A Proposal for Reusable Radiance Workflows

Mostapha Sadeghipour Roudsari
Ladybug Tools

mostapha@ladybug.tools

18th International Radiance Workshop
22 August 2019, New York

Why?

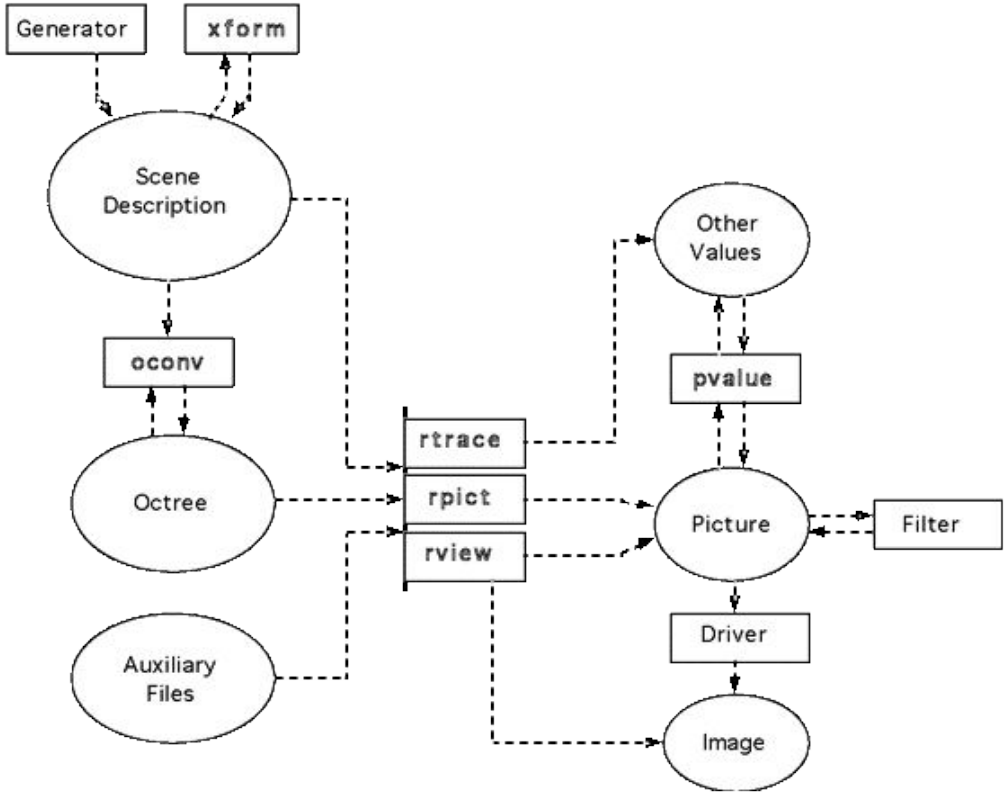
You should know if:

- You ever had to re-use a Radiance model generated by someone else.
- You ever had to re-use a Radiance model for a matrix-based daylight simulation
- You ever had to go through Radiance tutorials. Especially the ones by Andy ;)
- You were here yesterday for David's tutorial

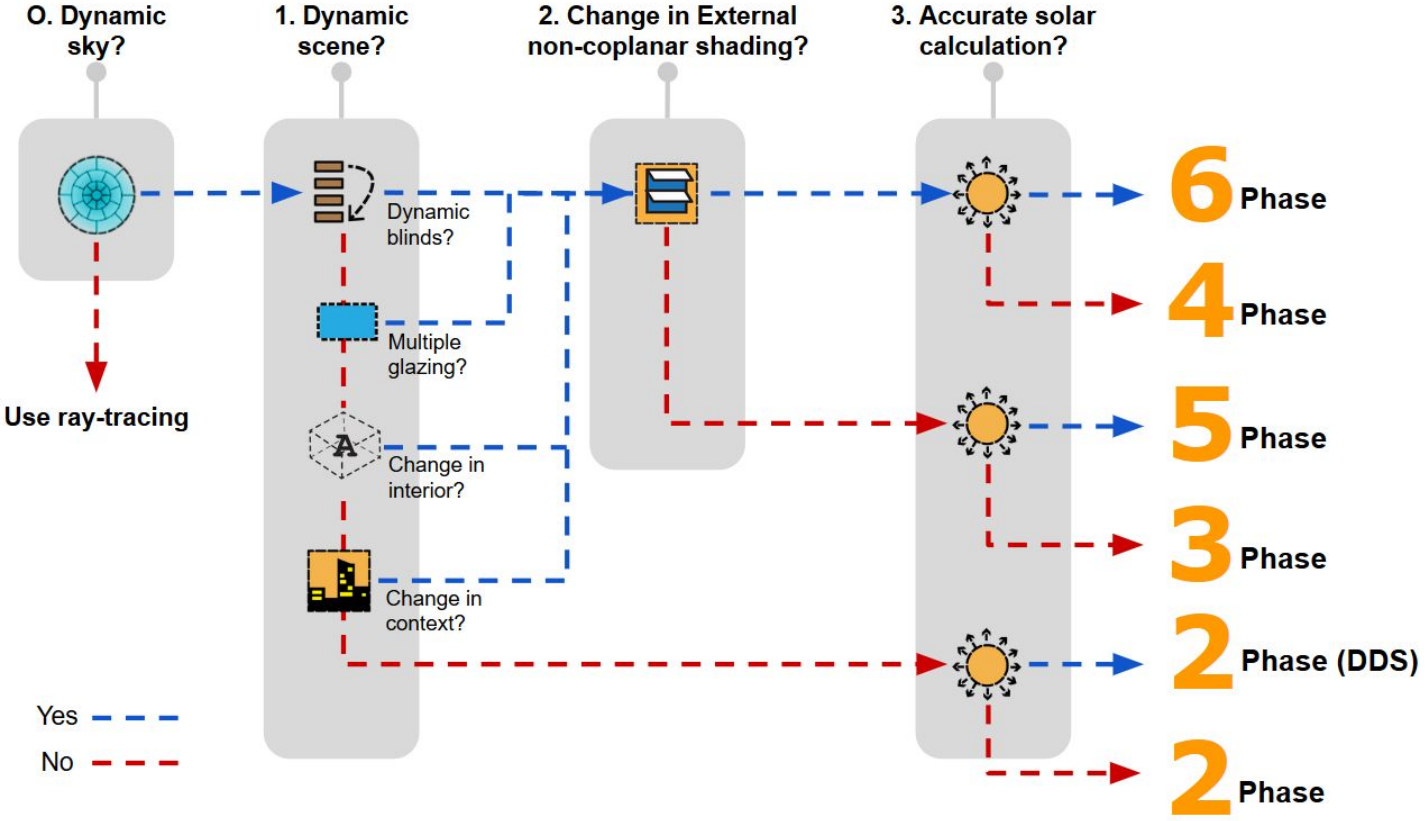
Why?

- There is no standard practice.
- It makes it hard to share and reuse the models. As s result, you cannot execute a Radiance folder generated by one interface with another interface.
- It makes it challenging to validate a model for a certain type of study.
- It’s almost impossible to validate the model without opening every single file manually.
- ...

