



Jun 18th, 2:00 PM - 3:20 PM

Fronting Integrated Scientific Web Applications: Design Features and Benefits for Regulatory Environments

Jonathan Flaishans

Oak Ridge Institute for Science and Education, flaishans.jonathan@epa.gov

Tao Hong

Oak Ridge Institute for Science and Education, hong.tao@epa.gov

Marcia Snyder

Oak Ridge Institute for Science and Education, snyder.marcia@epa.gov


Chancellor Pascale

Johns Hopkins University APL, chancellor.pascale@jhupl.edu

Thomas S. Purucker

U.S. Environmental Protection Agency, purucker.tom@epa.gov

Follow this and additional works at: <https://scholarsarchive.byu.edu/iemssconference>

 Part of the [Civil Engineering Commons](#), [Data Storage Systems Commons](#), [Environmental Engineering Commons](#), [Hydraulic Engineering Commons](#), and the [Other Civil and Environmental Engineering Commons](#)

Flaishans, Jonathan; Hong, Tao; Snyder, Marcia; Pascale, Chancellor; and Purucker, Thomas S., "Fronting Integrated Scientific Web Applications: Design Features and Benefits for Regulatory Environments" (2014). *International Congress on Environmental Modelling and Software*. 1.

<https://scholarsarchive.byu.edu/iemssconference/2014/Stream-F/1>

This Event is brought to you for free and open access by the Civil and Environmental Engineering at BYU ScholarsArchive. It has been accepted for inclusion in International Congress on Environmental Modelling and Software by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

Fronting Integrated Scientific Web Applications: Design Features and Benefits for Regulatory Environments

**FLAISHANS, JONATHAN¹; HONG, TAO¹; SNYDER, MARCIA¹; PASCALE, CHANCELLOR²;
PURUCKER, S. THOMAS³**

¹ Oak Ridge Institute for Science and Education, Flaishans.Jonathan@epa.gov, Hong.Tao@epa.gov,
Snyder.Marcia@epa.gov

² Johns Hopkins University APL, Chancellor.Pascale@jhuapl.edu

³ U.S. Environmental Protection Agency, Purucker.Tom@epa.gov

Abstract: Integrated decision support systems for regulatory applications benefit from standard industry practices such as code reuse, test-driven development, and modularization. These approaches make meeting the federal government's goals of transparency, efficiency, and quality assurance more attainable, while facilitating science module updates and incorporation of new modules. The übertool is a web-based dashboard suite of ecological risk assessment models supported by the United States Environmental Protection Agency that provides a cross-platform implementation for users. Its web-based approach provides users with a common interface to models developed in differing formats ranging from simple spreadsheet calculators to platform-dependent compiled executables. The dashboard combines features of models into a unified, web-based experience by utilizing a template-based, modular design. The modular structure allows coupling of multiple models with minimal code and the use of native source code with Python wrappers and RESTful servers. Porting models with an existing user base and graphical user interfaces to a web-based application raises issues related to recreating a familiar user experience within the templated design. The übertool compensates for this by using JavaScript, jQuery, and the Django web framework to enhance the basic Python modular structure of the dashboard. This added flexibility allows existing model users to easily adapt to the new web-based environment and allows developers to effectively implement key algorithmic components that make each model unique.

Keywords: Model dashboard; web applications; Python; front end design; user experience; coupled models; code templating; graphical user interfaces

1 INTRODUCTION

The United States Environmental Protection Agency (USEPA) is tasked with regulating pesticide use to protect the environment, consumers, and endangered and threatened species (National Research Council 2013). A crucial aspect of regulation is modelling pesticide exposure and its potential risk to organisms and the environment. Modelling pesticide risk utilizes numerous models developed from an ever-increasing body of scientific research. Over time the tools available for scientific models have changed, leaving a legacy of models implemented in a variety of computer languages (e.g., Fortran 95/2003, Microsoft .NET, etc.) and other software (e.g., Microsoft Excel). While these models are still relevant to the regulatory process, they cannot be easily integrated or updated as the science evolves. These limitations are addressed by integrating existing and developing pesticide risk assessment models in the "übertool." The übertool is a web-based dashboard suite of ecological risk assessment model interfaces supported by the USEPA. Benefits from integrating seemingly disparate model formats include code reuse and maintainability, seamless sharing of model inputs and outputs, and simple batch running of multiple risk assessment models. Additionally, the übertool is publically accessible as a cloud-based service hosting pesticide risk assessment models, helping achieve transparency goals set forth by Orszag (2009) for United States departments and agencies including the USEPA.

