Energy and Buildings 117 (2016) 321–331

Contents lists available at ScienceDirect

Energy and Buildings

journal homepage: www.elsevier.com/locate/enbuild

There's a measure for that!

Amir Roth[a,*], David Goldwasser[b], Andrew Parker[b]

[a] Department of Energy, Washington, DC 20585, USA
[b] National Renewable Energy Laboratory, Golden, CO 80406, USA

ARTICLE INFO

Article history:
Received 15 July 2015
Received in revised form
21 September 2015
Accepted 22 September 2015
Available online 26 September 2015

Keywords:
Whole-building energy simulation
Energy conservation measures
Parametric analysis
Uncertainty analysis
Calibration
Reporting
Quality-assurance
Visualization
Automation
Workflow

ABSTRACT

The OpenStudio software development kit has played a significant role in the adoption of the EnergyPlus whole building energy modeling engine and in the development and launch of new applications that use EnergyPlus for a variety of purposes, from design to auditing to code compliance and management of large portfolios. One of the most powerful features of the OpenStudio platform is Measure, a scripting facility similar to Excel's Visual Basic macros. Measures can be used to apply energy conservation measures to models—hence the name—to create reports and visualizations, and even to sew together custom workflows. Measures automate tedious tasks increasing modeler productivity and reducing error. Measures have also become a currency in the OpenStudio tools ecosystem, a way to codify knowledge and protocol and transfer it from one modeler to another, either within an organization or within the global modeling community. This paper describes some of the many applications of Measures.

1. Introduction

In current practice, common building energy modeling (BEM) tasks are codified and transferred via either documents—which every new modeler must interpret and implement in their chosen workflow—or via ad hoc observation and apprenticeship. These knowledge transfer channels have low bandwidth, are prone to human judgment and error and generally contribute to the wide variation in modeling practice and results observed across the industry. The US Department of Energy's (DOE) OpenStudio platform has a scripting facility called "Measures" that is transforming the way in which modeling tasks are performed and in which modeling knowledge is codified and shared [8].

OpenStudio is a strategic component in the DOE's efforts to increase the effective use of advanced BEM, specifically with EnergyPlus [3], in building design and operation. The OpenStudio software development kit (SDK) is a library that maintains a detailed representation of the building energy model and the results of the simulation and provides functions—importing 3D geometry, creating standard HVAC systems, exporting models to EnergyPlus for energy analysis and to Radiance for lighting analysis, running simulations—that are common to many BEM applications. The SDK dramatically reduces the effort of BEM application development as the OpenStudio graphical application—developed in about four months—demonstrates [6]. Over the past several years, public and private organizations have leveraged OpenStudio to deliver new applications and services, including DOE's Commercial Asset Score rating tool [12,13], Xcel Energy's EDAPT energy design-assistance project management portal [2], and the California Energy Commission's CBECC-Com code-compliance application [1].

OpenStudio Measures are enabled by a combination of three features. First, the SDK provides direct access to OpenStudio's internal model of the building and the results of the simulation. Second, SDK functions can be invoked from interpreted scripting languages like Ruby and C#. Finally, OpenStudio can interpret and apply scripts at runtime. Together these create a scripting facility that is similar to Microsoft Excel's Visual Basic macros. Just as Visual Basic has made Microsoft Excel a powerful and ubiquitous tool, Measures are having a similar impact on OpenStudio and its users. A modeler can encapsulate a specific procedure in a Measure and share that Measure with other modelers who can then apply it to their own models. Measures allow modeling procedures to be codified and shared in a consistent way that eliminates much of the human error

* Corresponding author.
E-mail addresses: amir.roth@ee.doe.gov (A. Roth), david.goldwasser@nrel.gov (D. Goldwasser), andrew.parker@nrel.gov (A. Parker).

http://dx.doi.org/10.1016/j.enbuild.2015.09.056
0378-7788/

inherent in document interpretation and data transcription. By improving modeling consistency and modeler productivity, Measures have become the most powerful feature of the OpenStudio platform and the core of its value proposition.

The original use of Measures—and the origin of the name "Measures"—is the transformation of an existing model representing the application of energy conservation measures (ECMs). However, Measures are as flexible as the underlying model representation and the scripting language and have found new uses and applications. The public repository Building Components Library (http://bcl.nrel.gov/) already contains over 190 Measures [4], and more are being added every week, by both the OpenStudio team and by users. The rest of the paper describes the various types of Measures and the applications they support.