Good old days

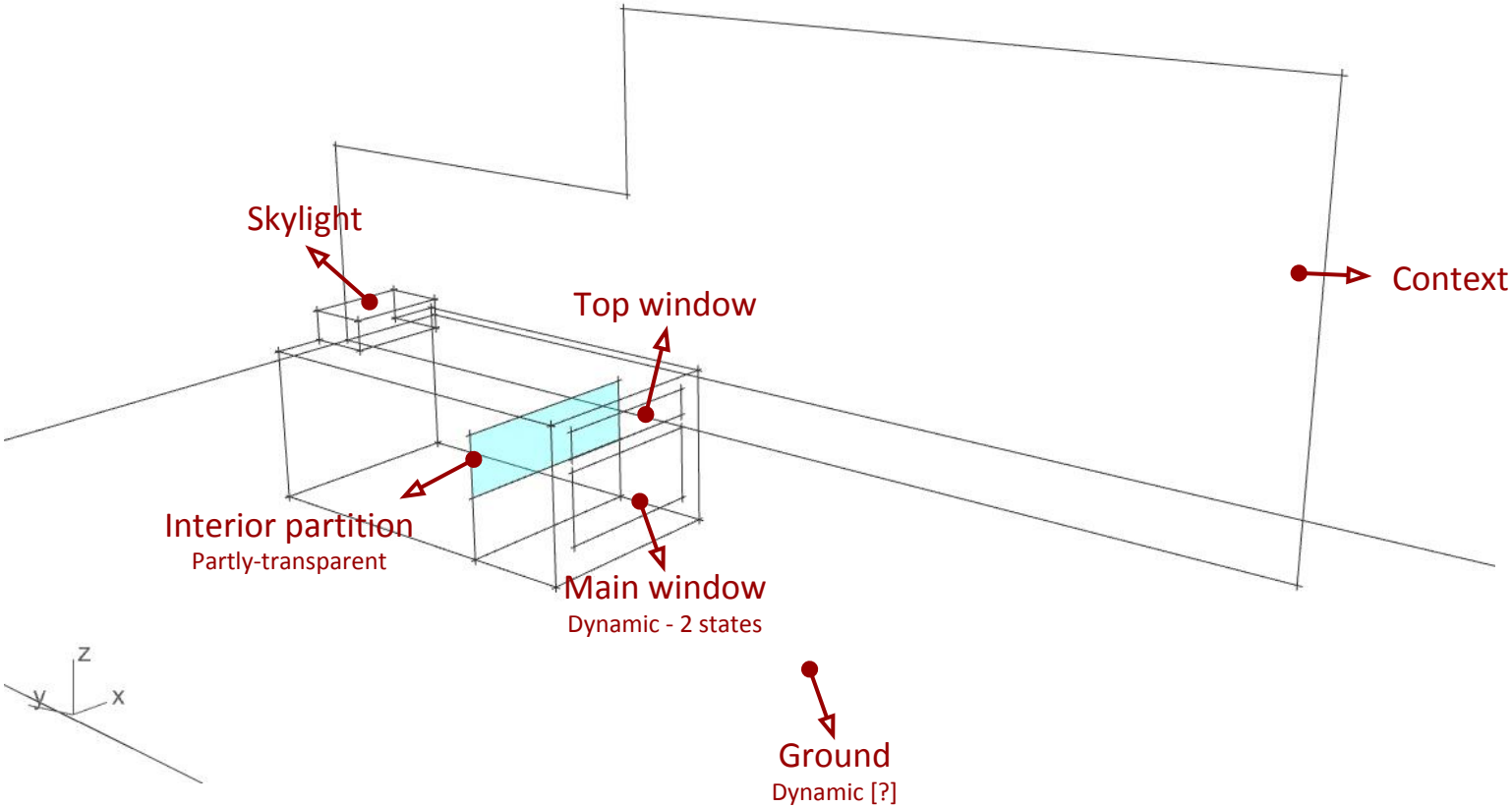


From: The RADIANCE 5.1 Synthetic Imaging System (https://floyd.lbl.gov/radiance/refer/ray.html)



From: Daylighting Simulations with Radiance using Matrix-based Methods (https://www.radiance-online.org/learning/tutorials)

A Simple Radiance Model



Why?

Create a multiphase project from input radiance files #155

Edit | New issue

Open mostaphaRoudsari opened this issue on Jul 15, 2017 · 0 comments

mostaphaRoudsari commented on Jul 15, 2017 — Member

This is not going to be part of the initial release but since I can't stop thinking about this! This will be the point that honeybee meets the advanced radiance users **who won't use honeybee[+] (or anything else in that sense) since they can do all they need with bash**.

We need a utility to take input radiance files and create a ready to go multi-phase folder. The only part that is currently missing is adding radiance input file parser, which we already have in the legacy version. Here is how I think it will work.

```
class FivePhaseGridBased(ThreePhaseGridBased):
    """...."""
    @classmethod
    def fromRadianceFiles(cls, radianceFiles, windowGroupNames, states, *args, **kwargs):
        """Creates the recipe from radiance input files.
```

Assignees
No one—assign yourself

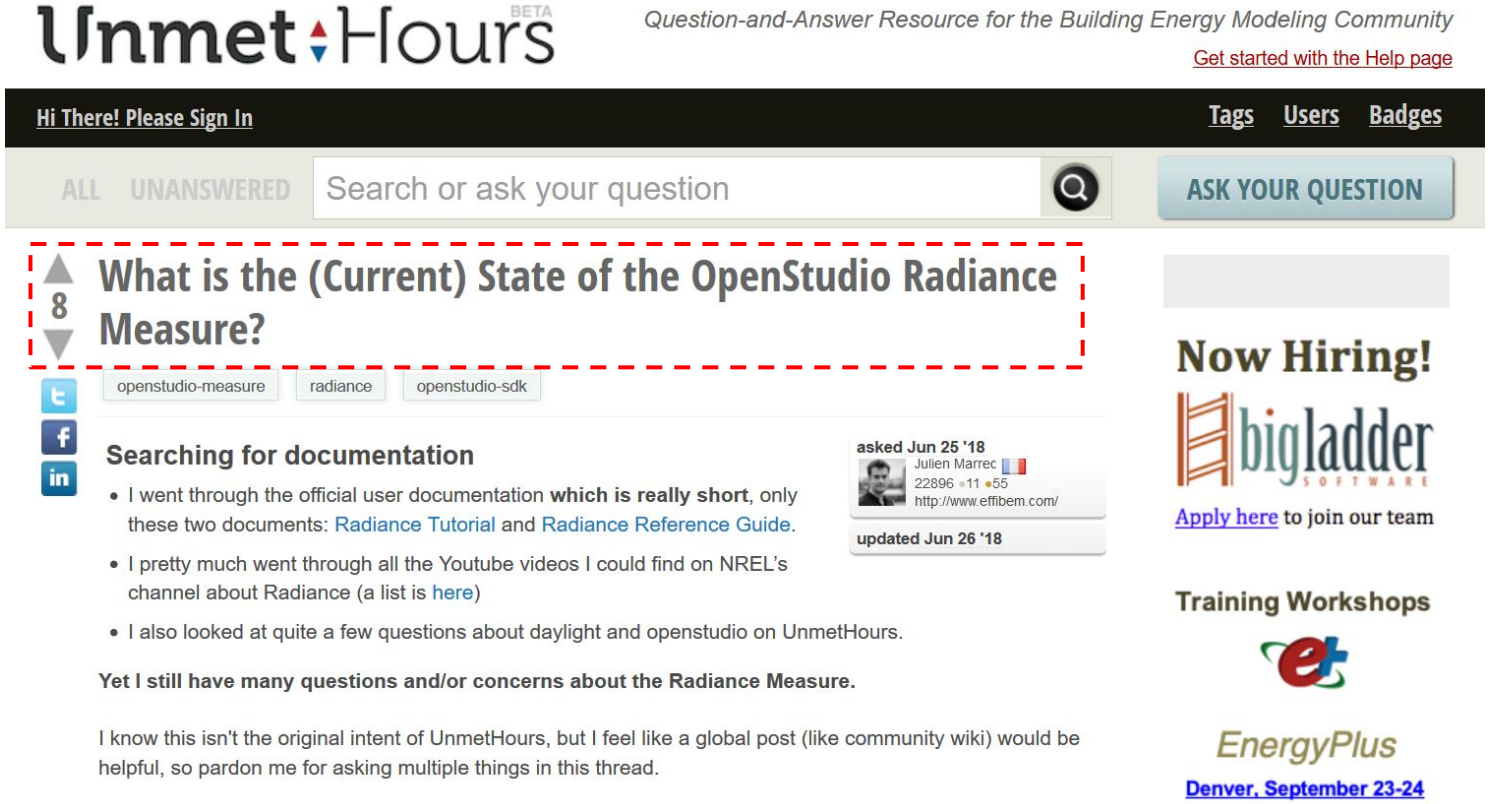
Labels
wish

Projects
None yet

Milestone
refacror recipes

From: https://github.com/ladybug-tools/honeybee/issues/155

Why?



