Pacific Northwest
NATIONAL LABORATORY

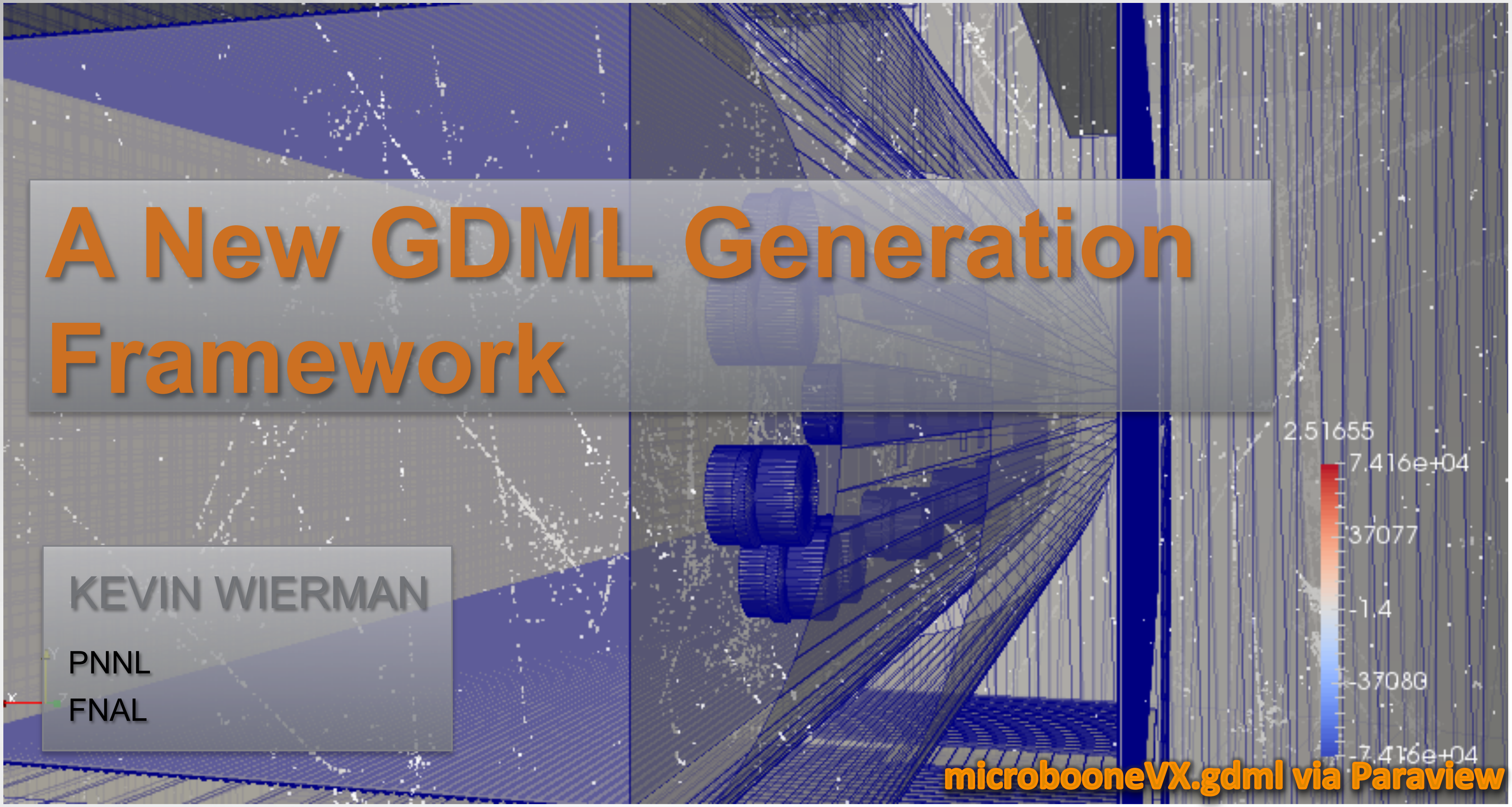
Proudly Operated by Battelle *Since 1965*