2. Model measures

Model Measures take an OpenStudio model as input and produce a transformed OpenStudio model as output. This structure aligns with the concept of ECMs, and many model Measures are used as ECMs. Model Measures can be simple "search-and-replace" routines, e.g., "replace exterior window construction A with construction B". However, with access to the full OpenStudio object model, they can also be arbitrarily complex. Full model access allows Measures to apply transformations selectively and surgically, e.g., "remove exterior window on facades oriented between degrees X and Y". More generally, it supports Measures that perform different transformations in different contexts as well as multiple coordinated transformations.

Example: Simplistic daylighting package

Fig. 1 shows a Measure that applies a simplistic Daylighting ECM. To exploit Daylighting in the northern hemisphere, fenestration is usually removed or minimized on East and West facing facades and shaded on the South facade to minimize solar heat gain and glare. The top shows the Measure code, the bottom shows a simple box model before and after Measure application.

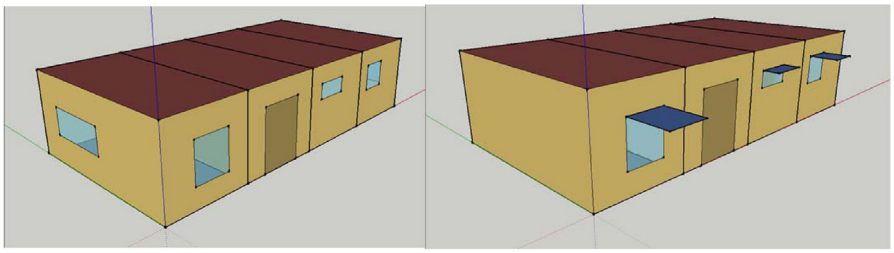
The Measure is written in the programming language Ruby. The `run` function is invoked when the measure executes. Statements beginning with # are "comments" and are not part of the code, rather they are inline documentation. The Measure iterates through all sub-surfaces in the model—in OpenStudio walls are surfaces and wall openings including windows are sub-surfaces. It tests each sub-surface to ensure that it is an exterior window, then retrieves the azimuth and performs a specific action based on the azimuth value. North facing windows—azimuth between 315 and 45—are untouched. East and West facing windows are removed. South facing windows are fitted with an overhang. The high-level method `addOverhangByProjectionFactor` is provided by the OpenStudio SDK.

Those who are familiar with EnergyPlus and its input format (IDF) will notice how compact this code is relative to the EnergyPlus objects it manipulates. EnergyPlus fenestration sub-surface objects are 22 fields long. Removing a sub-surface object is not a simple one-line modification to an IDF file. Neither is creating an overhang surface given a window sub-surface object. This economy is provided by the OpenStudio SDK and its abstractions.

Example: AEDG K-12 school daylighting package

The example Measure above is a highly simplified version of one of the most illustrative and visually impressive model Measures—the ASHRAE Advanced Energy Design Guide (AEDG) "K-12 School Daylighting Package" Measure. This Measure coordinates six different transformations. Like the simplistic Measure it: (i) reduces or eliminates glazing on east and west facing facades, and (ii) adds shading on southern facades to reduce glare and heat gain from direct solar beams—although it does this in a more robust and surgical way. It also: (iii) reconfigures glazing on north and south facing facades to maximize daylight harvesting while minimizing glazing area, (iv) properly uses clear and translucent glass,

```ruby
# define what happens when the measure is run
def run(model, runner, arguments)
   super(model, runner, arguments)
   # traverse all sub-surfaces in the model
   model.getSubSurfaces.each do |s|
      # test if it is an exterior window
      next if not s.outsideBoundaryCondition == "Outdoors"
      next if not s.subSurfaceType == "FixedWindow"
      # get the azimuth for the surface
      azimuth =  OpenStudio::convert(s.azimuth,"rad","deg").get
      # east or west
      if (azimuth >= 45 and azimuth < 135) or (azimuth > 225 and azimuth < 315)
         s.remove
      # south
      elsif (azimuth >= 135 and azimuth < 225)
         s.addOverhangByProjectionFactor(1,0) # (projection factor, offset)
      end # if azimuth
   end # model.getSubSurfaces.each do
   return true
end # def run
```



Before After

Fig. 1. Simplistic daylighting measure. Top: measure code. Bottom: before-and-after illustration of measure.