The Exploratory Modeling Workbench An open source toolkit for exploratory modeling, scenario discovery, and (multi-objective) robust decision making

Jan H. Kwakkel

Delft University of Technology, j.h.kwakkel@tudelft.nl

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


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.
The Exploratory Modeling Workbench
An open source toolkit for exploratory modeling, scenario discovery, and (multi-objective) robust decision making

Jan H. Kwakkel
Faculty of Technology, Policy and Management, Delft University of Technology, The Netherlands
(J.H.Kwakkel@tudelft.nl)

Abstract: There is a growing interest in model-based decision support under deep uncertainty, reflected in a variety of approaches and techniques being put forward in the literature. A key idea shared among these various approaches and techniques is the use of models for exploratory rather than predictive purposes. Exploratory modeling aims at exploring the implications for decision making of the various presently irresolvable uncertainties. This is achieved by conducting series of computational experiments that cover the various ways in which the various uncertainties might be resolved. This paper presents an open source library for performing exploratory modeling. This exploratory modeling workbench is implemented in Python. It is designed to (i) support the generation and execution of series of computational experiments, including support for parallelization on a single machine or high performance cluster; and (ii) support the visualization and analysis of the results from the computational experiments. The exploratory modeling workbench enables users to easily perform exploratory modeling with existing simulation models, identify the policy relevant combinations of uncertain factors, assess the efficacy of policy options to address these uncertain factors, and thus iteratively improve candidate strategies.

Keywords: Deep uncertainty, exploratory modeling, robust decision making, scenario discovery

1 Introduction

There is a rapidly growing interest in the challenge of offering decision support under deep uncertainty. To address the challenge of supporting decision making under deep uncertainty, a wide variety of approaches, techniques, and associated tooling has been put forward in recent years. Examples of approaches for supporting decision making under deep uncertainty include Robust Decision Making (Groves and Lempert 2007), Info-Gap decision theory (Ben Haim 2001), Decision Scaling (Brown, Ghile et al. 2012), Dynamic Adaptive Policy Pathways (Haasnoot, Kwakkel et al. 2013), and Scenario Discovery (Bryant and Lempert 2010). In parallel to these various approaches, there has been substantial development in software tools to support the application of these approaches. Examples include the closed-source Computer Assisted Reasoning software used by the RAND Corporation, the open source Scenario Discovery Toolkit (Bryant 2014), and the openMORDM library (Hadka, Herman et al. 2015).

From an analytical perspective, all model based approaches for supporting decision making under deep uncertainty are rooted in the idea of exploratory modeling (Bankes 1993, Bankes, Walker et al. 2013). Traditionally, model based decision support is based on a predictive use of models. Simulation models are used to predict future consequences and decisions are optimized in light of this. Under deep uncertainty, this predictive use of models is highly misleading. Instead, models should be used in an exploratory fashion, for what-if scenario generation, for learning about system behavior, and for the identification of critical combinations of assumptions that make a difference for policy (Weaver, Lempert et al. 2013).

In this paper, we introduce the exploratory modeling workbench. The exploratory modeling workbench is an open source library for performing exploratory modeling. By extension, the workbench can be used for the various model-based approaches for decision making under deep uncertainty. In scope, it is fairly similar to the openMORDM toolkit (Hadka, Herman et al. 2015), although there are some
interesting differences in the approach taken to supporting exploratory modeling. The workbench is implemented in Python. It currently runs under Python 2.7, although an increasing part of the code is Python 3 compliant. The code is available under an open source license, and can be found online on Github.

The remainder of this paper is structured accordingly. Section 2 introduces a theoretical framework that underpins the design of the exploratory modeling workbench. Section 3 discusses the design and key implementation details of the workbench, as well as a compare and contrast with some of other available open source tools for mode-based decision support under deep uncertainty. Section 4 demonstrates the use of the workbench for the Lake Problem (Lempert and Collins 2007, Hadka, Herman et al. 2015, Singh, Reed et al. 2015, Ward, Singh et al. 2015). Section 5 contains some concluding remarks and a discussion of future extensions.

2 Framework

A variety of analytic model-based approaches have been put forward in recent years to address the problem of deep uncertainty in model based decision support. These include, amongst others, Robust Decision Making, Multi-Objective Robust Decision Making, Dynamic Adaptive Policy Pathways, Info-Gap Decision Theory, Real Options Analysis, and Decision Scaling. An idea common to these various approaches is to use models to explore the consequences of the resolution of the various uncertain factors, rather than using models for predictive purposes. That is, these approaches built on the idea of exploratory modeling (Bankes 1993, Bankes, Walker et al. 2013, Weaver, Lempert et al. 2013).

