A Flexible Framework for Building Energy Analysis

Elaine T. Hale, Daniel L. Macumber, Evan Weaver, and Diwanshu Shekhar,
National Renewable Energy Laboratory[1]

ABSTRACT

In the building energy research and advanced practitioner communities, building models are perturbed across large parameter spaces to assess energy and cost performance in the face of programmatic and economic constraints. This paper describes the OpenStudio software framework for performing such analyses. The framework may be accessed through C++, Ruby, and C#, and features a flexible problem formulation method that allows for customizable building energy modeling variables and simulation workflows. Two case studies are presented. The first demonstrates the use of the framework as the back end to a standalone graphical user interface application. It highlights a user interface's ability to make the functionality accessible to a broad audience. The second demonstrates the framework's use in scripted analyses and highlights the flexibility and repeatability offered programmatically to advanced users. Finally, as the framework is available under the open source, GNU lesser general public software license, we discuss how one might extend its functionality to cover more problem types and algorithms.

Introduction

Once a building energy modeler has created and evaluated a building energy model, she may progress to evaluating multiple energy efficiency measures (EEMs) relevant to that model. This is often done by perturbing the baseline building model to determine how energy and cost performance are affected by each EEM. For example, a modeler might increase the efficiency of the HVAC unit, try a different HVAC system, swap out the lighting design for one with a lower lighting power density (LPD), or change the size and location of windows on a façade. Although each change may seem desirable in isolation, their actual appropriateness for a given building will depend on cost, location, geometry, loads, HVAC system, occupant behavior, and economic criteria. Furthermore, the impact of one perturbation may change in the presence of another. Capturing the combined effects of multiple perturbations is a key reason to use a whole building energy simulation engine such as EnergyPlus (DOE 2011). For a single design and small numbers of EEMs, the analysis process can be managed with the help of basic modeling tools, spreadsheets, and text editors. However, as the number of prototype designs, locations, or EEMs increases, more sophisticated tools are needed to determine which model alternatives to simulate, to manage the processes of creating and running the models, and to assemble results from hundreds or thousands of models.

[1] The Alliance for Sustainable Energy, LLC (Alliance), is the manager and operator of the National Renewable Energy Laboratory (NREL). Employees of the Alliance for Sustainable Energy, LLC, under Contract No. DE-AC36-08GO28308 with the U.S. Dept. of Energy, have authored this work. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this work, or allow others to do so, for United States Government purposes.

The literature contains many examples of toolsets that have been developed to support large-scale studies. Academic studies have included demonstrations of optimization, sensitivity analysis, uncertainty quantification, and control algorithms (Caldas & Norford 2003; Barker & Horowitz 2004; Hale & Long 2010, Hopfe 2009; Keeney & Braun 1996; Ren & Wright 1997; Wetter & Wright, 2004). Researchers and analysts also run large numbers of simulations to complete tasks such as validating standards, assessing technical potential, and making prescriptive design recommendations (Bonnema et al. 2010; Griffith et al. 2007; Jarnagin et al. 2006; Thornton et al. 2011). The toolsets to support these studies are typically homegrown software platforms (some of which become more widely available over time) that manage the processes of determining which design alternatives to evaluate, managing calls to the energy simulation program, and aggregating results for analysis.

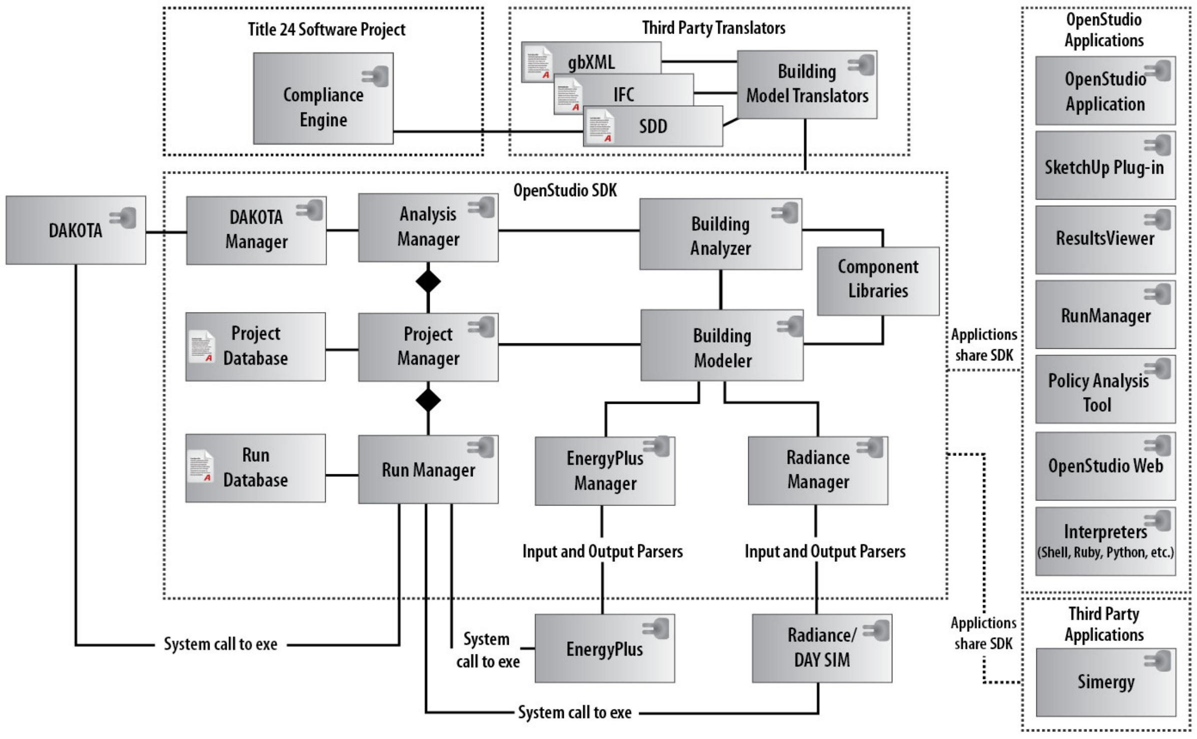
A few tools are available for end users without the resources to develop their own custom analysis toolset. EnergyPlus has native parametric capabilities (DOE 2011). GenOpt provides a library of optimization algorithms that can be coupled to particular values in a text input file such as EnergyPlus input data file (IDF) (Wetter 2011). BEopt is available for optimizing the design of residential buildings against energy and economic metrics simultaneously (NREL 2012). Opt-E-Plus is an NREL internal tool that has features similar to BEopt but is customized for commercial buildings (Ellis et al. 2006). Each of these tools is useful to its target audience, but none has the full ability to easily define new building design parameters; offer a large selection of algorithms and support the addition of new ones; manage large numbers of runs and all the data generated from them, possibly across multiple systems and system types (including clusters); and make it easy to extract and evaluate the final results. Based on prior experience with Opt-E-Plus, NREL's commercial buildings group is developing an analysis framework within OpenStudio to address these shortcomings.