From: https://unmethours.com/question/32359/what-is-the-current-state-of-the-openstudio-radiance-measure/

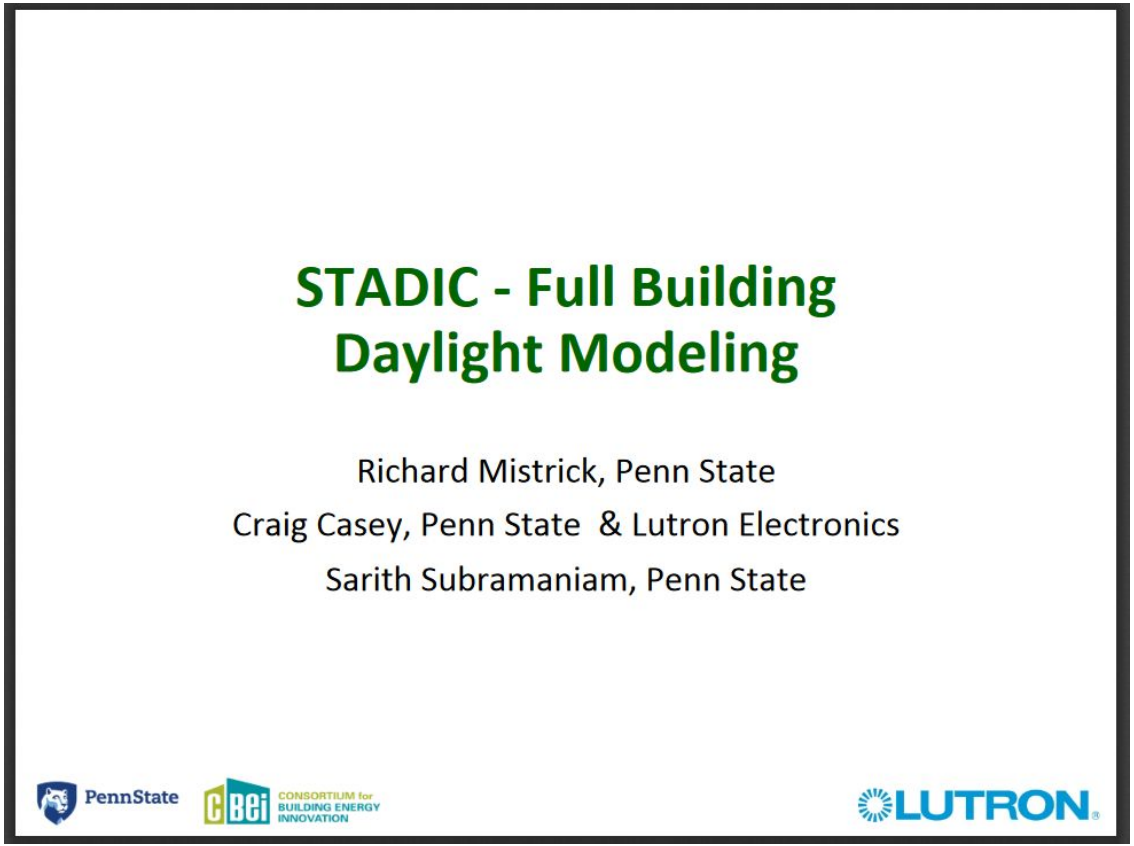
Proposal

What if we start using a standard folder structure to describe the Radiance model?

+

Separate input data from workflow description and from workflow execution

This is a proposal!



From: https://www.radiance-online.org/community/workshops/2015-philadelphia/presentations/day1/Mistrick-STADIC.pdf

Background

- Annual simulations will soon be the norm.
- Codes and rating systems (such as LEED) are beginning to require or encourage annual performance modeling of energy as well as the evaluation of recently developed annual daylight metrics (DA, sDA, ASE, etc.).
- Many spaces include complex geometry and, in most cases, require simulation and operation of shades or blinds (i.e., complex fenestration systems, CFS).
- BSDF's simplify modeling of complex fenestration.
- Summary - Simulations are getting complex and new tools are necessary.

PennState CBEI CONSORTIUM for BUILDING ENERGY INNOVATION LUTRON

From: https://www.radiance-online.org/community/workshops/2015-philadelphia/presentations/day1/Mistrick-STADIC.pdf

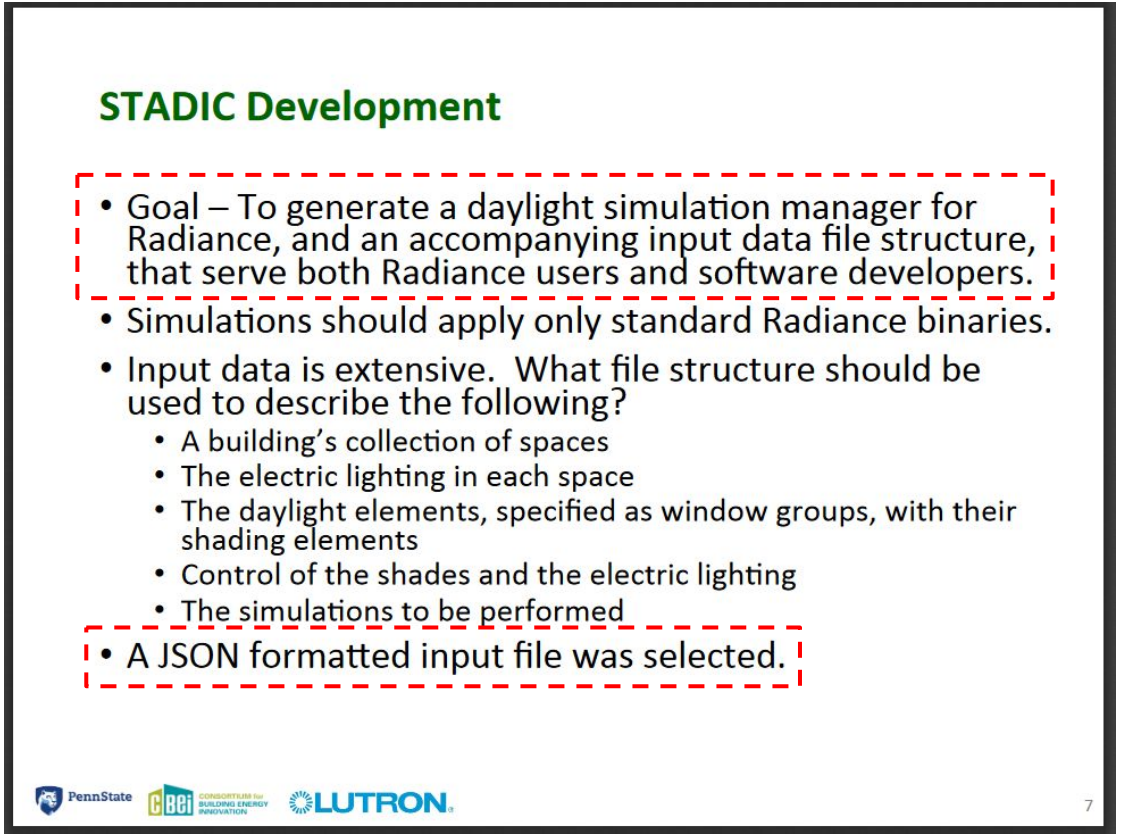
Recent Radiance Developments

- rfluxmtx
- gendaymtx
- rcontrib
- 3-phase and 5-phase simulations with BSDF's
- . . . and others

Putting these all together for a real project can become complicated and time consuming.

PennState CEBEI CONSORTIUM for BUILDING ENERGY INNOVATION LUTRON

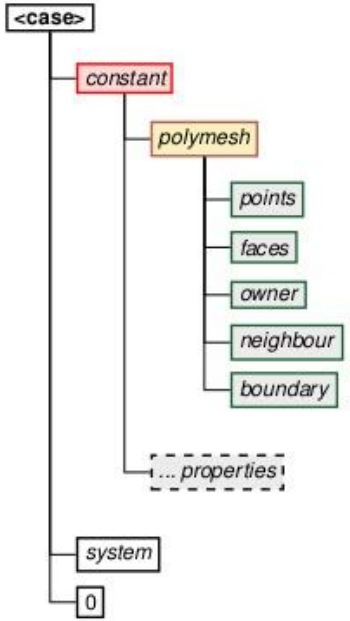
From: https://www.radiance-online.org/community/workshops/2015-philadelphia/presentations/day1/Mistrick-STADIC.pdf



From: https://www.radiance-online.org/community/workshops/2015-philadelphia/presentations/day1/Mistrick-STADIC.pdf

OpenFOAM folder structure

Directory structure

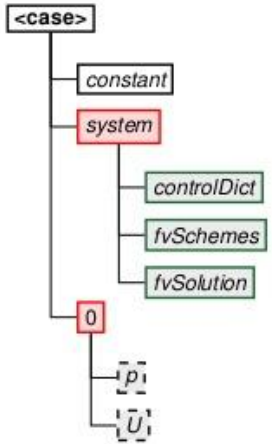


1. *constant*:
   This directory contains the information which remains constant throughout the simulation. It contains the following:
   - 1.1 *polymesh*:
     Contains all the mesh information including:
     (a) *points* → nodal positions
     (b) *faces* → face connectivities
     (c) *owner* → owner cell labels
     (d) *neighbour* → neighbour cell labels
     (e) *boundary* → boundary information
   - 1.2 *...properties*:
     Files which specify physical properties for a particular application eg. gravity, viscosity, thermal properties etc.

From: https://www.slideshare.net/JibranHaider/openfoam-for-beginners

OpenFOAM folder structure

Directory structure



2. *system*:
   This directory contains all the parameters associated with the solution procedure. It contains at least the following files:
   - 2.1 *controlDict*:
     Specifies the run control parameters such as start/end time, time step, write interval etc.
   - 2.2 *fvSchemes*:
     Contains the finite volume discretisation schemes used for the solution procedure such as spatial and temporal discretisations.
   - 2.3 *fvSolution*:
     Contains equation solvers, algorithm controls and tolerances for the implicit solvers.
3. 0:
   The '0' directory corresponds to zero time. It contains the initial and boundary conditions for variables (ie. pressure p, velocity U) in individual files.