2 DASHBOARD DESIGN

The übertool uses a modular design implemented by Python wrapper code, the Django web framework, and a set of text and HTML templates, creating a reusable code structure to front each model in a predictable manner (Amol 2014). Model web pages are represented by a set of Python modules that correspond to the web application's description, input, output, algorithms, references, QA/QC, batch, and history pages as shown in Figure 1. These front end modules are coupled with text and HTML templates to dynamically generate the web pages. Additional back end modules run processes specific to each model. Example back end modules include model parameters, Django template rendering, and connections to a RESTful server for models running in native code. This design facilitates rapid development of each web application comprising the übertool and provides hooks to link models with minimal coding.

The suite of models currently in the übertool is chosen based on the USEPA's requirement to regulate pesticides. These models are integrated into the übertool by creation of their Python model module. Simpler models such as those based on spreadsheet calculators are ported to pure Python modules; whereas, models currently in development and/or based on legacy code such as Fortran 95/2003 are left in their native code and a Python wrapper module is utilized. In the case of the latter, the model is executed via a RESTful server where the native code is hosted on either an Instance as a Service (IaaS) or a Platform as a Service (PaaS). The result is a cross-platform, web-based pesticide risk assessment dashboard (Purucker, S.T. et al. 2014).



Figure 1. Navigation tabs representing the front end Python modules for each übertool model.

2.1 Description, Algorithms, and References Pages

The description, algorithms, and references pages represent the basic dynamic content generating Python modules in the übertool. Each page loads a sequence of HTML templates that build the resulting pages. All subsequent front end modules load a similar set of templates and module-specific functions. The templates partition the HTML into easily managed, logical sections of code such as the top header, model navigation, übertool navigation, and footer of the web pages; these templates are hard-coded, reusable blocks of code. Variables can be coded into the templates with Django, allowing Python modules to pass model-specific parameters to the templates increasing their flexibility. An example of this is the model name visible on each model page.

2.2 Input and Output Pages

Single model runs on the übertool are handled by the input and output pages. Figure 2 details how the front end and back end communicate during model runs. The input page imports the parameters module, where model input variables are stored, and it is processed by Django to render the HTML input form. An optional "tooltips" dictionary can be loaded at this time as well. Tooltips further explain specific model inputs to users and are handled through jQuery and Python dictionaries (Figure 3). Models with more complex and/or numerous inputs are further processed with jQuery to create a dynamic, user-friendly interface such as tabbed navigation (Figure 5). Submitting the input form calls the output module, where user inputs are processed into their respective Python variables and passed to the model module. The model module executes the model's methods and returns the output to the output module. If the user chooses to save the model run on the input page, the output is saved to the MongoDB server for later retrieval. The tables module formats the output into HTML tables using Django templates, and the formatted output page is displayed to the user. Optionally, the output page can be exported in PDF and HTML formats for users to store locally and easy distribute (Figure 4).

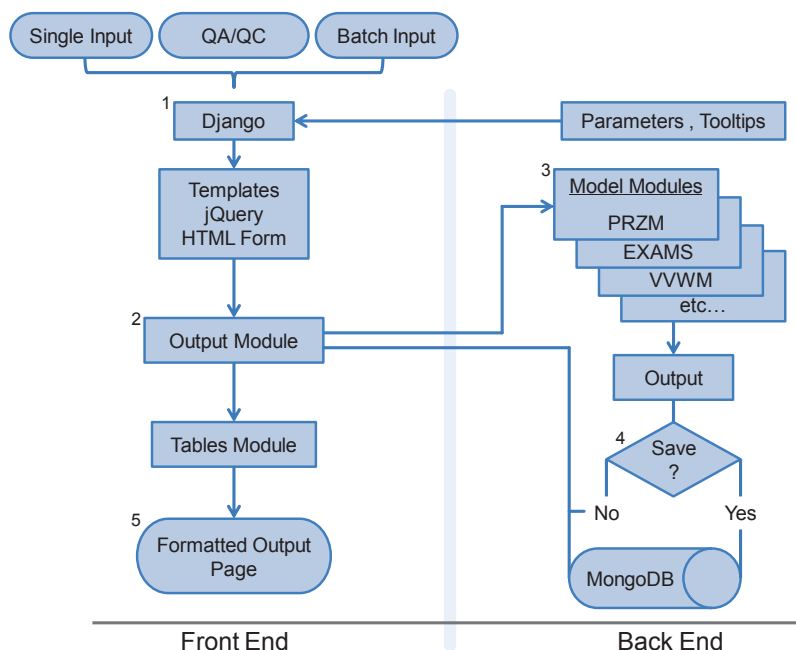


Figure 2. Process of a model run on the übertool. 1) Input page formatting; 2) Inputs being sent to model module; 3) Output processing by model module; 4) Optional saving of output to MongoDB; 5) Generation of output page.