OpenStudio is a software development kit (SDK) and a collection of end-user applications for energy modeling and higher-level functionalities (Ball et al. 2012). Developed largely with public funds, OpenStudio is cross-platform software (Windows, Mac, and Linux) that is available for free under a lesser GNU public license. In this paper, we describe OpenStudio's current analysis framework, which meets the requirements for a full-featured analysis package given above. OpenStudio's functionalities are packaged into reusable modules accessible through C++, Ruby, and C# (support of more languages is technically possible). This allows the functionality to be used by developers, working in their programming language of choice, to complete particular projects. For example, the Policy Analysis Tool (PAT), described later in this paper, is a proof-of-concept application using the analysis framework. Extending PAT's graphical user interface (GUI) to match and exceed the features of Opt-E-Plus is planned as future work. However, one lesson learned from the development and support of Opt-E-Plus is that no GUI can meet the needs of all users. For this reason, we expect many users to prefer accessing the analysis framework through Ruby scripting, as that will always be a more flexible, powerful, and repeatable way to access the functionality. We provide an example in which custom Ruby scripts based on the analysis framework are being used to evaluate EEMs to be included in the U.S. Department of Energy's *Advanced Energy Retrofit Guide for Hospitals.*

Software Architecture

The core functionality of OpenStudio is organized into a number of related modules as depicted in Figure 1. Each module is written in C++ and made available to Ruby and C# using the Simplified Wrapper and Interface Generator (Beazley et al. 2009). The intended clients for this SDK are traditional computer applications (e.g., OpenStudio Application, ResultsViewer, RunManager, and PolicyAnalysisTool), plug-in computer applications (e.g., OpenStudio plug-in for Google SketchUp), Web applications (via Ruby on Rails), and user-written and -maintained Ruby scripts.

Figure 1. OpenStudio Architecture



Credit: Marjorie Schott/NREL

The fundamental building block of the SDK is the Building Modeler, which is an object-oriented representation of a single building energy model. As EnergyPlus is the primary simulation engine targeted by OpenStudio, the underlying data model was initially based on EnergyPlus input data dictionary (IDD). However, this model has been rapidly evolving to improve the modeling experience and support simulation engines for different, but related, domains. For instance, daylight simulation with Radiance is supported by the ability to model individual luminaires and more than two daylight sensors per thermal zone, features that EnergyPlus does not support but are often used in Radiance models. In addition to expanding the data model, the OpenStudio API provides methods for inspecting and perturbing building energy models at higher levels of abstraction than single data fields. These methods include returning all surfaces that reference a given construction, setting window to wall ratio, and adding template HVAC systems. The OpenStudio SketchUp Plug-In and OpenStudio Application provide examples of using such methods, as do some of the Ruby scripts that ship with OpenStudio. Other modules build upon the Building Modeler; we next describe the modules that comprise the OpenStudio analysis framework.

