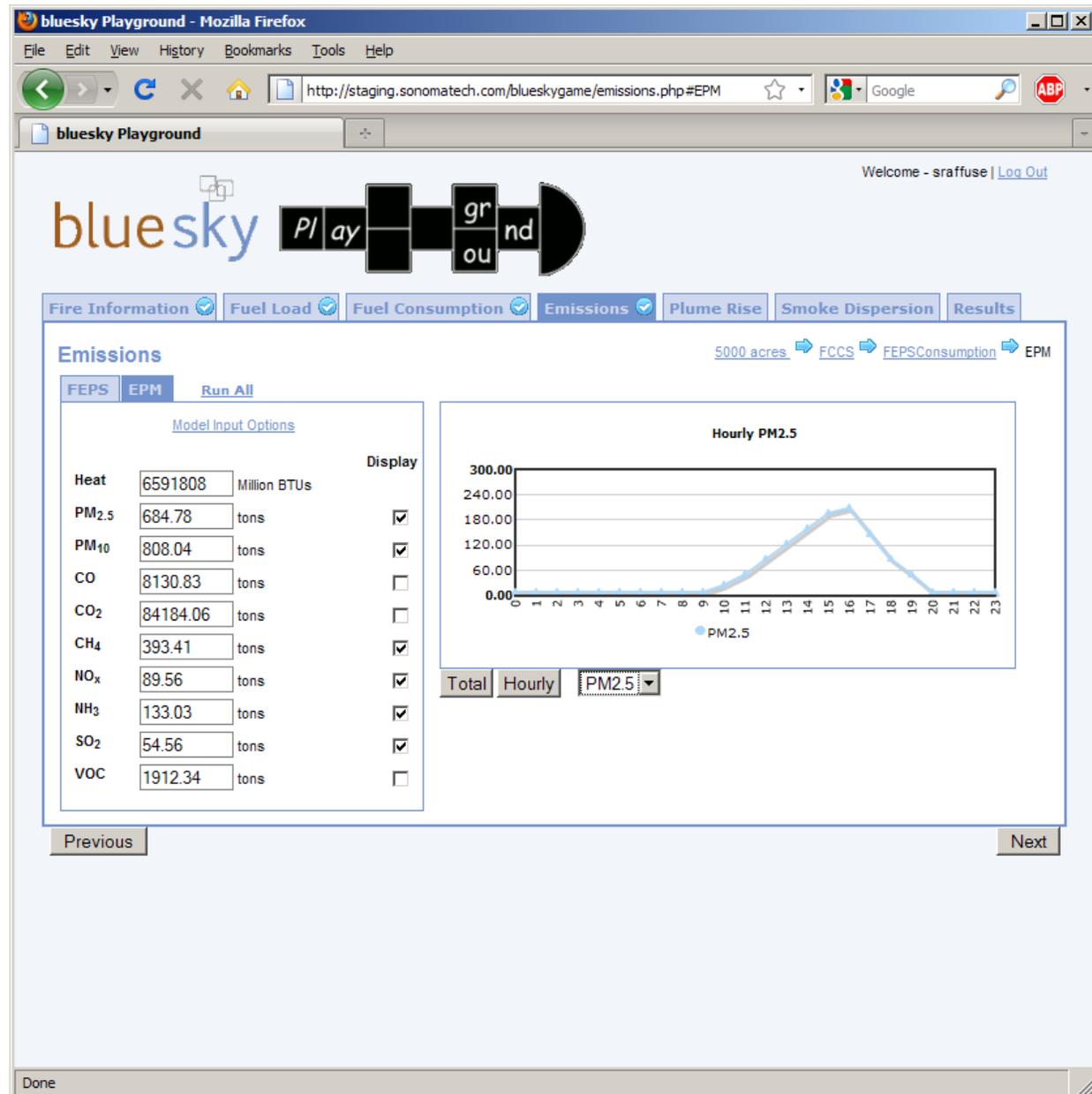
FEPS EPM

Total Hourly

Previous Next

Done

This screen shows a time series of hourly emissions. Only one pollutant from one model can be shown at a time on the time series graph.



Plume Rise is how high the smoke plume is lofted into the atmosphere do to buoyancy and vertical winds. BlueSky models plume rise with a plume top and plume bottom. The graph shows the hourly modeled plume top and bottom. Individual hourly values can be edited in the spreadsheet on the left.