There are three key ideas that jointly underpin the design of the exploratory modeling workbench. These are the XLRM framework, running simulation models as if they are a function, and a taxonomy of robustness frameworks. We now elaborate these three ideas and how they influence the design of the workbench.

The first idea which underpins the workbench is the system diagram (Walker 2000), or XLRM framework (Lempert, Popper et al. 2003). This diagram is shown in Figure 1, where X stands for the exogenous or external factors. These are factors that are outside the control of the decision-makers. L stands for policy levers. R stands for relationships inside the system, and M stands for performance metrics or outcomes of interest. In the workbench, this framework is used to structure the exploratory modeling. Exogenous factors are presented as uncertainties, policy levers are presented as policies, model structures represent relationships, and performance metrics are presented by outcomes.

The second key idea behind the design of the workbench is the idea of running a simulation model as if it where a subroutine. Adopting the XLRM notation, a simulation model is simply a function called with a set of parameters $X$ and $L$. The return of the function is a set of outcomes of interest $M$. So

$$f(X, L) = M$$

In the workbench, there is an interface between the actual simulation model and the workbench. This interface enables running the simulation model as if it where a function. This enables the workbench to interface with any modeling or simulation package that exposes some kind of API. For example,
Vensim, a package for System Dynamics modeling exposes a DLL. The workbench can use this DLL to perform exploratory modeling with simulation models implemented in Vensim.

By combining the first two ideas, it is possible to use the workbench to explore uncertainty pertaining relationships within the model. This can be done either by parameterizing the uncertain relationship through some categorical variable where each category represents one possible realization of the uncertain relationship, or by working with multiple distinct simulation models. The workbench can be used for both, and implementation concerns dictate the most efficient way of exploring model uncertainty.

The third foundational idea for the workbench is a taxonomy of robustness frameworks. Most recently, Herman, Reed et al. (2015) presented such a taxonomy. The workbench evolved separately from this specific taxonomy, but is coherent with it. In the taxonomy of Herman, Reed et al. (2015), there are four components:

1. **Generation of policy options**: The generation of policy options can be achieved either through multi-objective search, the policy options can be pre-specified, or the policy options can emerge from iterative vulnerability analysis.

2. **Generation of states of the world**: States of the world are the scenarios against which candidate policy options are evaluated. These can be pre-specified, or generated using design of experiment approaches.

3. **Robustness evaluation**: There is a wide literature on robustness metrics. Broadly speaking, three families of metrics can be identified: (i) satisficing metrics; (ii) regret metrics; and (iii) descriptive statistics of the distribution of outcomes over the states of the world.

4. **Vulnerability analysis**: vulnerability analysis aims at identifying the relative influence of the various uncertain factors on policy robustness. This can be done through factor prioritization based approaches as found in the sensitivity analysis literature, or through scenario discovery.

With respect to generation of alternatives, the workbench can be used with both pre-specified alternatives, but it also possible to use (multi-objective) search in order to identify promising alternatives. Since the workbench offers various techniques for vulnerability analysis, the iterative process of policy design is also supported.

For the generation of states of the world, the workbench offers support for using both user specified states of the world, as well as support for Monte Carlo sampling, Latin Hypercube sampling, and Full Factorial sampling. Users interested in more sophisticated experimental designs can use the SALi library, which can easily be combined with the workbench, as evidenced by the Sobol example that comes with the workbench.

The workbench does not include pre-defined robustness metrics, but it is straightforward to implement any robustness metric in Python as part of the analysis of the results.

For the vulnerability analysis, the workbench offers support for Scenario Discovery using either Classification and Regression Trees or the Patient Rule Induction Method. In addition, it implements a random forest based feature scoring approach, which can be used as an alternative to traditional global sensitivity analysis techniques. Users interested in established sensitivity analysis techniques could combine the SALi library with the workbench.

### 3 The Exploratory Modeling Workbench

The exploratory modeling workbench is implemented in Python. Python is a widely used open source programming language. It is a high-level generic programming language that supports various programming paradigms. Python places a strong emphasize on code readability and code expressiveness. Python is increasingly popular for scientific computing purposes due to the rapidly expanding scientific computing ecosystem available for Python. The workbench extensively uses libraries from this scientific computing ecosystem. Currently, the workbench only supports python 2.7, although more than 80% of the code is already python 3 compliant.