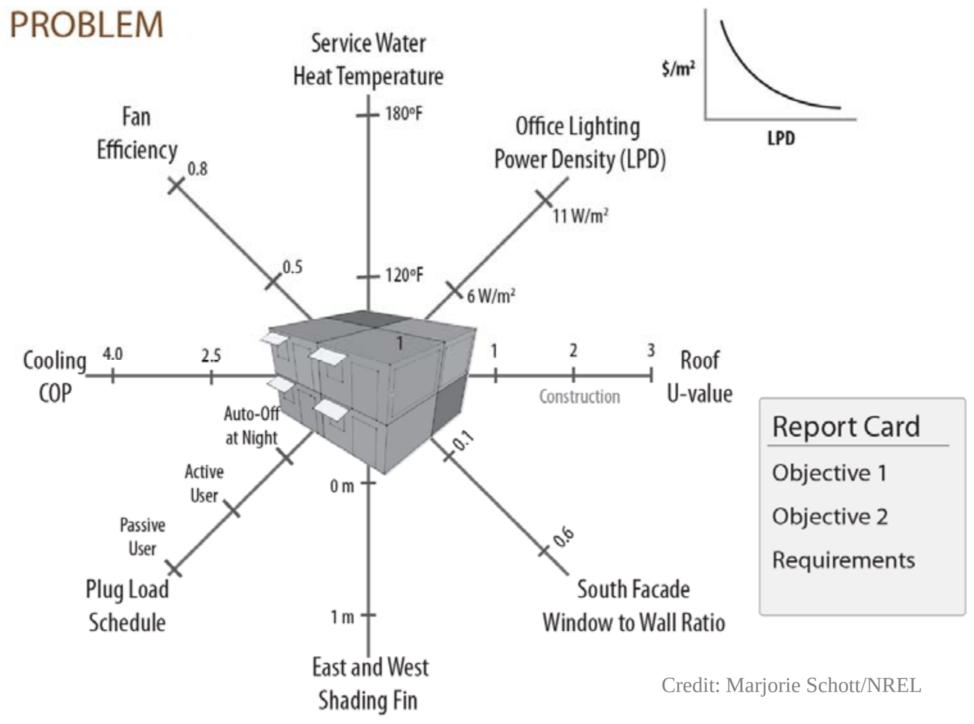
Analysis Framework

OpenStudio users can build custom analysis tools on top of the OpenStudio analysis framework, which is implemented across several of the modules shown in Figure 1. Groups of related models are defined using the Building Analyzer, which contains the notions of a problem definition, an algorithm, and an analysis. Because even the work of constructing a perturbed model from a particular seed file can take a significant amount of time, the Building Analyzer block relies on the Analysis Manager to set up and schedule (via the Run Manager) the actual work of creating and simulating models. As they are created, results are written to the Project Database, using the Project Manager functional block, to enable algorithm restart, and data queries for retrieving high-level results and individual models. The process of conducting a large-scale simulation study can be decomposed into five steps. Each step is described in detail below, in relation to OpenStudio's current capabilities.

Step 1: Define the Seed Model or Seed Models

The first step in using the analysis framework for a project is to define the seed model or seed models. The software supports the use of both OpenStudio model (OSM) and EnergyPlus IDF as the seed model format. New OSMs can be constructed using the SketchUp Plug-In and the OpenStudio Application in concert. OSMs can also be constructed by importing an existing IDF as a starting point. Supporting data needed to create a new model, such as standards compliant constructions and schedules, is available from the Building Component Library (BCL) (Fleming, Long & Swindler 2012).

Figure 2. Example Building Energy Design Problem



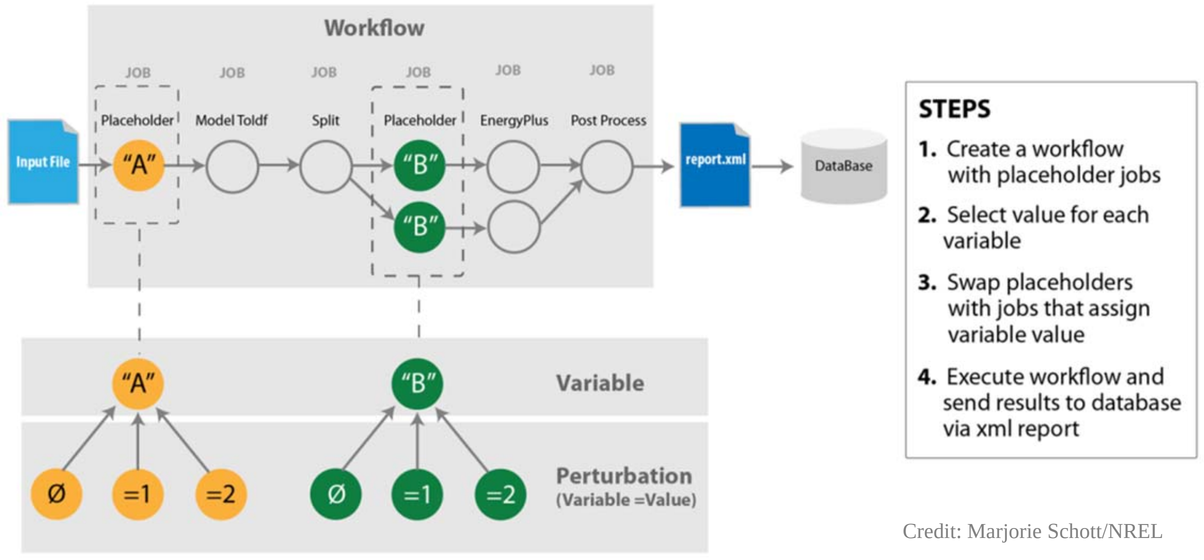
Credit: Marjorie Schott/NREL

Step 2: Define the Analysis Problem

After defining the seed models, the user defines the problem to be analyzed. Variables and the range of values they may take, either continuous or discrete, are defined. Result functions of interest are also specified, especially if the modeler intends to run an optimization algorithm that requires objective functions. Explicit constraints are not yet supported, but would be a desirable addition to the framework. Finally, a simulation workflow, including pre- and post-processing steps, is specified. We typically define a post-process step that extracts high-level results in XML format for consumption by the Analysis Manager and Project Manager. An example problem definition is depicted in Figure 2. In this figure, each axis of the parameter space is a variable that may be set to a particular value in the context of this analysis. This figure also depicts a "report card" showing the objective and constraint functions that will be evaluated for each building.