From: https://www.slideshare.net/JibranHaider/openfoam-for-beginners

```
blockMesh [project folder]

snappyHexMesh [project folder]
```

Radiance?

```
daylight-factor [project folder]

five-phase [project folder] --view back_view.vf
```

Good old days

RAD

NAME

rad - render a RADIANCE scene

SYNOPSIS

rad [**−s**][**-n** | **-N npr**][**−t**][**−e**][**−V**][**−w**][**−v view**][**−o device**] **rfile** [**VAR=value ..**]

DESCRIPTION

Rad is an executive program that reads the given *rfile* and makes appropriate calls to *oconv(1), mkillum(1), rpict(1), pfilt(1),* and/or *rvu(1)* to render a specific scene. Variables in *rfile* give input files and qualitative information about the rendering(s) desired that together enable *rad* to intelligently set parameter values and control the simulation.

Normally, commands are echoed to the standard output as they are executed. The *−s* option tells *rad* to do its work silently. The *−n* option tells *rad* not to take any action (ie.

From: https://floyd.lbl.gov/radiance/man_html/rad.1.html

Good old days

DAYFACT

NAME

dayfact - compute illuminance and daylight factor on workplane

SYNOPSIS

dayfact [falsecolor options]

DESCRIPTION

Dayfact is an interactive script for computing workplane illuminance, and daylight factors and potential daylight savings using *rtrace(1)*. The script *falsecolor(1)* is then used to draw contour lines on the resulting Radiance picture.

AUTHOR

Greg Ward

ACKNOWLEDGEMENT

Work on this program was initiated and sponsored by the LESO group at EPFL in Switzerland.

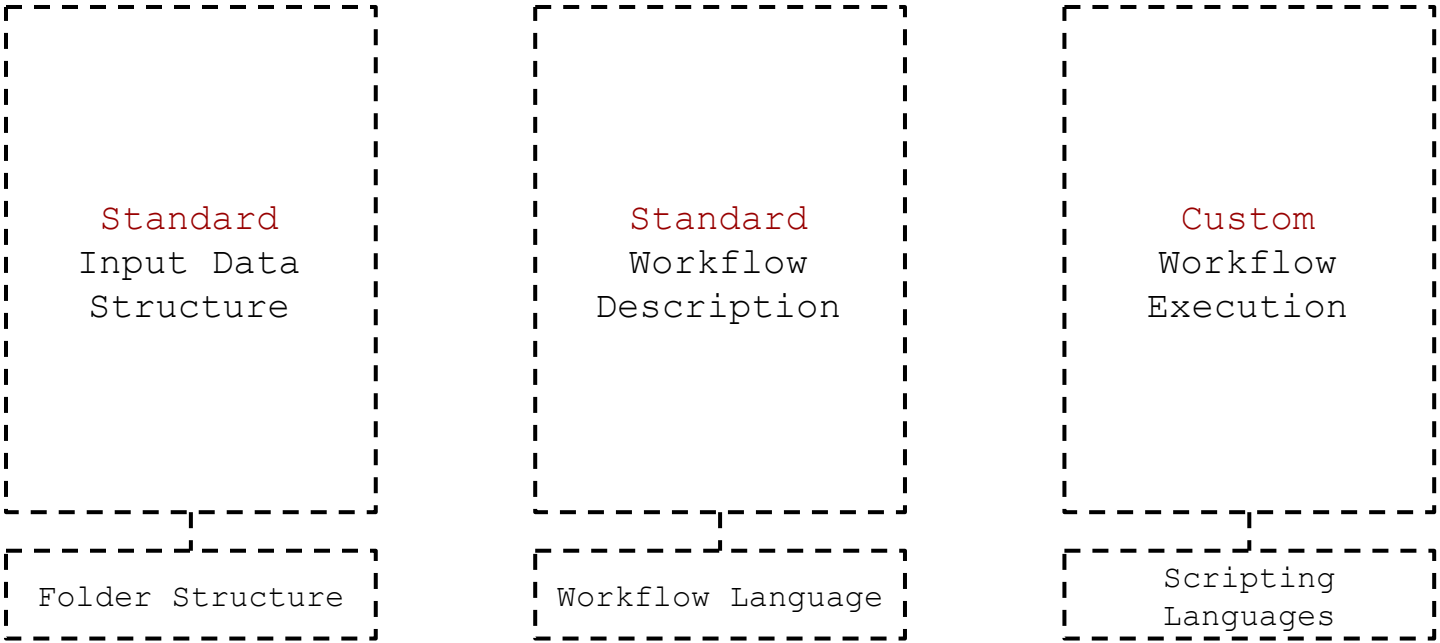
From: https://floyd.lbl.gov/radiance/man_html/dayfact.1.html

Who and where?

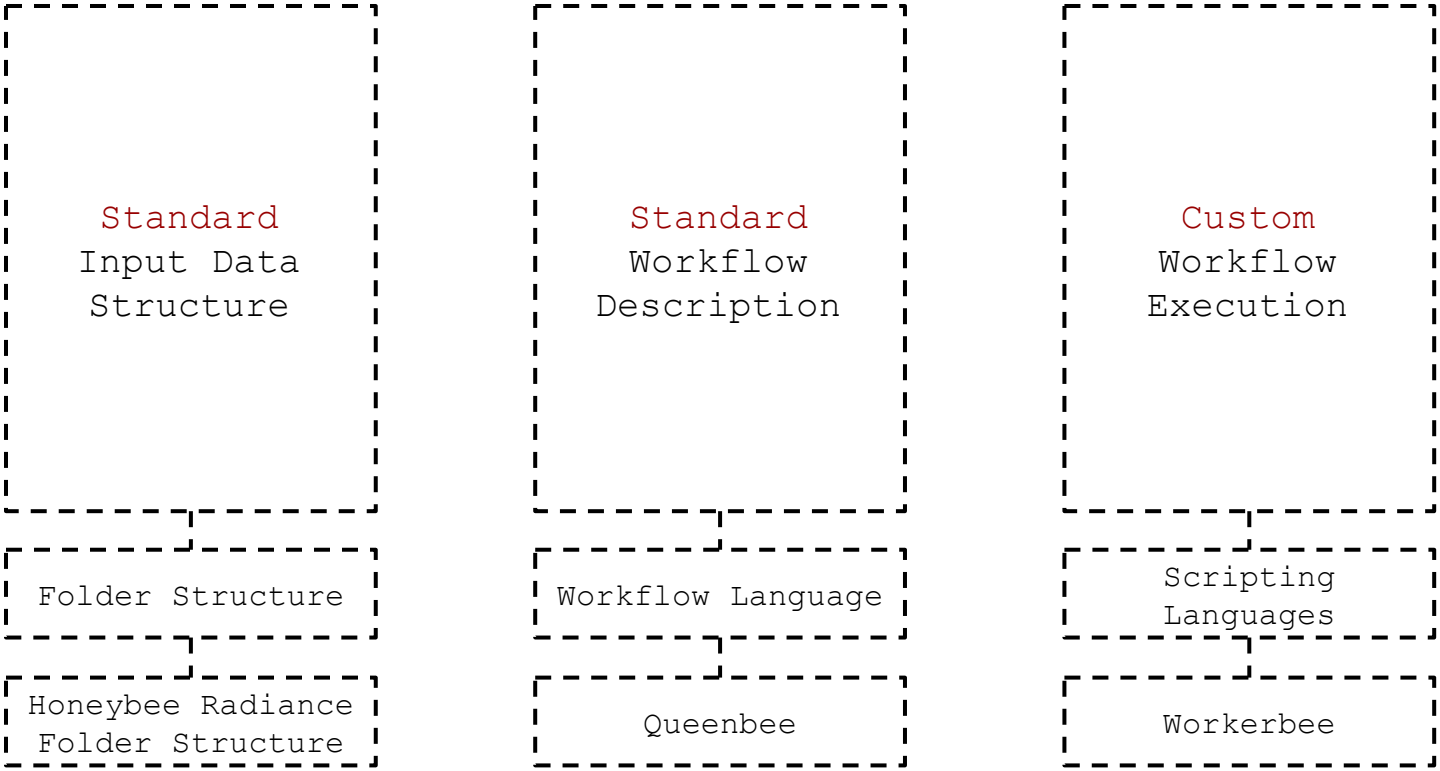
1. Who is going to implement the commands?
2. How to ensure several implementation of the same command?
3. ...