A New GDML Generation Framework

KEVIN WIERMAN

PNNL

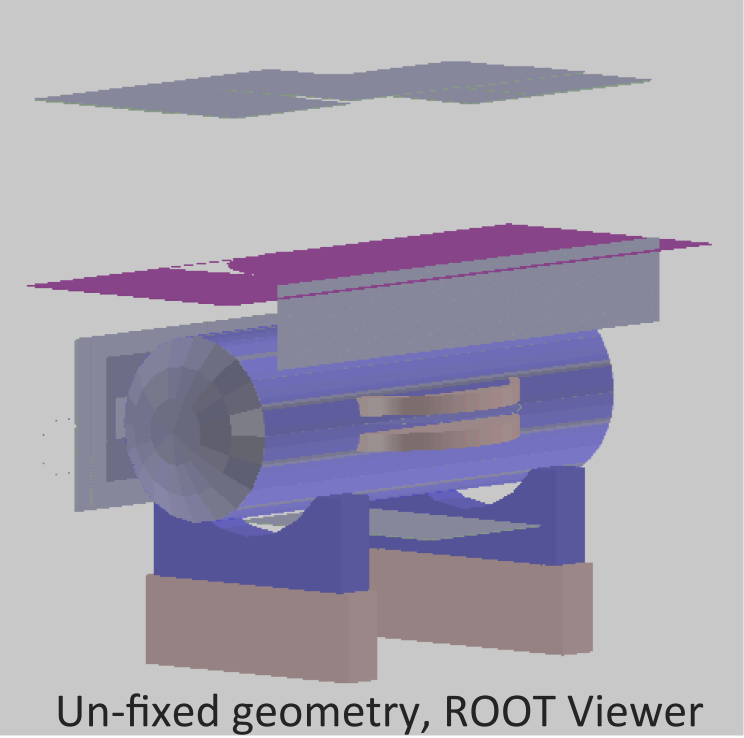
FNAL



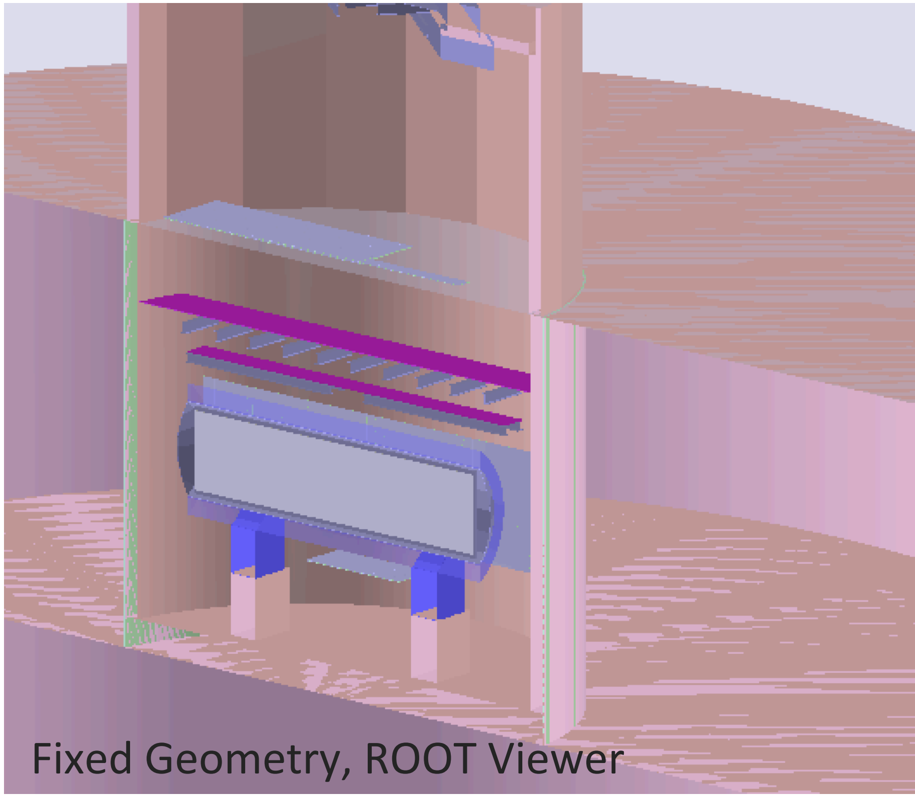
microbooneVX.gdml via Paraview

The Current GDML Generation Scheme

- Perl-based GDML Script (uBooNE)
  - Affected by changes in ROOT
  - Throws warnings/not errors
  - Difficult to diagnose
- Python based solutions
  - Multiple scripts exist
  - Documentation exists, but not obvious
  - They all seem to do the same thing



Un-fixed geometry, ROOT Viewer