An example simulation workflow is shown in Figure 3. This figure depicts an example workflow that the Run Manager might use to perturb an input seed model (Placeholder et al.), translate the model into IDF format (ModelToIdf), perform an energy simulation (EnergyPlus), and extract results (Post Process). The workflow consists of a series of jobs to be executed. Some jobs correspond to traditional simulation steps, such as running the EnergyPlus simulation or performing post-processing. However, in OpenStudio this workflow also perturbs the seed model by setting each variable value in sequence, thereby producing a distinct design alternative. Each variable is represented by one placeholder job whose value is set to a distinct number/design alternative for each candidate design at runtime. The values are set through the substitution of a real (non-placeholder) job that performs the actual work of changing the model's input data. The order of the placeholder jobs is fixed, equal to the order of the variables set in the problem definition.

Figure 3. Example Building Energy Design Problem


Workflow
JOB
Placeholder
Model ToIdf
Split
EnergyPlus
Post Process
Input File
"A"
"B"
report.xml
DataBase
Variable
Perturbation
(Variable =Value)
Ø
=1
=2
STEPS
1. Create a workflow with placeholder jobs
2. Select value for each variable
3. Swap placeholders with jobs that assign variable value
4. Execute workflow and send results to database via xml report

Credit: Marjorie Schott/NREL

The current OpenStudio analysis framework includes three types of perturbations. The first type is a null perturbation that does not change the input model at all. The second type is a ruleset perturbation, which may only be applied to an OpenStudio model. A ruleset consists of an ordered list of rules. A rule is an ordered list of filters followed by an ordered list of actions. Each rule searches the entire model, filtering objects based on object type, attribute value, or relationship. Only objects that pass all the filters have the actions, which set attribute values and relationships, applied to them. In the analysis framework, fully formed rulesets can be used to define discrete perturbations on a building model. Single rules that set one attribute of type double (a real variable) can be used to define a continuum of possible perturbations. An example ruleset containing one rule that changes the construction type of all the exterior walls is:

Rule 1	Filter 1	If object type is surface, and
	Filter 2	if surface type is wall, and
	Filter 3	if outside boundary condition is outdoors,
	Action 1	Then set construction to construction to myMassWall

The final type of perturbation available is the Ruby script. This type of perturbation may be applied to either OpenStudio models or EnergyPlus IDFs and has access to the entire OpenStudio SDK. This method of representing EEMs is therefore very flexible. For example, our colleagues and we have written scripts that:

- Apply overhangs to all exterior windows.
- Change the window to wall ratio of a façade, with a user-specified floor-to-sill height.
- Apply a packaged variable air volume system with parallel fan powered terminals, direct exchange cooling, and electric resistance heating to each story of a building model.
- Swap wall, roof, and window constructions directly in IDF.
- Change HVAC equipment efficiencies directly in IDF.
- Import objects stored in an EnergyPlus IDF or an input macro file (IMF) section.

The scripts can be written free form, or they can be written following the OpenStudio measures interface. The measures interface partitions the script into one part that defines arguments and another part that runs on a model or a workspace (an IDF), so that the arguments can be displayed interactively in our modeling applications.

Step 3: Select an Algorithm

In the analysis framework, users can explicitly specify combinations of variable values themselves, or they can use algorithms to automatically generate combinations of variable values. OpenStudio has custom code for performing full factorial design of experiments (meshes) on discrete variables. It also includes a version of the sequential search algorithm for optimizing discrete problems with two objective functions from BEopt and Opt-E-Plus (Ellis et al. 2006; NREL 2012). We are currently working to give users access to many of the algorithms available in DAKOTA, an algorithm library developed by Sandia National Laboratories (Adams et al. 2009).