Solubility:	<input type="text" value="0.128"/>	Chemical solubility in water @25°C; mg/L
Mammalian LD₅₀ (mg/kg-bw):	<input type="text" value="5000"/>	
Species of the tested mammal:	<input type="text" value="Make a selection"/>	

Figure 3. Showing a “tooltip” (right) for the “Solubility” input on a model input page.

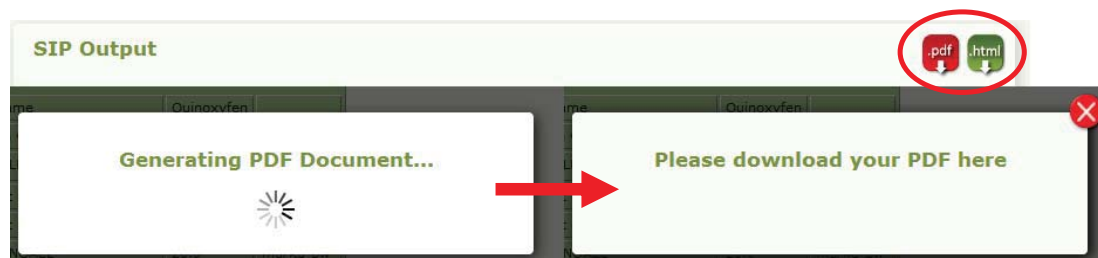


Figure 4. Illustration of the PDF export feature on output pages. The user clicks the “.pdf” or “.html” buttons (top right) to choose which format the output page is exported to save the results locally.

The modular design of the übertool allows for code reuse between models. Figure 2 shows the Single Input, QA/QC, and Batch Input modules reuse the parameters, tooltips, and model back-end modules. The model modules can be linked together to allow for the daisy-chaining of models. Examples of this are described in the übertool section of this paper. If the model being run is kept in native code, the model module acts as a Python wrapper to communicate with a Python-based RESTful server hosting the native model code as further explained by Tao, H. et al. (2014).

2.3 Batch Processing Pages

Batch processing of individual models is handled by the batch input and batch output modules. The batch input module is similar to the input page's module but instead of requiring user input from a HTML form, the inputs are delivered in a comma-delimited values (CSV) file. The user is provided with a template CSV file for download containing an ordered list of model inputs, with each row representing a new batch run. The file can be run with the default values or edited to meet the user's needs. The user is required to select the CSV file to upload, and it is uploaded upon model submission to the server, where the batch output module loops over the CSV. The first row of the CSV is parsed and the ordered inputs are appended to Python lists before being passed as parameters to the model object. This process is repeated until all the rows in the CSV have been parsed and sent to the model object. The output values are stored in Python lists with indexes matching their iteration order. The formatted HTML output page is generated in the same way as a single run but is looped to account for each model iteration. Additionally, the batch output module provides a "Batch Summary" section on the output page displaying a statistical summary of the iterations including mean, minimum, and maximum values of inputs and outputs. The user can export all of the information on the batch output page as PDF and HTML formats or choose to save it in the cloud for later retrieval.

2.4 QA/QC Pages

The QA/QC pages are a parameterized unit test for the web-applications. Each model in the übertool suite is tested with reasonable inputs provided by the model creators and/or primary users. Test cases can range from a single set of inputs for the simplest models to multiple scenarios for models with a larger range of input options. The QA/QC module parses a non-editable CSV file stored on the server containing the test case(s) of model inputs and their expected outputs. The inputs are passed to the model object, and the test case is run through the model's methods. The resulting outputs are displayed on the QA/QC page alongside their "reference" outputs to verify the model logic. The QA/QC section of each übertool model is intended to verify the numerical output of web-applications against their native model format under a known test case. Its purpose is not for scientific validation of algorithms which comprise the risk assessment model. Further unit testing is handled with continuous integration solutions to monitor the rapid development and release process of the übertool (Kawauchi 2014).

2.5 History Page

The history module shows the user a list of previous model runs. The results of each model run are saved at the user's discretion to a MongoDB database in the cloud and can be retrieved by the user's request. The retrieved results display a fully functioning version of the original output page, including the PDF and HTML export options and any graphs or charts. This feature is available to users by checking a box on the input page before submitting the model inputs and includes the ability to add metadata describing the model run. The user's saved model runs are displayed in chronological order as hyperlinks on the history page. Clicking a model run directs the user to the original output page which is pulled from the MongoDB database.