The workbench is composed of four packages: em_framework, analysis, connectors, and util. In addition, there is a folder with a wide variety of examples. The em_framework package contains the core classes and functions of the workbench. These key components are
• **ModelStructureInterface**: an abstract base class that specifies a generic interface to any simulation model
• **ModelEnsemble**: a singleton class that is responsible for performing the computational experiments
• **Uncertainty classes**: the workbench distinguishes between ParameterUncertainty and CategoricalUncertainty. The former is a continuous range of numbers (either floats, or integers). The latter is a set of values.
• **Outcome**: a simple class for specifying the name of an outcome and a Boolean for indicating whether an outcome is a time series or not
• **Support for parallelization**: the workbench offers support for the parallel execution of code either on a single machine or on any standard high performance cluster. This is implemented by creating a layer of abstraction between the multiprocessing and ipyparallel libraries and the workbench. Either library is supported. This implies that if ipyparallel is configured properly, the workbench can be used on a high performance cluster.
• **Data storage**: a basic easily extendable mechanism for storing the results of the computational experiments
• **Samplers**: the workbench offers support for Monte Carlo, Latin Hypercube, and Full Factorial designs. Again, at the root is an abstract base class, making integration of more sophisticated sampling, such as offered by SALib, straightforward

With these components, it is possible to realize the idea of running a simulation model as if it was a function. The parallelization support reduces runtime. The data storage is straightforward, simple, and human readable, and has proven to be sufficient in virtually all use cases.

The analysis package contains a variety of functions and classes for visualization of results, and more in depth analysis. Visualization focuses on displaying time series data, and primarily supports the rapid exploration of the results. The analysis functionality can be decomposed into feature scoring metrics and scenario discovery. Feature scoring is a machine learning alternative to global sensitivity analysis and can be used to get insight into the relative influence of the various uncertain factors on model outcomes. For scenario discovery, both the Patient Rule Induction Method, and Classification and Regression Trees are supported, and employ as much as possible an identical interface.

The connector package contains off the shelve connectors for Excel, Vensim (System Dynamics), and NetLogo (Agent Based Modelling). With these off the shelve connectors, it becomes very easy to perform exploratory modeling on simulation models implemented in these programs. In the trivial case, it is sufficient to only specify the uncertain factors, their ranges or sets of values, and the outcomes. Other simulation packages can easily be added if desired, and proof of concept exist for some. The key class that needs to be provided is a simulation package specific extension to ModelStructureInterface.

The util package contains various utility functions. Most importantly it offers support for logging which functions even when running in parallel on either a single machine or a high performance cluster. It also offers functionality for serializing results and storing them to disk, and loading the results back into memory. The storage format currently is a simple tarbal of csv files, with separate metadata text files. This is a format that is human readable, cross platform, and simple.

4 **The Lake Problem**

The exploratory modeling workbench includes an example folder. This folder contains a variety of examples that demonstrate the functionality of the workbench. Many of these examples have been drawn from published cases. Here, we use the Lake Problem as an example for demonstrating some of the key functionality of the workbench. A repository with the source code and data can be found on GitHub (https://github.com/quakel/lake_problem)

We demonstrate some of the key capabilities of the exploratory modeling workbench using the Lake problem. The lake problem is a stylized and hypothetical decision problem where the population of a city has to decide on the amount of annual pollution it will put into a lake. It the pollution in the lake passes a threshold, it will suffer irreversible eutrophication. A full specification of the model can be...
found in the GitHub repository, including some details on where the presented implantation differs from earlier implementations.

The lake problem is characterized by both stochastic uncertainty and deep uncertainty. The stochastic uncertainty arises from the natural inflow. To reduce this stochastic uncertainty, multiple replications are performed and the average over the replication is taken. Deep uncertainty is presented by uncertainty about the mean $\mu$ and standard deviation $\sigma$ of the lognormal distribution characterizing the natural inflow, the natural removal rate of the lake $\beta$, the natural recycling rate of the lake $q$, and the discount rate $\delta$. Table 1 specifies the ranges for the deeply uncertain factors, as well as their best estimate or default values.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mu$</td>
<td>0.01 – 0.05</td>
<td>0.02</td>
</tr>
<tr>
<td>$\sigma$</td>
<td>0.001 – 0.005</td>
<td>0.0017</td>
</tr>
<tr>
<td>$\beta$</td>
<td>0.1 – 0.45</td>
<td>0.42</td>
</tr>
<tr>
<td>$q$</td>
<td>2 – 4.5</td>
<td>2</td>
</tr>
<tr>
<td>$\delta$</td>
<td>0.93 – 0.99</td>
<td>0.98</td>
</tr>
</tbody>
</table>

4.1 Specifying the model structure interface

The first step in performing exploratory modeling with the workbench is to implement the lake model. This implementation can be split into two separate parts. The first part is the lake model itself, as specified above. We have chosen to implement this as a separate function that takes various parameters as keyword arguments with default values, and returns the four specified outcomes of interest, as well as the concentration of phosphor in the lake over time.