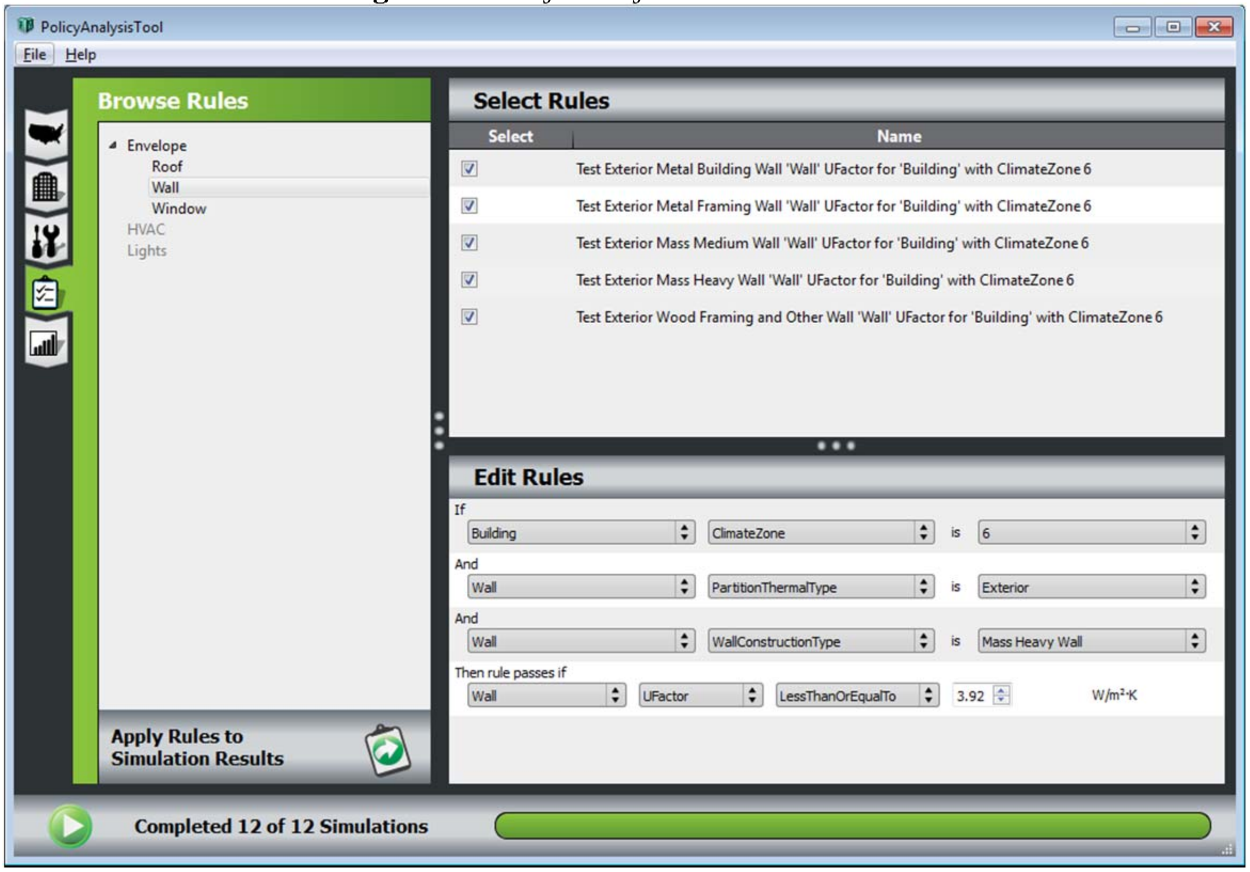
Step 4: Run the Analysis

An analysis is formed by grouping a seed model, a problem, and an algorithm into one entity. Once that is done, user-specified sets of variable values can be used to construct data points (alternative design specifications) and add them to the analysis. The Analysis Manager is then instantiated and used to run the analysis. Any user-specified points are queued first. After that, the Analysis Manager interacts with the algorithm to construct and run additional data points. The Analysis Manager has explicit dependencies on the Project Manager, which it uses to store the generated data in a Project Database, and the Run Manager, which handles the scheduling of individual jobs, possibly on a local machine and a separate cluster resource simultaneously. To the end user, using the Analysis Manager to run an analysis with a DAKOTA algorithm looks very similar to using a native OpenStudio algorithm. On the back end, the Analysis Manager, in concert with the Run Manager and the Building Analyzer, handles all the communications with DAKOTA, which at this time happens through files saved to disk.

Step 5: Evaluate the Analysis

After an analysis has been run, the Project Database can be used to perform queries on the results data. In the course of the run, each data point is populated with its last input file (which represents the fully instantiated design alternative), its EnergyPlus SQLite output file, and its report XML file. The last input file and SQLite output file can be opened using OpenStudio and browsed in detail. More often, the high-level values in the XML report file are of interest. These values are saved directly to the project database for fast access. These values are often programmatically retrieved from the project database and assembled into tables and plots for human inspection.

Figure 3. Policy Analysis Tool Rules Tab



Select Rules

Select	Name
☑	Test Exterior Metal Building Wall 'Wall' UFactor for 'Building' with ClimateZone 6
☑	Test Exterior Metal Framing Wall 'Wall' UFactor for 'Building' with ClimateZone 6
☑	Test Exterior Mass Medium Wall 'Wall' UFactor for 'Building' with ClimateZone 6
☑	Test Exterior Mass Heavy Wall 'Wall' UFactor for 'Building' with ClimateZone 6
☑	Test Exterior Wood Framing and Other Wall 'Wall' UFactor for 'Building' with ClimateZone 6