2.6 Existing User Base and User Experience

The suite of models hosted as web applications by the übertool have a pre-existing user base of agency regulators and research scientists who are accustomed to native versions of the risk assessment models. To accommodate these users the templated design structure of the übertool is augmented with jQuery and JavaScript to retain the user experience (UX) of current graphical user interfaces (GUI). A side-by-side comparison of native and web-application versions of an example model is shown in Figure 5. Here tabbed navigation of model inputs is created using jQuery to mimic the Visual Basic (VB) GUI. This allows the übertool to preserve the UX expected by users while efficiently utilizing the screen real estate available to the web application within the templated design structure.

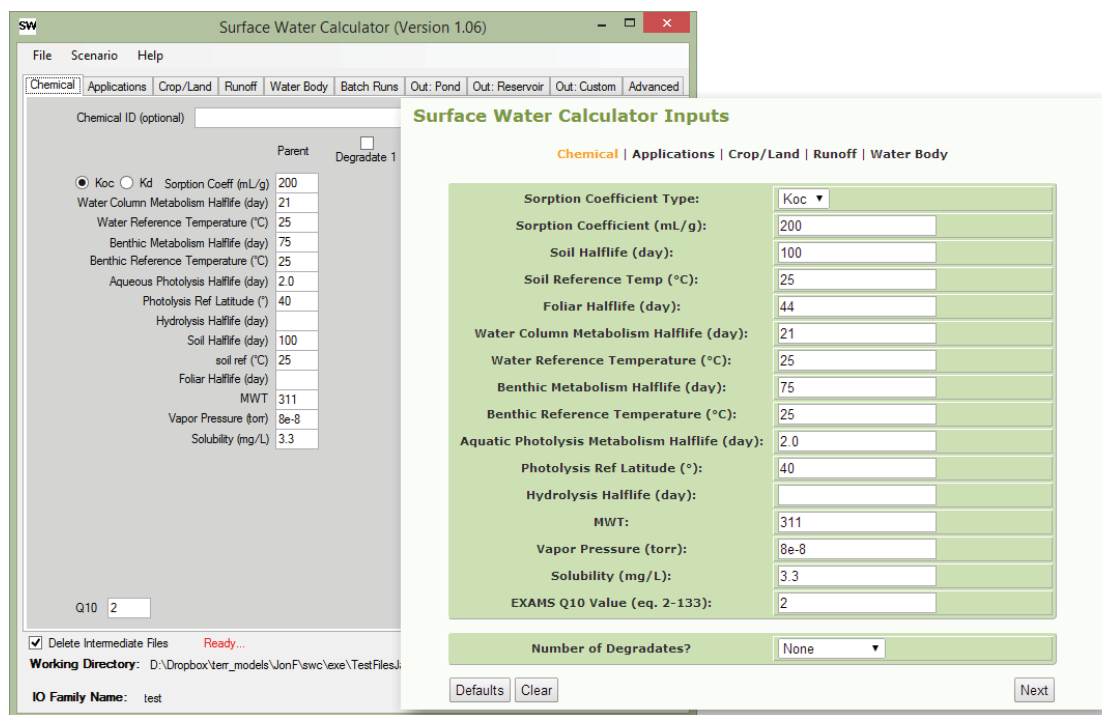


Figure 5. Comparison of a native model GUI (left) and the übertool web-application GUI (right), enhanced using jQuery and JavaScript.

3 THE ÜBERTOOL

The benefit of porting a range of pesticide risk assessment models into an integrated web-application environment lies in the ability to easily couple varying model formats and provide platform-independent access to them. The Python wrapper code applied to each model allows for code reuse and inputs and outputs to be shared and daisy-chained between each other. Many pesticide models handle specific processes and benefit with being linked to others to assess potential environmental risk as discussed by Villeneuve and Garcia-Reyero (2011).