The second part is the specification of the ModelStructureInterface, which makes it possible to perform exploratory modeling on the lake model using the workbench. In order to specify this class, we need to specify the uncertain factors by specifying their names and ranges (or sets of values in case of categorical uncertain factors). We specify the various outcomes of interest including whether it is a time series or not. These two together specify what should go into the model, and what is expected to come out if run_model is called. The method run_model is the key part that we need to specify.

Because of the stochastic uncertainty of the natural inflow, we specify that we want to run the model for N replications. We call the model with the case, or computational experiment, for each replication, and take the average over the replications for each outcome of interest.

In order to be able to explore the performance of various policies, we also implement the model_init method. This method receives a policy, and any other keyword arguments relevant for model initialization. In this case, we can suffice with a simple implementation where we take the decision field from a dictionary and save this to an attribute.

This completes the specification of the ModelStructureInterface, which we can now use for performing exploratory modeling.

4.2 Generation of states of the world and vulnerability analysis

The next step is to use the ModelStructureInterface for performing a first series of experiments. To this end, we instantiate this class, we also instantiate a ModelEnsemble class and set the model interface as the model_structure attribute on the ensemble. For illustrative purposes, we add four candidate pollution strategies to the ensemble. The series of computational experiments will be run for each of these candidate strategies. The four strategies are a constant relies of 0.01, 0.05, 0.01, and a slowly changing discharge that first increases and than decreases. We can run the experiments in parallel, by setting the parallel attribute to True. By default, the ensemble will use all the available cores on the machine, but if desired the user can specify a different number using the processes attribute. We can now perform a series of computational experiments using the perform_experiments method. By default, this method will use Latin Hypercube sampling.
Once the experiments have been performed, we can analyze the results. A first simple visualization is to look at the dynamics over time of the phosphor concentration in the lake, split by policy. It would create substantial visual clutter if we would visualize each of the individual experiments. We find it useful, instead, to visualize the bandwidth of outcomes with some individual traces on top of this envelope. This is straightforward with the lines function.

A second useful visualization in this case is to look at the pair wise scatter plot for each of the four objectives. Again, this can easily be achieved with the pairs_scatter function.

![Figure 2](image)

**Figure 2** Simple visualization of the concentration of phosphor over time (left), and the trade offs between the four objectives (right)

More elaborate analyses involve feature scoring and scenario discovery. Feature scoring is a family of machine learning techniques for identifying the relative importance of various features for a certain outcome or class of outcomes. As an example, we perform scoring on the concentration of phosphor in the lake at the end of the simulation.

The results of this analysis are shown in Table 1. As can be seen, the parameter $b$ is by far the most important factor affecting the terminal value of the concentration of phosphor in the lake. The second and third parameter are the policy and the parameter $q$. Interestingly, the mu and sigma, which together specify lognormal distribution of the natural inflow, have only a limited impact on the final concentration of phosphor.
A second analysis technique is to perform scenario discovery using either the Patient Rule Induction Method (PRIM) or Classification and Regression Trees (CART). Here we show the application of PRIM. The implementation of PRIM which comes with the workbench is designed for interactive use through jupyter notebook. In practice, the code would thus be executed in several steps and the user would make choices based on the results shown. The first step in applying scenario discovery is to specify which scenarios are of interest. Here, we use a simple classification. Any experiment where the concentration in the lake at the end of the runtime is below 1, is of interest.

Figure 3 shows the results for PRIM. On the left, we see the trade off between coverage and density. On the right, we see the identified box, including information on the significance of each of the limits, and the coverage and density of the box. This particular box is the final one on the peeling trajectory shown on the left.

5 Concluding Remarks
This paper has introduced the exploratory modeling workbench. We explained the key ideas that underpin the design of the workbench, and demonstrates some of the key functionality of the workbench using the Lake Problem. This demonstration only shows some of the essential functionality. A reader interested in learning more about the workbench and what it can do is kindly referred to the examples that come with the workbench, as well as the github repository containing the lake model example presented here.

ACKNOWLEDGMENTS

Any particular assistance out of the ordinary may be acknowledged. Please, do not use the numeral for the acknowledgements section. It is not necessary from the organisers’ point of view to record the permission of the author’s organisation to publish the paper or the information contained therein.

---

It is expected that between submission of this paper and the actual conference, the lake model example will be further expanded. Planned expansions include Scenario discovery using CART, demo of SALib integration with a Sobol example, multi-objective optimization of outcomes (the MO stage of MORDM), and multi objective robust optimization.


Herman, J. D., P. M. Reed, H. B. Zeff and G. W. Characklis (2015). "How should robustness be defined for water systems planning under change." Journal of Water Resources Planning and Management 141(10).


Lempert, R. J., S. Popper and S. Bankes (2003). Shaping the Next One Hundred Years: New Methods for Quantitative, Long Term Policy Analysis. Santa Monica, CA, RAND.