Edit Rules

If
Building ClimateZone is 6

And
Wall PartitionThermalType is Exterior

And
Wall WallConstructionType is Mass Heavy Wall

Then rule passes if
Wall UFactor LessThanOrEqualTo 3.92 W/m²·K

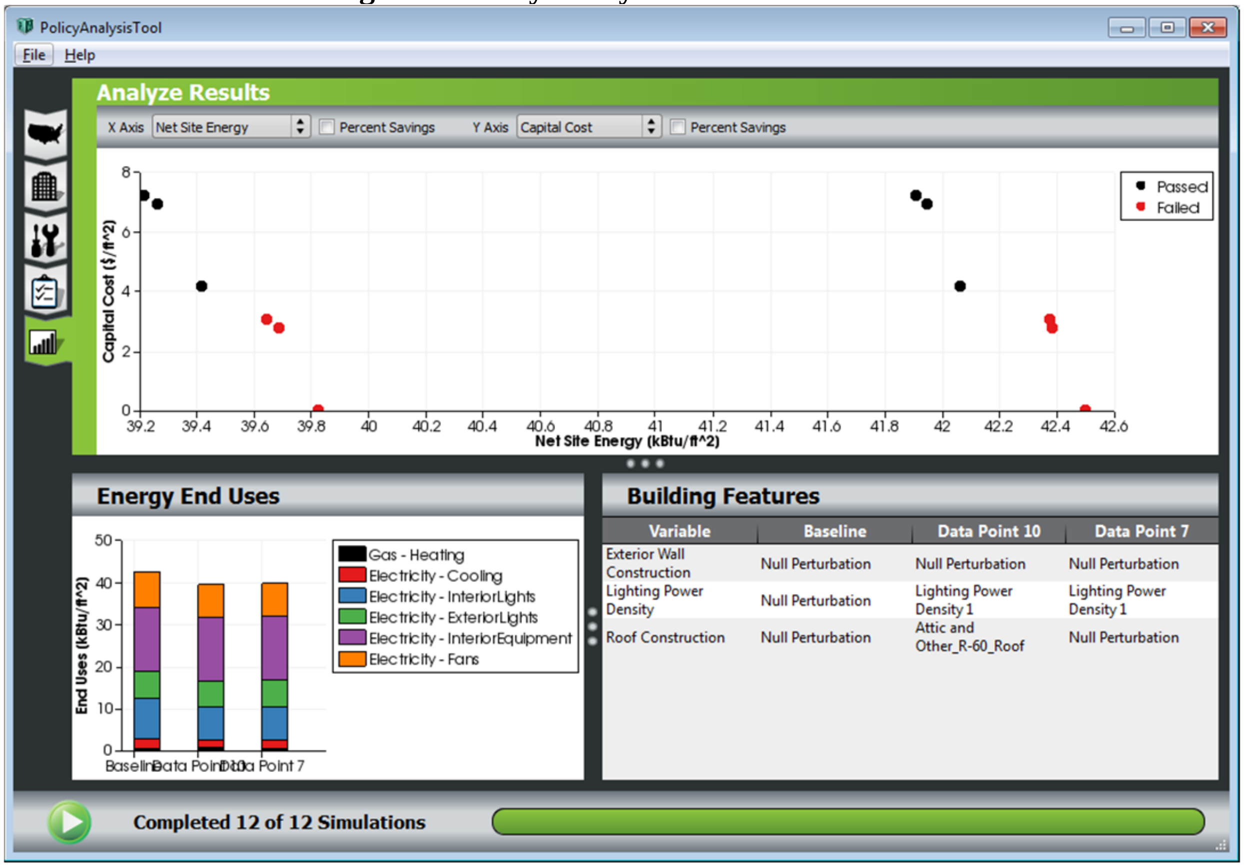
Completed 12 of 12 Simulations

Case Study I: Policy Analysis Tool

The Policy Analysis Tool is a proof-of-concept application that uses the analysis framework for energy standards evaluation and validation. The tabs on the left side guide the user through the steps of performing the analysis. On the first two tabs, the user selects a climate zone and a building type. The third tab lets users choose candidate roof constructions, exterior wall constructions, and LPDs to apply to the baseline model. At runtime, those perturbations are implemented using the rulesets described above; the roof and wall constructions are applied using a rule very much like Rule 1. A full factorial design of experiments algorithm is always applied; because the computational requirements of running a full mesh are prohibitive, the Policy Analysis Tool ensures that users do not swamp their systems by limiting the total number of possible combinations to 16.

The Policy Analysis Tool has an additional standards capability that only applies to California climate zones. For analyses in California climate zones, the user can modify a ruleset (consisting of rules made of filters and a pass/fail action) that partially implements the prescriptive path of CEC Title 24 2008 using the graphical interface shown in Figure 3. Because many rules in the standard will not apply to a given building, the CLIPS expert system library (Riley 2011) is used to determine which rules to check and in which order.

Figure 4. Policy Analysis Tool Results Tab



X Axis: Net Site Energy ☐ Percent Savings Y Axis: Capital Cost ☐ Percent Savings

Variable	Baseline	Data Point 10	Data Point 7
Exterior Wall Construction	Null Perturbation	Null Perturbation	Null Perturbation
Lighting Power Density	Null Perturbation	Lighting Power Density 1	Lighting Power Density 1
Roof Construction	Null Perturbation	Attic and Other_R-60_Roof	Null Perturbation

Completed 12 of 12 Simulations

The results tab of the application is shown in Figure 4. Initially, this tab is largely blank. When the user clicks the run button, an analysis is created on the back end. The seed model is the selected baseline model. The problem has three variables, one for each type of perturbation: roof construction, exterior wall construction, and LPD. The variables are all discrete, and have as many perturbations as the user selected. The simulation workflow applies all the perturbations, translates the model to EnergyPlus IDF format, runs an EnergyPlus simulation, applies a postprocessing step to pull high-level results into the project database, and, if applicable, applies the standards ruleset to determine whether the model passes. As simulations are completed, high-level results for each point are shown using a perturbation summary table (lower right corner of Figure 4), a scatter plot of economic objective versus energy objective (top), and a bar chart showing the breakdown of energy use intensity by end use (lower left). The scatter plot shows all data points simultaneously; the focus of the table and the bar chart changes based on user selection.

Case Study II: Advanced Energy Retrofit Guide for Hospitals

The *Advanced Energy Retrofit Guide for Hospitals* project analyzes 15 EEMs commonly applied to hospitals. Its goal is to provide an optimum package of EEMs in terms of net present value (NPV) and energy cost savings. The problem can thus be stated as a bi-objective design optimization problem seeking to maximize energy cost savings and NPV.

Hospital buildings have complex HVAC systems. At the beginning of this project, the OpenStudio model was not yet full featured enough to support this building type. The analysis was therefore built using EnergyPlus IDFs as the seed models. Each EEM was defined in an individual Ruby script that opens an input IDF file, affects the perturbation, and saves the perturbed model with a different file name.