Fixed Geometry, ROOT Viewer

The Bottom Line

- The perl scripts produce broken geometry
  - Due to formula evaluation
  - Debugging not easy
- ROOT TGeo may be fixed in the future
  - Not exactly helpful in debugging geometry issues
- GDML is the supported interface to LArG4
  - I'm currently treating this as immutable
- Perl is not a supported product
- However, Python2.7 is a supported product
- It would be nice to have something easy to debug

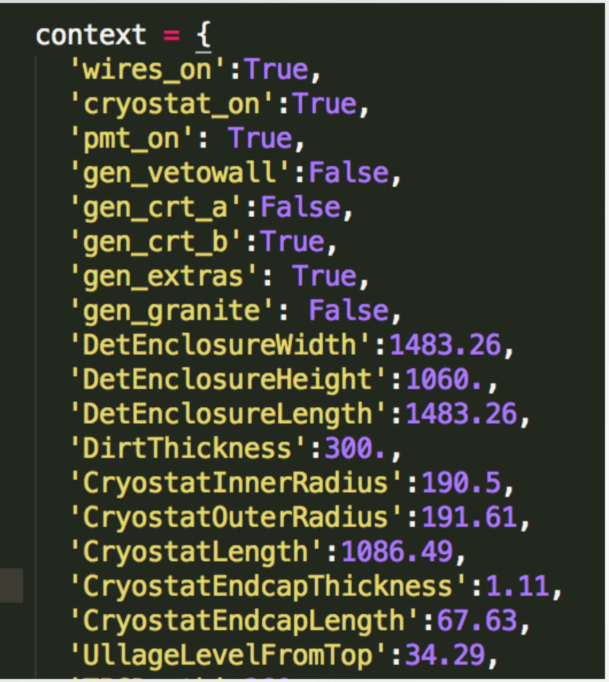
What is the (non-temporary) solution?

