Outline

• Problem Statement and Solution
• Measures in OpenStudio
  o What is an OpenStudio Measure?
  o Structured Rule sets
  o Freeform Scripts
• Use Cases for OpenStudio Measures
  o Interactive Measure Application
  o Noninteractive Measure Application
  o Composing Parameter Spaces With Measures
• OpenStudio Measures and the BCL
• Conclusions
Problem Statement and Solution
Problem Statement and Solutions

- Building energy analysis is often time-intensive, error-prone, and nonreproducible
  - GUIs help
  - Libraries of data help
  - Still need to automate tedious workflows
- Entire energy analyses can be scripted end to end using the OpenStudio Ruby API
- Common tasks within an analysis can be automated using OpenStudio Measures
OpenStudio Approach

OpenStudio and SketchUp Plug-in
Model creation and inspection

Use measures interactively from GUIs or noninteractively as part of model’s simulation workflow

OpenStudio Measures
Workflow automation and model transformations

Drag and drop components from BCL into current model

Building Component Library (BCL)
Store building energy modeling data for reuse

Share measures in BCL; measures may refer to BCL data
Measures in OpenStudio
Measures in OpenStudio

• **What is a “Measure” in OpenStudio?**
  - Repeatable transformation of an input building energy model into an output energy model

• **Examples of OpenStudio Measures**
  - Set window to wall ratio to 40% on all façades
  - Replace all T-12 light fixtures with T-5 fixtures
  - Improve fan efficiency from 50% to 70%
  - Import IDF text snippets and set object names
Measures in OpenStudio

• Users may drag and drop multiple Measures at two points in the OpenStudio simulation workflow

Workflow automatically applies Measures, runs simulation, and performs postprocessing
Measures in OpenStudio

• Measures may be written generically to be reused across many building models or written for a specific building model

• Measures may take input parameters
  o WindowToWallRatio Measure takes arguments ‘wwr’, ‘offset’, and ‘application_type’

• Types of OpenStudio Measures
  o Structured Rulesets
  o Freeform Scripts
Structured Ruleset

- Attributes allow OpenStudio ModelObject methods to be called through a string key at runtime
  - `Surface::windowToWallRatio() == getAttribute("windowToWallRatio")`
- Some attributes are read only, others allow read/write
- A Ruleset is composed of Rules
- A Rule is composed of Filters and Actions (there are many types of filters and actions)
- Attribute filters pass or reject objects based on attribute values
- Attribute actions change objects by calling `setAttribute`
- Structured Ruleset can be serialized to XML format

### Example Rule to set WWR on exterior walls:

<table>
<thead>
<tr>
<th>Filter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filter 1</td>
<td>If type is “OS:Surface”</td>
</tr>
<tr>
<td>Filter 2</td>
<td>If <code>getAttribute(“surfaceType”)</code> is “Wall”</td>
</tr>
<tr>
<td>Filter 3</td>
<td>If <code>getAttribute(“outsideBoundaryCondition”)</code> is “Outdoors”</td>
</tr>
<tr>
<td>Action 1</td>
<td>Then <code>setAttribute(“windowToWallRatio”, 0.4)</code></td>
</tr>
</tbody>
</table>

### Initial Structured Ruleset GUI:

- Credit: Elaine Hale/ NREL
Freeform Script

- User defines Ruby class
- Class must have methods ‘name’, ‘arguments’, and ‘run’
- Run method can use any method exposed in the OpenStudio Ruby API
- Freeform Script serialized as Ruby script (text file)

Example code to set WWR on exterior walls:

```ruby
def run(model, runner, arguments)
  wwr = arguments['wwr']
  offset = arguments['offset']
  application_type =
    arguments['application_type']

  heightOffsetFromFloor = nil
  if (application_type.valueAsString ==
      "Above Floor")
    heightOffsetFromFloor = true
  else
    heightOffsetFromFloor = false
  end

  model.getSurfaces.each do |s|
    next if not runner.inSelection(s)
    next if not (s.outsideBoundaryCondition ==
                "Outdoors")
    new_window = s.setWindowToWallRatio(
               wwr.valueAsDouble,
               offset.valueAsDouble,
               heightOffsetFromFloor)
  end
end
```
Use Cases for OpenStudio Measures
Interactive Measure Application

- User can run script at will
- Input arguments are collected at runtime
- User’s selection is passed to the script
- Example scripts available in SketchUp plug-in
  - Includes WindowToWallRatio Measure

Before WindowToWallRatio Measure

After WindowToWallRatio Measure

Credit: David Goldwasser / NREL
Noninteractive Measure Application

- User must save all input parameters ahead of time
- Script is saved with the model; can be thought of as part of the model
- Script is run every time the model is simulated
- Example scripts available in OpenStudio Application
  - Includes WindowToWallRatio Measure

Credit: David Goldwasser / NREL
Composing Parameter Spaces with Measures

- Parameter space is composed of several variables
- Measures are used to set a variable to a given value
- Measures are applied as part of the simulation workflow for each model
- Once parameter space is defined, scripted analyses can be run
  - Design of Experiments
  - Optimization
  - Uncertainty Quantification
- Used to generate data for 179D DOE Calculator web tool
  - 250,000 E+ simulations in two weeks
  - Entire analysis rerun in one week after inputs models and measures were revised

Credit: Marjorie Schott/ NREL
The Building Component Library (BCL)

Create an account to save your components and get access to BCL API.

Search for and download BLC components from website, http://bcl.nrel.gov

Credit: David Goldwasser/NREL

http://bcl.nrel.gov
OpenStudio and the BCL

Drag and drop downloaded BLC components into your model

Search for and download BLC components within OpenStudio Application

Credit: Evan Weaver/NREL
OpenStudio Measures and the BCL

• BCL allows Measures as well as Components
  o Measures may refer to Components
  o Components downloaded as needed when Measure runs
  o Measures in BCL still in early stages

• Future integration with applications planned
  o Drag and drop Measures into user script libraries, simulation workflow, and parameter spaces

Credit: Daniel Macumber/NREL
Conclusions
Conclusions

• GUIs and component libraries reduce time, decrease errors, and improve repeatability in energy modeling
• Entire custom analyses can be scripted end to end using the OpenStudio Ruby API
• Common tasks within an analysis can be automated using OpenStudio Measures
• Initial integration of Measures with OpenStudio, SketchUp Plug-in, and BCL is complete
• Future work will improve integration with GUIs and generate libraries of shareable Measures
Acknowledgments

Funding Organizations:
  o U.S. Department of Energy Building Technologies Program
  o California Energy Commission

Coauthors:
  o Elaine Hale, Kyle Benne, and David Goldwasser

OpenStudio Team Members:
  o Brian Ball, Larry Brackney, Luigi Gentile Polese, Nick Long, Marjorie Schott, Alex Swindler, Jason Turner, and Evan Weaver

179D Case Study:
  o Brent Griffith, Matt Leach, Eric Bonnema, Katherine Fleming, and Michael Deru
Backup Slides
Future Work

• Parameters for Structured Rulesets
• Run Structured Rulesets from OpenStudio and SketchUp Plug-in
• Each measure should generate a report on what it did
• Check if measure applies to given model (alternatively report says it did nothing)
• New measures are added to the BCL over time
• User generated components and measures in BCL