Proposal - Separation of concerns

Proposal - Separation of concerns

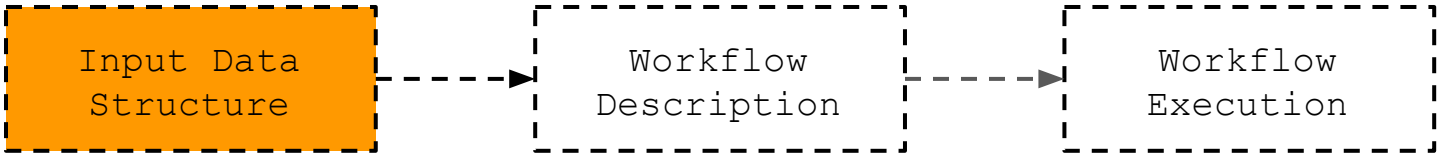


Proposal - Separation of concerns

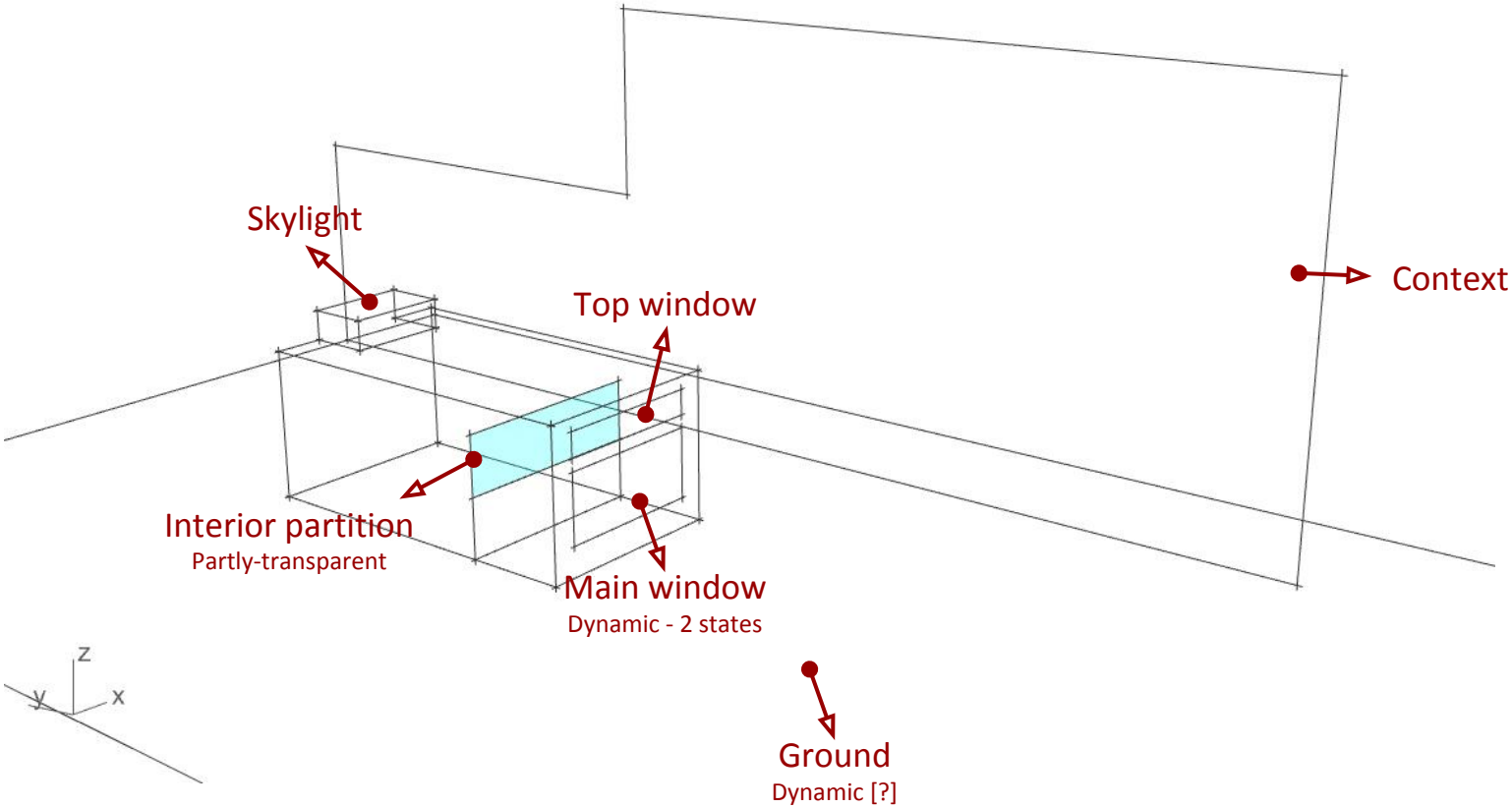


1. Radiance Folder Structure

http://github.com/ladybug-tools/radiance-folder-structure



Sample Model

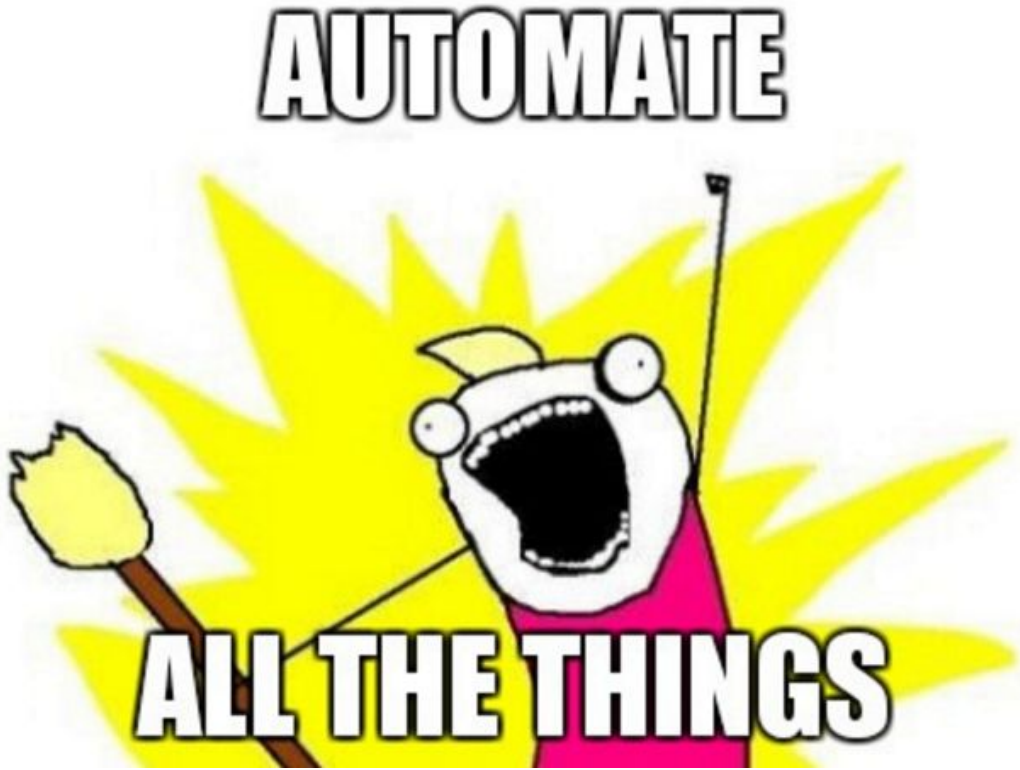


Radiance Folder Structure

http://github.com/ladybug-tools/radiance-folder-structure

Radiance Folder Structure - Summary

1. Mixture of folders and small YAML / JSON files
2. Flexible to support simple and advanced workflows
3. Supports dynamic models
4. Possible to create manually
5. Possible to generate programmatically


AUTOMATE
ALL THE THINGS

Honeybee Radiance Folder

http://github.com/ladybug-tools/honeybee-radiance-folder

2. Queenbee - Workflow Language

http://github.com/ladybug-tools/queenbee

What is workflow language?

"[Workflow language] is a way to describe command line tools and connect them together to create workflows."

From: http://www.commonwl.org/user_guide/01-introduction/index.html

What is workflow language?

"[It] is a specification and not a specific piece of software, tools and workflows described using [WL] are portable across variety of platforms that support the language standard."