Proposed Solution: Mako Templates

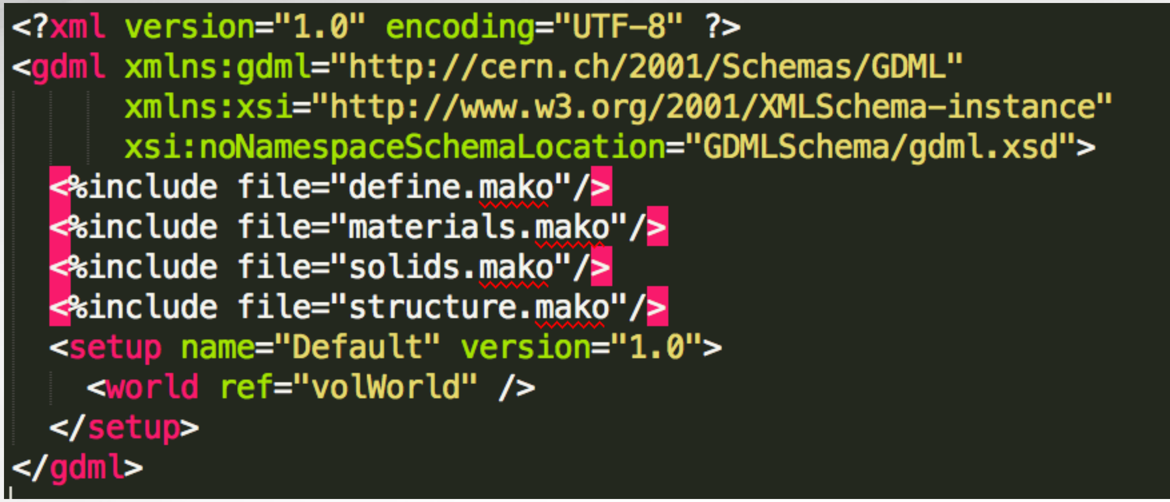
- X/HTML Document generator
  - Python package
  - Uses XML templates to expand small modules to full documents
  - Comes with many features that we can use

How to Use Templates

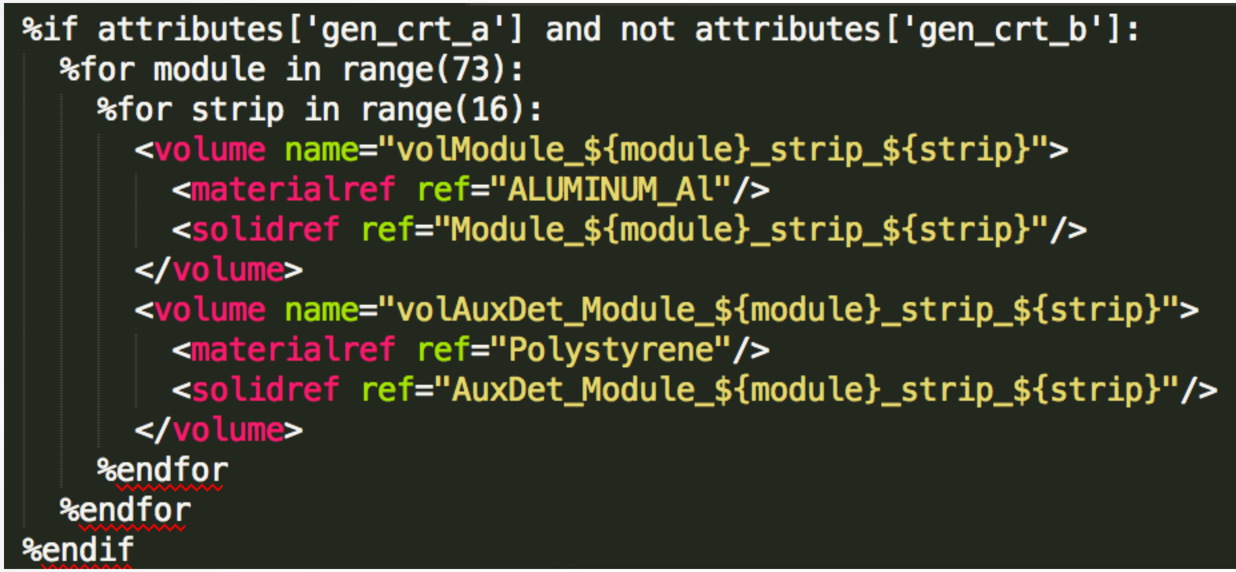
- Define a context
- Write a top-level file
- Expand



```
context = {
    'wires_on':True,
    'cryostat_on':True,
    'pmt_on': True,
    'gen_vetowall':False,
    'gen_crt_a':False,
    'gen_crt_b':True,
    'gen_extras': True,
    'gen_granite': False,
    'DetEnclosureWidth':1483.26,
    'DetEnclosureHeight':1060.,
    'DetEnclosureLength':1483.26,
    'DirtThickness':300.,
    'CryostatInnerRadius':190.5,
    'CryostatOuterRadius':191.61,
    'CryostatLength':1086.49,
    'CryostatEndcapThickness':1.11,
    'CryostatEndcapLength':67.63,
    'UllageLevelFromTop':34.29,
```



```
<?xml version="1.0" encoding="UTF-8" ?>
<gdml xmlns:gdml="http://cern.ch/2001/Schemas/GDML"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="GDMLSchema/gdml.xsd">
  <%include file="define.mako"/>
  <%include file="materials.mako"/>
  <%include file="solids.mako"/>
  <%include file="structure.mako"/>
  <setup name="Default" version="1.0">
    <world ref="volWorld" />
  </setup>
</gdml>
```