The screenshot shows the BlueSky Playground interface in a Mozilla Firefox browser. The page title is "bluesky Playground - Mozilla Firefox" and the URL is "http://staging.sonomatech.com/blueskygame/plume.php#WRAP". The interface includes a navigation menu with options like "Fire Information", "Fuel Load", "Fuel Consumption", "Emissions", "Plume Rise", "Smoke Dispersion", and "Results". The "Plume Rise" section is active, showing a graph titled "Plume Top and Bottom" and a data table.

The graph shows the hourly modeled plume top and bottom. The Y-axis is labeled "Feet" and ranges from 0.00 to 2,000.00. The X-axis is labeled "Hour" and ranges from 0 to 23. The legend indicates that the dark grey area represents the "Plume Top" and the light grey area represents the "Plume Bottom".

The data table shows the following values:

| Hour | Plume Bottom (Feet) | Plume Top (Feet) |
|------|---------------------|------------------|
| 4    | 0.67                | 1.78             |
| 5    | 0.67                | 1.78             |
| 6    | 0.67                | 1.78             |
| 7    | 0.67                | 1.78             |
| 8    | 2.67                | 7.11             |
| 9    | 7.41                | 19.75            |
| 10   | 29.63               | 79               |
| 11   | 118.51              | 316.02           |
| 12   | 362.93              | 967.8            |
| 13   | 474.02              | 1264.07          |
| 14   | 599.94              | 1599.83          |

Smoke Dispersion is the final step in the BlueSky modeling framework. BlueSky Playground provides a map viewer for reviewing smoke dispersion output. Ground level PM<sub>2.5</sub> concentrations are shown.

Users can select the length of time to model. Longer runs take longer to return results.

The user can also adjust the map by changing the central latitude and longitude and by specifying the map width. Clicking the remap button will tell BlueSky to make new maps.

BlueSky dispersion output is hourly. Each hour is shown as an individual image. The hour shown can be changed using the controls at the top of the image. Clicking the Animate button will cycle through the images automatically. It may take a few minutes for all of the images to be downloaded to the user's local cache so they appear smoothly.

The screenshot shows the BlueSky Playground web interface in a Mozilla Firefox browser. The browser address bar shows the URL: <http://staging.sonomatech.com/blueskygame/smoke.php>. The page title is "bluesky Playground". The interface includes a navigation menu with tabs for "Fire Information", "Fuel Load", "Fuel Consumption", "Emissions", "Plume Rise", "Smoke Dispersion", and "Results". The "Smoke Dispersion" tab is selected. Below the navigation menu, there are several model configuration options: "5000 acres", "FCCS", "FEPSConsumption", "EPMEmissions", "WRAPPlumeRise", and "CALPUFF". The "CALPUFF" model is selected. The "Smoke Dispersion" section contains a control panel with the following settings: "Hours To Model" set to 24, "Center of Map" with Longitude: -114.961 and Latitude: 45.46, and "Map Width" set to 299 miles. There is a "Remap" button. Below the control panel is a legend for PM<sub>2.5</sub> concentrations in  $\mu\text{g}/\text{m}^3$ , with color-coded ranges: 2-10 (dark green), 11-25 (green), 26-40 (light green), 41-60 (yellow), 61-80 (orange), 81-175 (red), 176-300 (dark red), 301-500 (purple), and >500 (dark purple). The main map area shows a geographical map with a smoke plume dispersion output. The plume is shown as a series of green and yellow rectangular blocks, indicating the concentration of PM<sub>2.5</sub> at different locations. The map includes a "Previous" button on the left and a "Next" button on the right. At the top of the map area, there are controls for "Animate", navigation arrows, and a "Hour" dropdown menu set to 15. The browser status bar at the bottom shows "Done".

At any time the user can go back to a specific modeling step and change the parameters. All modeling steps that follow the changed step will be cleared as their results will no longer be valid. When the user goes forward again, all intermediate steps that need to be run are executed.

Once the user is satisfied with their choices, they can review them on the Results page, which shows the modeling pathway and the changes the user made.

If the user wishes, they may save the run parameters.

In future versions, the user will be able to print a summary of the run or load a previously saved run.