The Ruby code to load the input IDF is shown below. First, a path to the file is created using the new method of the OpenStudio::Path object. Second, the IDD file type is specified using the to_IddFileType method on the string “EnergyPlus”. Finally, the file is loaded into memory with OpenStudio::Workspace.load. The OpenStudio Workspace is an in-memory instantiation of an IDF model where the name references are automatically maintained by treating the reference as a bidirectional pointer, rather than as a string.

```
myIdfInputpathFile = OpenStudio::Path.new('in.idf')
myIdfFileType = “EnergyPlus”.to_IddFileType
myWorkspace = OpenStudio::Workspace.load(myIdfInputpathFile,myIdfFileType)
if myWorkspace.empty?
    raise "Idf not found or the file is malformed."
else
    myWorkspace = myWorkspace.get
end
```

After the IDF file is loaded into a workspace, the script goes on to perturb the model. In this project, each EEM could be implemented using two types of modifications—adding a new EnergyPlus object, or changing the value of a field in an existing object. An example of adding an EnergyPlus Branch object follows.

```
iddObjectType_Branch = OpenStudio::IddObjectType.new("Branch")
idfObject_Branch = OpenStudio::IdfObject.new(iddObjectType_Branch)
workspaceObject_BranchCondenserLoop = myWorkspace.addObject(idfObject_Branch).get
```

Object addition involves defining an IDD object type, creating the IDF object, and adding the IDF object to the workspace. Modifying input field values involves digging a little deeper to find the desired object and to set the field value:

```
iddObjectType_BranchList = OpenStudio::IddObjectType.new("BranchList")
# retrieve all objects of type BranchList
workspaceObject_BranchList = myWorkspace.getObjectsByType(iddObjectType_BranchList)
workspaceObject_BranchList = myWorkspace.sort(workspaceObject_BranchList)

for i in 0..(workspaceObject_BranchList.size-1)
   # get the branch list name
   objectNameField = workspaceObject_BranchList[i].getString(0).get
   # if the name is not “TowerWaterSys Demand Branches”
   if  objectNameField =~ /^TowerWaterSys Demand Branches/
     index = workspaceObject_BranchList[i].numFields - 1
     # get the current outlet branch
     currentVal_OutletBranch = workspaceObject_BranchList[i].getString(index).get
     # add an economizer branch to the condenser
     workspaceObject_BranchList[i].setString(index,”Economizer condenser Branch”)
     # keep the current outlet branch by placing it in the next position
     workspaceObject_BranchList[i].setString(index+1, currentVal_OutletBranch)
   end
end
```

Finally, the perturbed IDF file is saved to disk so the run manager can pick it up and hand it to the next perturbation script or to EnergyPlus:

```
myIdfOutputFilePath = OpenStudio::Path.new(measureCode + "out.idf").to_s
myWorkspace.save(myIdfOutputFilePath,true)
```

This project also has a custom Ruby postprocessing script for calculating energy cost savings and NPV relative to the baseline (seed) model. At this time, special points (such as the baseline) are not automatically available to postprocesses for calculating such output values. Thus, to enable the direct use of energy cost savings and NPV as objective functions, the baseline model was pre-run, with relevant results assembled into a comma separated value (CSV) format. The postprocess then made direct use of this CSV file when aggregating results for each new data point.

The analysis was set up to run using the Sequential Search optimization algorithm available in OpenStudio. As stated earlier, this algorithm works to solve biobjective optimization problems over discrete variables. Methodologically, it is a heuristic algorithm that greedily traces out the Pareto tradeoff curve in a prespecified direction. In this case, it would start from the baseline point, constantly moving in the direction of increasing energy cost savings, always choosing the point with the highest NPV for the current energy cost savings level.

Extending the Framework

OpenStudio is an open source project. In addition to allowing users to reuse functionality from the SDK, this also allows users to submit code improvements back to the main project. These improvements could include new search algorithms, improvements in speed or performance, Run Manager improvements to run on new types of distributed systems, or many other new features. Another key contribution users can make is to share the perturbations they have used in their analysis, either in ruleset or ruby script form, through the BCL. As the features of the analysis framework grow and the library of available perturbations is built up, new users will find it easier to create and perform their own building energy analyses.

Conclusions and Future Work

The analysis functionality contained in the OpenStudio SDK gives software developers the ability to quickly create tools for conducting custom, large scale building energy simulation studies. The SDK is under active development, but is mature enough to use today. Seed models may be created using the OpenStudio modeling applications. Designs can be parameterized in several ways, using discrete and continuous variables, written as rulesets or free-form scripts. A library of algorithms is provided, and the details of data and run management are taken care of automatically. The functionality can be exposed through graphical user interfaces built on the SDK, or it can be used directly by advanced users writing Ruby scripts. Graphical applications, such as the Policy Analysis Tool, can reach wider audiences, but will never be able to meet the needs of every user. Advanced users can implement their own analysis tools using custom Ruby scripts. Because these scripts use the OpenStudio SDK, less time is spent developing code than in a fully homegrown software solution. Because the scripts can be rerun at any time, the entire analysis highly repeatable.

Future planned work includes adding robust standards and costing functionalities, expanding the types of problems that can be defined in the framework, adding algorithms both natively and from DAKOTA, and enabling users to share reusable pieces of their work (especially building components, rulesets, and measure scripts) through the BCL. This will enable the community to share and refine EEMs for general use. Eventually, we hope that a graphical user interface built on the OpenStudio analysis framework will allow the general public to perform automated building energy optimization in a simple and intuitive way.

Acknowledgements

This work was financially supported by the U.S. Department of Energy Building Technologies Program, and by the California Energy Commission (Task No. BEC7.1335 and WW2D.1000, respectively). The authors are greatly indebted to our sponsors at those organizations, and also to our colleagues at the National Renewable Energy Laboratory. OpenStudio would not be what it is today without Brian Ball, Kyle Benne, Larry Brackney, Luigi Gentile Polese, David Goldwasser, Nick Long, Marjorie Schott, Alex Swindler, and Jason Turner. Bob Hendron leads the *Advanced Energy Retrofit Guide for Hospitals* project.