From: http://www.commonwl.org/user_guide/01-introduction/index.html

Workflow language samples

```
#!/usr/bin/env cwl-runner

cwlVersion: v1.0
class: CommandLineTool
baseCommand: echo
inputs:
  message:
    type: string
    inputBinding:
      position: 1
outputs: []
```

From: http://www.commonwl.org/user_guide/02-1st-example/index.html

Workflow language samples

```
# The following workflow executes a diamond workflow
#
#   A           A: Create Octree
#  / \
# B   C         B: Raytrace for sensor, C: Raytrace for view
#  \ /
#   D           D: Results post-processing

apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: dag-diamond-
spec:
  entrypoint: diamond
```

Modified from: https://github.com/argoproj/argo/blob/master/examples/dag-diamond.yaml

Workflow language samples

```
[ … ]
templates:
- name: diamond
  dag:
  tasks:
  - name: A-create-octree
      template: echo
      arguments:
      parameters: [{name: message, value: creating octree}]
  - name: B-sensor-ray-tracing
      dependencies: [A-create-octree]
      template: echo
      arguments:
      parameters: [{name: message, value: B-sensor-ray-tracing}]
```

Modified from: http://www.commonwl.org/user_guide/02-1st-example/index.html

Workflow language samples

```
[ … ]

- name: C-view-ray-tracing
    dependencies: [A-create-octree]
    template: echo
    arguments:
    parameters: [{name: message, value: C-view-ray-tracing}]
- name: D-postprocessing
    dependencies: [B-sensor-ray-tracing, C-view-ray-tracing]
    template: echo
    arguments:
    parameters: [{name: message, value: D-post-processing}]
```

Modified from: http://www.commonwl.org/user_guide/02-1st-example/index.html

Workflow language samples

```
[ … ]

- name: echo
  inputs:
  parameters:
  - name: message
  container:
  image: alpine:3.7
  command: [echo, "{{inputs.parameters.message}}"]
```

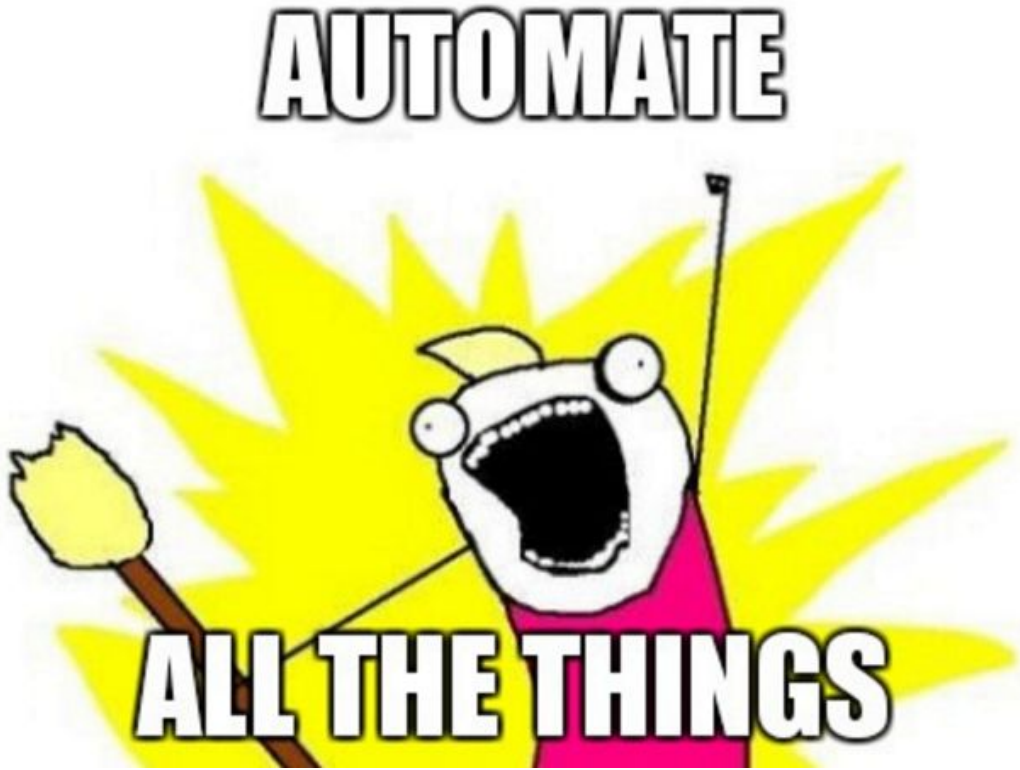
Modified from: http://www.commonwl.org/user_guide/02-1st-example/index.html

Queenbee - Workflow Language

http://github.com/ladybug-tools/queenbee

Queenbee - Summary

1. A language to define workflows
2. Supports local execution and using containers
3. Human readable!
4. Possible to create and modify manually
5. Using OpenAPI for documentation schemas
6. Possible to generate programmatically
7. Possible to visualize and validate!


AUTOMATE
ALL THE THINGS

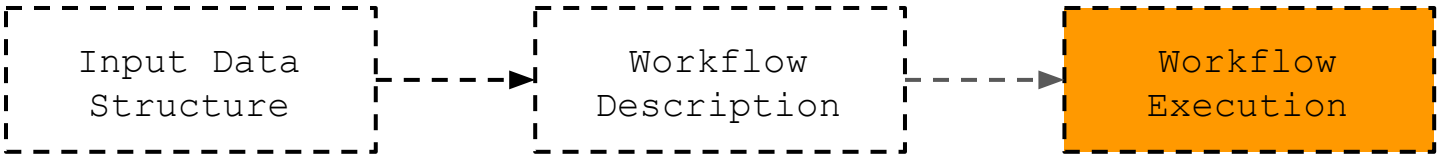
honeybee-radiance-workflow

A Collection of Queenbee Workflows for Radiance

http://github.com/ladybug-tools/honeybee-radiance-workflow

3. Workerbee - Workflow Executor

http://github.com/ladybug-tools/workerbee

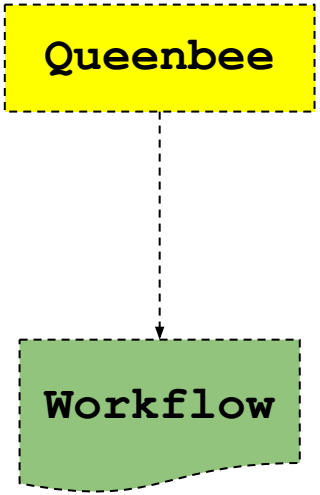


In a couple of months from now

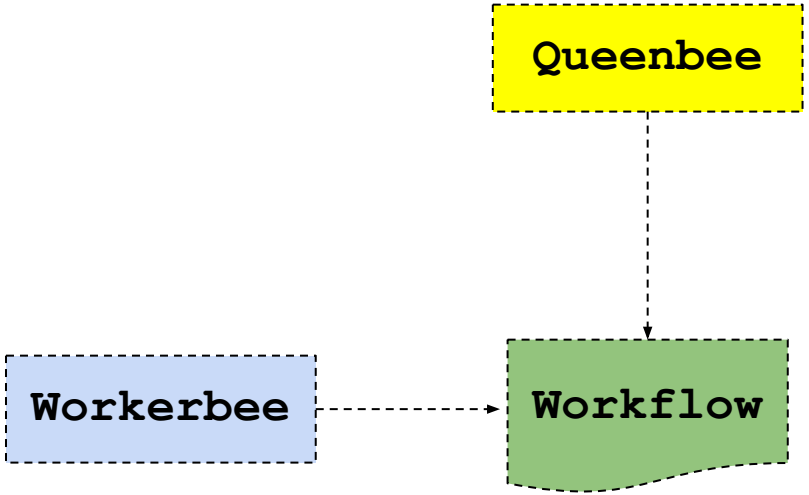
```
queenbee validate five-phase.yaml [project folder]

workerbee luigi five-phase.yaml [project folder] --view back_view.vf
```