An example of a coupled model used by the USEPA for pesticide risk assessment includes the Pesticide Root Zone Model (PRZM) and the Exposure Analysis Modelling System (EXAMS) (Carsel, R. F. et al. 1985; Burns, L. A. et al. 1982). PRZM models the transport of pesticide leachate from the root zone through the soil, while EXAMS handles the fate of said pesticide in small water bodies such as an adjacent farm pond. When coupled these two models provide complementing capabilities to estimate pesticide risk; therefore, an executable GUI was created to link these Fortran 95/2003 models called EXAMS - PRZM Exposure Simulation Shell (EXPRESS), which is only available as a Microsoft Windows application (Burns 2006). The modular design of the übertool allows the two models to be easily coupled as a platform-independent web application using Python wrapper code and a RESTful server (Hong, T. et al. 2014). The coupled version of the EXAMS - PRZM is offered alongside the standalone versions without duplicating source code. By reusing code and avoiding code duplication the übertool promotes rapid development of web applications with little overhead that can quickly adapt to updates to as well as new science modules.

As science progresses and the models are updated, it can take time for the updated models to be approved for regulatory use. The übertool can host all model versions side by side, providing users easy access to all relevant model versions. As example, currently in development is a new version of PRZM, PRZM5, and it is available alongside the previous version on the übertool. While the previous version of PRZM is currently being used for pesticide regulation modelling, the availability of the newest version provides the public with access to the version that will likely replace the current.

Additionally, PRZM5 is intended to be coupled with a new water body model named the Variable Volume Water Body Model (VWWM). Coupled together they are available as the Surface Water Calculator (SWC). The SWC is currently in development as a Microsoft Windows executable like EXPRESS was before it. With minimal code the übertool couples these two Fortran 95/2003 native models in a Python wrapper that can be accessed platform-independently, while retaining the GUI of the native VB wrapper (Figure 5). This same technique can be applied to any of the übertool modules allowing for the suite of models to reuse code and be easily linked as needed.

4 CONCLUSIONS

The übertool dashboard provides an integrated decision support system for pesticide regulators that is cross-platform and cloud-based. This provides simultaneous benefits of a unified UX between the suite of models, while retaining the native model GUI, and increasing the transparency of the pesticide regulatory process. Current model users will be familiar with web-application versions and the general public will have access to the same models used in pesticide regulation by the USEPA. The modular Python framework on which the übertool is built allows rapid web application development, including maintaining multiple versions and cross-language communication of the models. Legacy and actively-developing models in native code can be implemented using Python wrapper code and RESTful servers. The suite of models in the übertool can be linked with minimal code allowing the sharing inputs and outputs, creating a flexible cloud-based modelling framework. The übertool aims to unify the fragmented ecosystem of risk assessment modelling by simplifying and increasing its accessibility to the public through the cloud.

ACKNOWLEDGMENTS

The authors would like to acknowledge the Oak Ridge Institute for Science and Education (ORISE) as well as the faculty and staff at the USEPA Office of Research and Development Ecosystems Research Division in Athens, GA.

REFERENCES

- Amol, 2014. Web Frameworks for Python. From: <https://wiki.python.org/moin/WebFrameworks>
- Burns, L. A. et al. 1982. Exposure analysis modelling system (EXAMS): user manual and system documentation. EPA-600/3-82-023, Environmental Protection Agency, Washington, DC.
- Burns, L. A. 2006. User manual for EXPRESS, the “EXAMS-PRZM Exposure Simulation Shell”. EPA/600/R-06/095, Environmental Protection Agency, Washington, DC.
- Carsel, R. F. et al. 1985. The pesticide root zone model (PRZM): A procedure for evaluating pesticide leaching threats to groundwater. *Ecological modelling*, 30(1), 49-69.
- Carsel, R. F. et al. 1998. PRZM-3: A model for predicting pesticide and nitrogen fate in the crop root and unsaturated soil zones; Users Manual for Release 3.0. Environmental Protection Agency, Washington, DC.
- Django Software Foundation, 2014. Django Python Web Framework. From <https://www.djangoproject.com/>.
- Hong, T et al. 2014. Back-end science model integration for ecological risk assessment. Proceedings from 7th Intl. Environ. Model. Softw. Society. San Diego, CA USA.
- jQuery Foundation, The, 2014. jQuery. From: <http://www.jquery.com>
- Kawauchi, K. 2014. Meet Jenkins. From: <https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins>
- National Research Council, 2013. Assessing risks to endangered and threatened species from pesticides. The National Academies Press
- Orszag, P.R. 2009. Memorandum for the heads of executive departments and agencies: Open government directive. United States Govt
- Purucker, S.T et al. 2014. Decision support and web-based implementation of algorithms for the ecological assessment of pesticides. Proceedings from 7th Intl. Environ. Model. Softw. Society. San Diego, CA USA.
- Villeneuve, D.L. and Garcia-Reyero, N. 2011. Predictive ecotoxicology in the 21st Century. *Environ Toxicology and Chemistry*. Vol. 30, No. 1.