References

Adams, B. M., W. J. Bohnhoff, K. R. Dalbey, J. P. Eddy, M. S. Eldred, D. M. Gay, K. Haskell, P. D. Hough, L. P. Swiler. 2009. *DAKOTA, A Multilevel Parallel Object-Oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 5.0 User's Manual*. SAND2010-2183, December 2009. Updated December 2010 (Version 5.1). Updated November 2011 (Version 5.2). Sandia National Laboratories, Albuquerque, N.M.

Ball, B., K. Benne, K. Fleming, L. Gentile Polese, D. Goldwasser, K. Gowri, B. Griffith, R. Guglielmetti, E. Hale, R. Hitchcock, N. Long, D. Macumber, M. Schott, A. Swindler, J. Turner, E. Weaver. 2012. "OpenStudio." Version 0.7. http://openstudio.nrel.gov.

Beazley, D., L. Ballabio, W. Fulton, M. Gossage, M. Koppe, J. Lenz, M. Matus, J. Stewart, A. Yerkes, S. Yoshiki, S. Singhi. X. Delacour & O. Betts. 2009. "Simplified Wrapper and Interface Generator (SWIG)." http://www.swig.org.

Bonnema, E., D. Studer, A. Parker, S. Pless & P. Torcellini. 2010. *Large Hospital 50% Energy Savings: Technical Support Document*. NREL/TP-550-47867. National Renewable Energy Laboratory, Golden, Colo.

Caldas, L. G. & L. K. Norford. 2003. "Genetic Algorithms for Optimization of Building Envelopes and the Design and Control of HVAC Systems." *Journal of Solar Energy Engineering* 125(3): 243-351.

Christensen, C., G. Barker & S. Horowitz. 2004. "A Sequential Search Technique for Identifying Optimal Building Designs on the Path to Zero Net Energy." In *Proceedings of the Solar 2004 Conference Including Proceedings of the 33rd ASES Annual Conference* and *Proceedings of the 29th National Passive Solar Conference*. Portland, Ore.: 877-882.

[DOE] U.S. Department of Energy. 2011. "EnergyPlus Energy Simulation Software." Version 7.0. http://www.eere.energy.gov/builldings/energyplus. Washington, D.C.

Ellis, P. G., B. Griffith, N. Long, P. Torcellini & D. Crawley. 2006. "Automated Multivariate Optimization Tool for Energy Analysis." NREL/CP-550-40353. National Renewable Energy Laboratory, Golden, Colo.

Fleming, K., N. Long & A. Swindler. 2012. "The Building Component Library: an Online Repository to Facilitate Building Energy Model Creation." In *Proceedings of the 2012 ACEEE Summer Study on Energy Efficient Buildings*. Pacific Grove, Calif.

Griffith, B., N. Long, P. Torcellini, R. Judkoff, D. Crawley & J. Ryan. 2007. *Assessment of the Technical Potential for Achieving Net Zero-Energy Buildings in the Commercial Sector*. NREL/TP-550-41957. National Renewable Energy Laboratory, Golden, Colo.

Hale, E. T. & N. L. Long. 2010. “Enumerating a Diverse Set of Building Designs using Discrete Optimization.” In *Proceedings of SimBuild 2010*, 4th National Conference of IBPSA-USA, New York, NY: 77-84.

Hopfe, C. J. 2009. “Uncertainty and sensitivity analysis in building performance simulation for decision support and design optimization.” Ph.D. Thesis. Technische Universiteit Eindhoven, Eindhoven, The Netherlands.

Jarnagin, R. E., B. Liu, D. W. Winiarski, M. F. McBride, L. Suharli & D. Walden. 2006. *Technical Support Document: Development of the Advanced Energy Design Guide for Small Office Buildings*. PNNL-16250. Pacific Northwest National Laboratory, Richland, Wash.

Keeney, K. & J. Braun. 1996. “A Simplified Method for Determining Optimal Cooling Control Strategies for Thermal Storage in Building Mass.” *HVAC&R Research* 2(1): 402-414.

[NREL] National Renewable Energy Laboratory. 2012. “BEopt.” Version 1.2. http://beopt.nrel.gov. Golden, Colo.

Ren, M. J. & J. A. Wright. 1997. “Predictive Optimal Control of Fabric Thermal Storage Systems.” In *Proceedings of the 5th International IBPSA Conference*, Prague, Czech Republic: 71-78.

Riley, G. “CLIPS: A Tool for Building Expert Systems.” Version 6.3. http://clipsrules.sourceforge.net.

Thornton, B. A., M. I. Rosenberg, E. E. Richman, W. Wang, Y. Xie, J. Zhang, H. Cho, V. V. Mendon, R. A. Athalye & B. Liu. 2011. *Achieving the 30% Goal: Energy and Cost Savings Analysis of ASHARE Standard 90.1-2010*. PNNL-20405. Pacific Northwest National Laboratory, Richland, Wash.

Wetter, M. & J. Wright. 2004. “A Comparison of Deterministic and Probabilistic Optimization Algorithms for Nonsmooth Simulation Based Optimization.” *Building and Environment* 39(8): 989-999.

Wetter, M. 2011. *GenOpt® Generic Optimization Program User Manual Version 3.1.0*. Lawrence Berkeley National Laboratory, Berkeley, Calif.