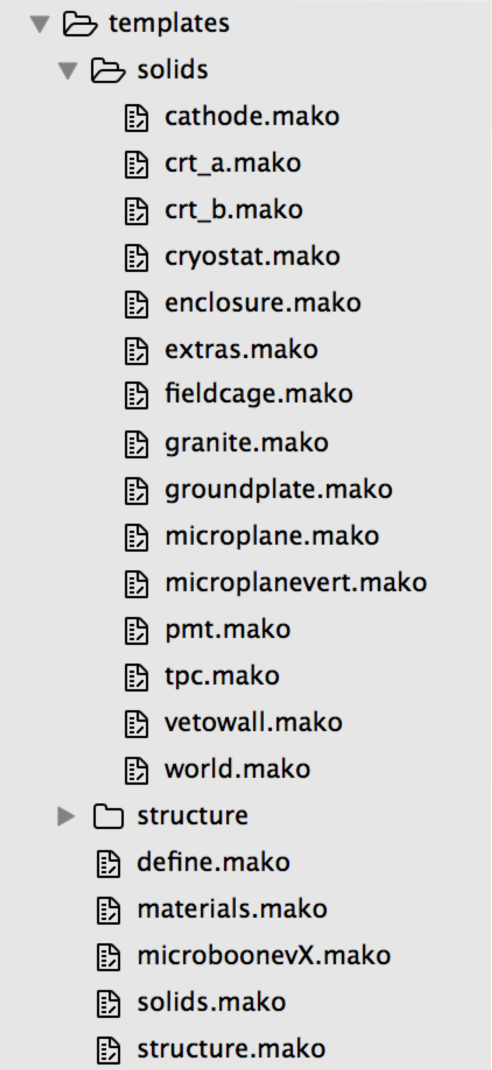


```
%if attributes['gen_crt_a'] and not attributes['gen_crt_b']:
  %for module in range(73):
    %for strip in range(16):
      <volume name="volModule_${module}_strip_${strip}">
        <materialref ref="ALUMINUM_Al"/>
        <solidref ref="Module_${module}_strip_${strip}"/>
      </volume>
      <volume name="volAuxDet_Module_${module}_strip_${strip}">
        <materialref ref="Polystyrene"/>
        <solidref ref="AuxDet_Module_${module}_strip_${strip}"/>
      </volume>
    %endfor
  %endfor
%endif
```

Benefits of Using Mako Templates

- Inheritance and includes
  - More manageable code blocks
- Python statement evaluation
  - Formula expression based generation
- Control Sequences
  - Modifiable geometry at generation
- Ships with debugging tools:

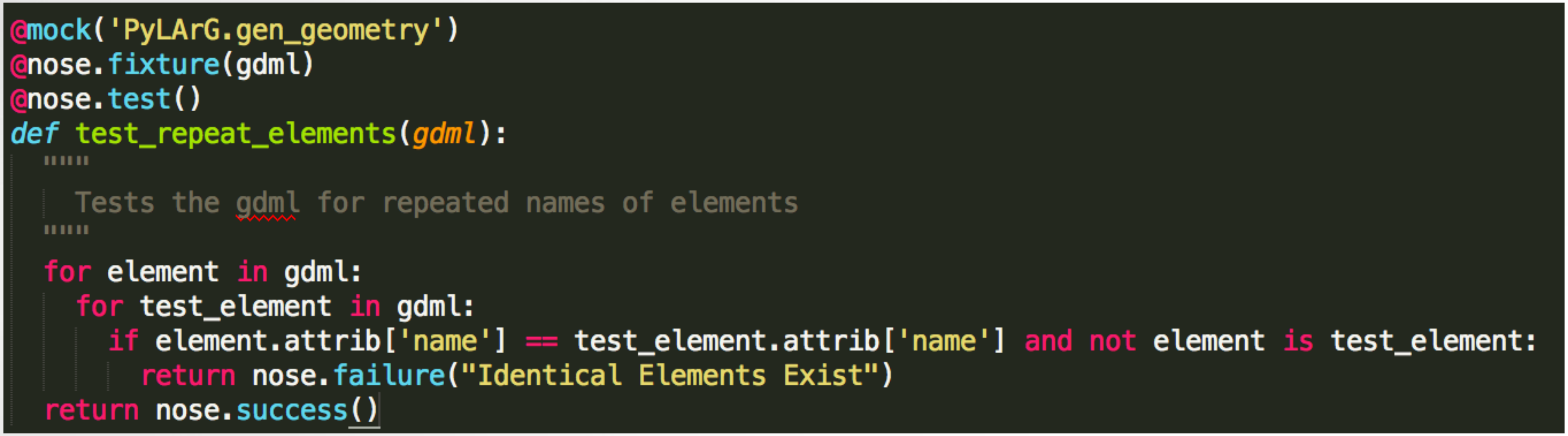
```
from mako import exceptions
try:
    template = lookup.get_template(uri)
    return template.render()
except:
    return exceptions.text_error_template().render()
```



```
▼ templates
  ▼ solids
      cathode.mako
      crt_a.mako
      crt_b.mako
      cryostat.mako
      enclosure.mako
      extras.mako
      fieldcage.mako
      granite.mako
      groundplate.mako
      microplane.mako
      microplanevert.mako
      pmt.mako
      tpc.mako
      vetowall.mako
      world.mako
  ▶ structure
    define.mako
    materials.mako
    microboonevX.mako
    solids.mako
    structure.mako
```

Bonus: Unit Tests

- Base code is python
  - Current tests in nose/mock
  - Easily convert to py.test or unittest



```
@mock('PyLArG.gen_geometry')
@nose.fixture(gdml)
@nose.test()
def test_repeat_elements(gdml):
  """
    Tests the gdml for repeated names of elements
  """
  for element in gdml:
    for test_element in gdml:
      if element.attrib['name'] == test_element.attrib['name'] and not element is test_element:
        return nose.failure("Identical Elements Exist")
  return nose.success()
```

Current Status

- A basic version of the code exists in:
  - uboonecode:feature/kwierman_geo_overhaul
- An external copy exists in github repo: kwierman/PyLArG
  - Also comes with gdml->vtk conversion utilities
    - This is a conversation for another time
- Comes with compare.py
  - Compares 2 xml trees
  - Shows that mako generator and perl script (with fix!!) creates identical geometry

Goals

- Include mako1.0.6 in supported python packages
- Re-write generator script to be generic for multiple experiments
  - Thus, experiments will only need to contribute context & templates
- Include unit tests in LArSoft TDD framework
- Convert context to fhicl
- Document
  - Document
    - Document

Other Option

- Brett Viren (DUNE):
  - GeGeDe (https://github.com/brettviren/gegede)
  - Dune Specific Version (https://github.com/dune/duneggd)
  - Allows for python based generation of geometry and export to a variety of formats
- Options for future development
  - Include VTK export option
  - Use mako templates for individual objects

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by **Battelle** *Since 1965*

Questions?