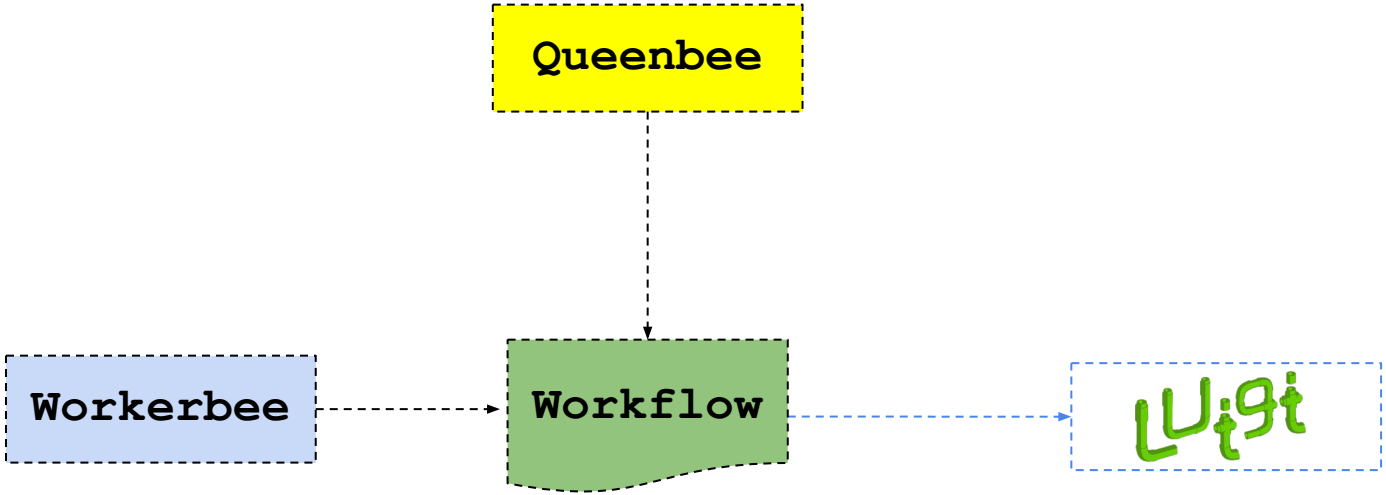
Workerbee: Execute workflows



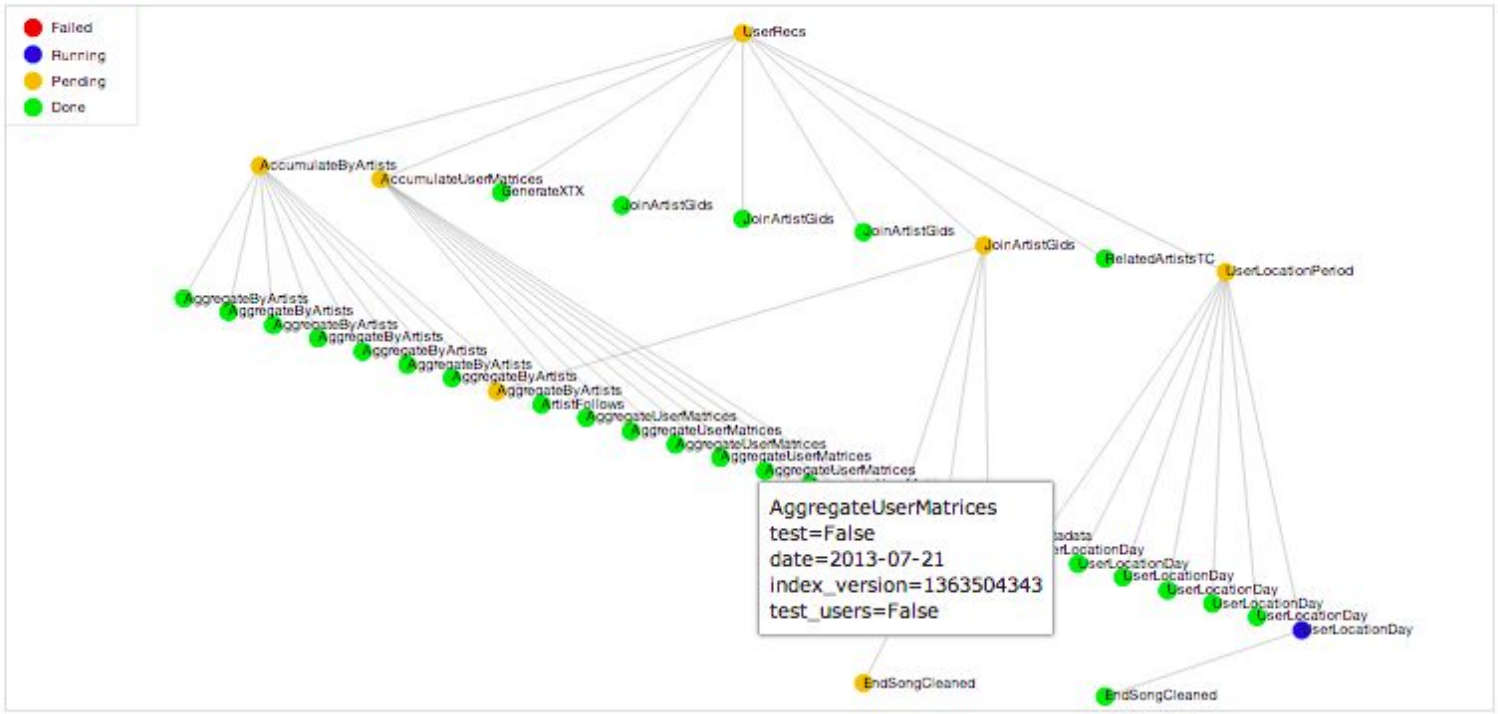
Workerbee: Execute workflows



Workerbee: Execute workflows

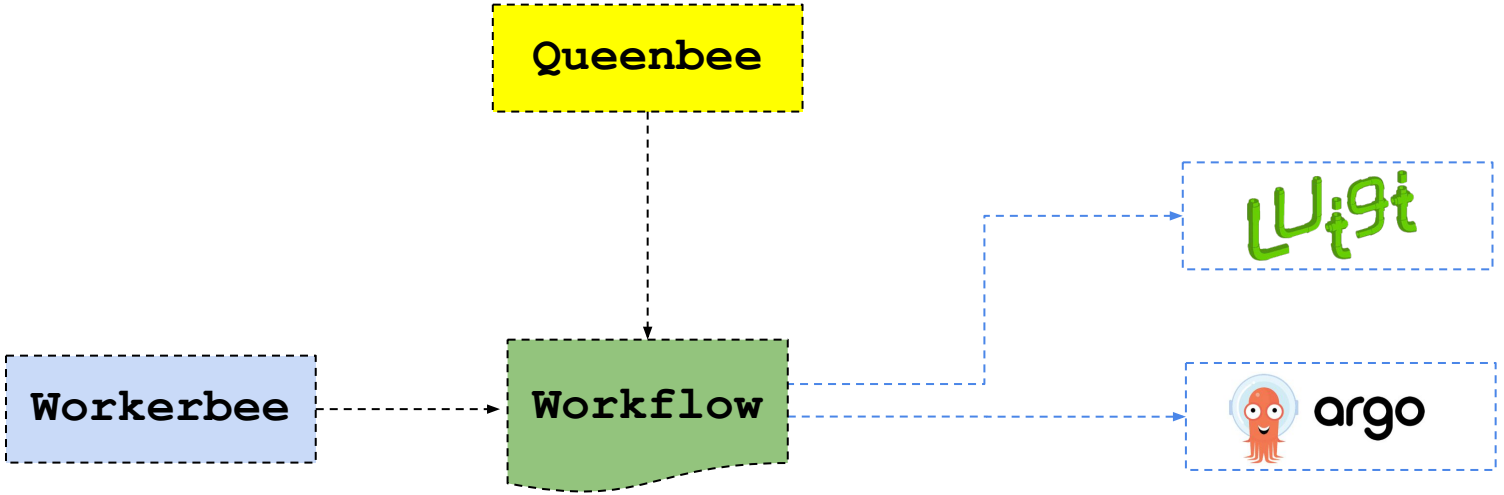


Workerbee: Execute workflows locally

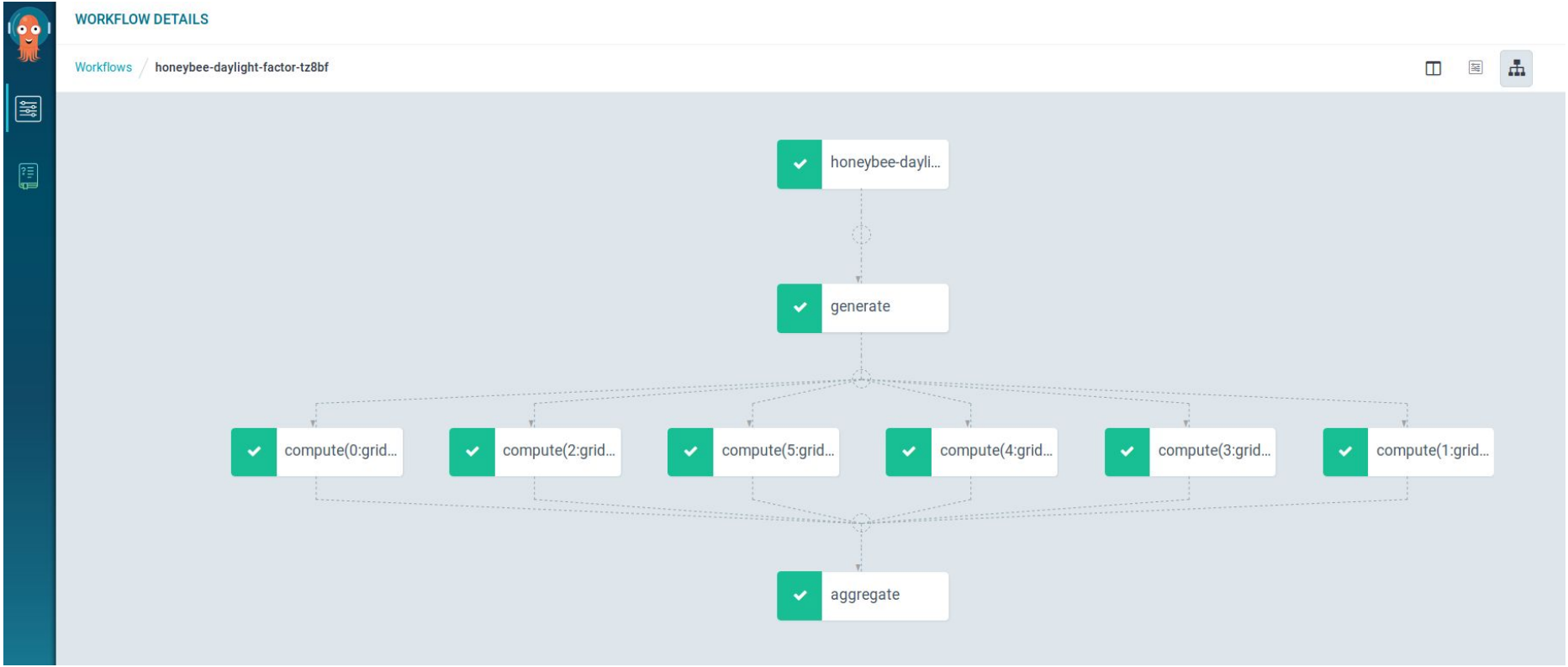


https://luigi.readthedocs.io/en/stable/

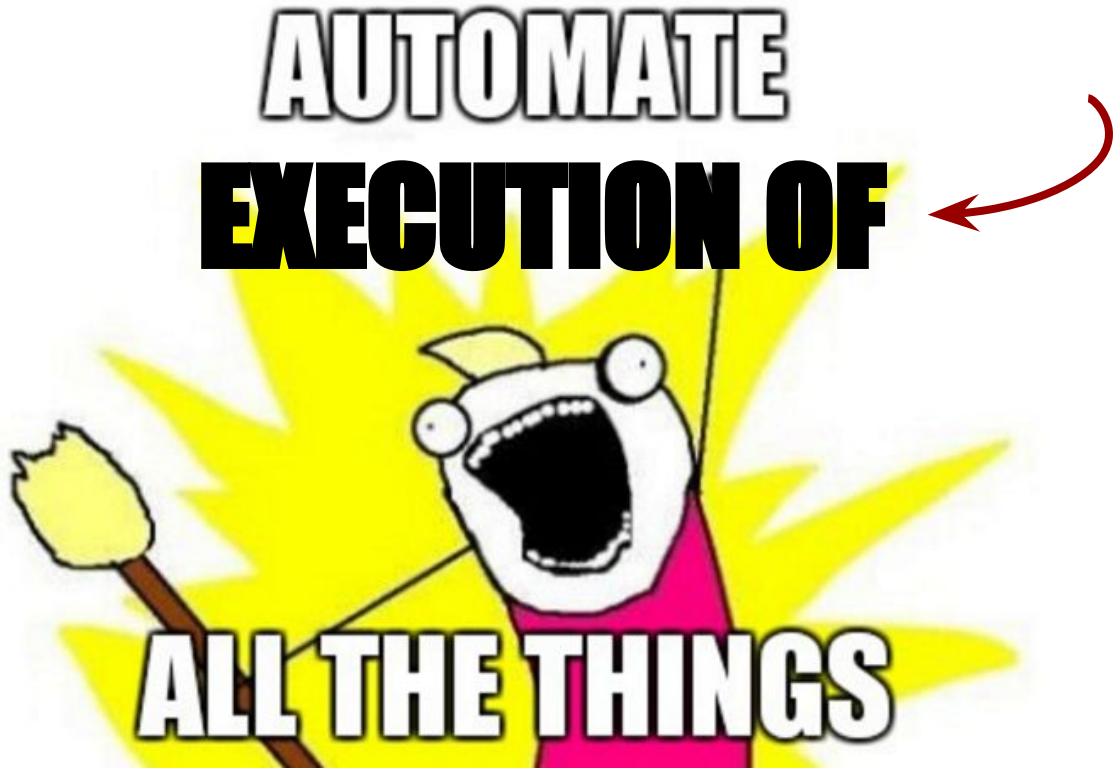
Workerbee: Execute workflows



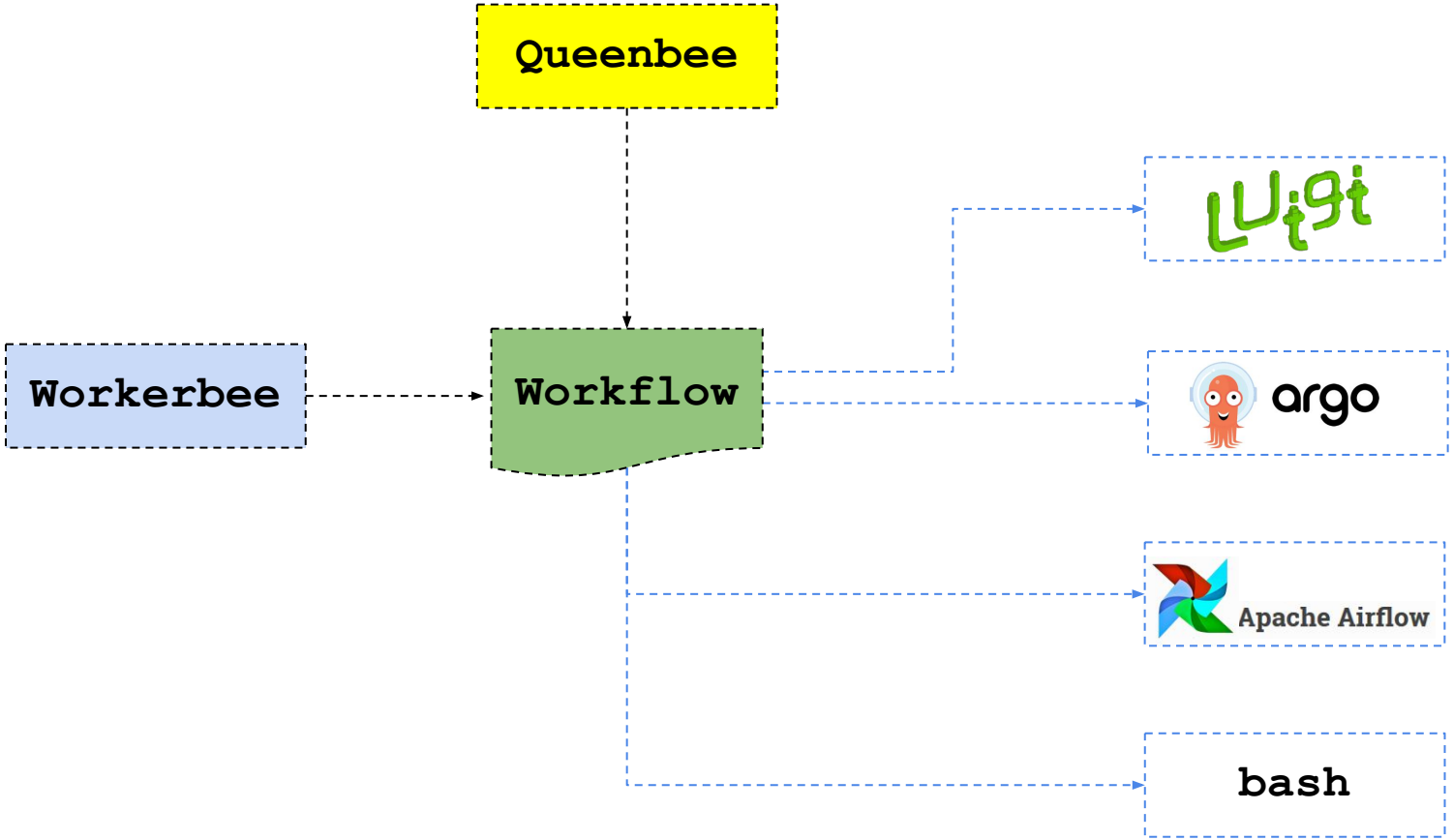
Workerbee: Execute workflows on cloud



https://argoproj.github.io/


AUTOMATE
EXECUTION OF
ALL THE THINGS

Workerbee: Execute workflows everywhere



Get involved

http://github.com/ladybug-tools/radiance-folder-structure

http://github.com/ladybug-tools/honeybee-radiance-folder

http://github.com/ladybug-tools/queenbee

http://github.com/ladybug-tools/workerbee

Get involved

Thank you!

@ladybug_tools

@_pollination