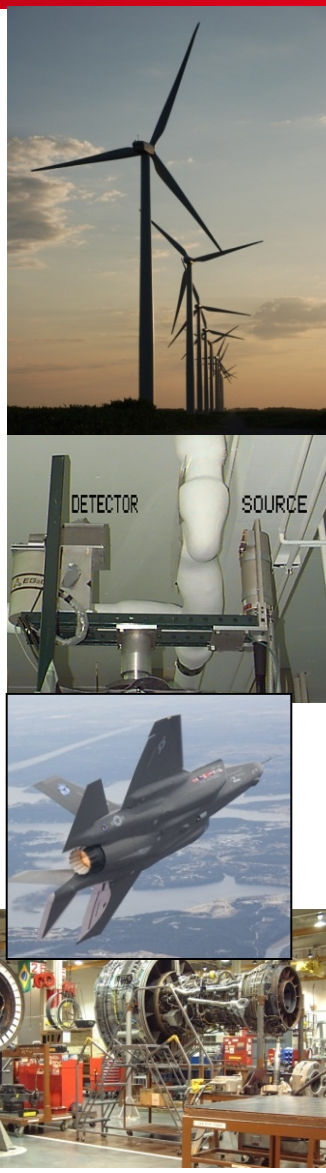
IOWA STATE UNIVERSITY

Center for Nondestructive Evaluation

ProcessTrak

Stephen. D. Holland

4/22/2016



This material is based on work supported by the Air Force Research Laboratory under Contract #FA8650-10-D-5210, Task Order #023, and performed at Iowa State University

Motivation

- Lab data processing or analysis is easy when you can collect all steps into a single MATLAB or Python script that runs quickly.
- Once the steps are complicated enough or time consuming enough that you can't just re-run them each time, you end up with a mess.
- ProcessTrak provides a framework for avoiding/minimizing the mess

The mess

- With processing split into multiple scripts, you end up with a lot of intermediate files.
- It is often unclear which intermediates come from which script and/or which order the scripts must be run to recreate the final output
- It is often unclear that all scripts have been run in the correct order and therefore unclear that the final output is correct and meaningful.

The solution

- Based on datacollect2, represent experiment log as XML
- Processing starts by copying raw experiment log (.xlg) to processed experiment log (.xlp)
- Each processing step operates ***in place*** on the processed experiment log (.xlp).
- Processing steps defined in the .prx file, generally written in Python.
- Can be re-executed out-of-order as needed.
- Provenance-based result integrity checking.

Example .prx file

```
<processinginstructions xmlns=
"http://limatix.org/processtrak/processinginstructions"
xmlns:xlink="http://www.w3.org/1999/xlink">

  <inputfiles>
    <inputfile xlink:href="example_input.xlg"/>
  </inputfiles>
  <!-- elementmatch says what elements in the experiment log to
       Iterate over by default -->
  <elementmatch>xpath_condition</elementmatch>
  <step name="pullparams" descr="Pull in parameters from other files">
    <script xlink:href="pullparams.py">
      <!-- Parameters can be generated from other elements of the
           Experiment log, or explicitly in the .prx file: -->
      <parameter name="foo">
       <numvalue>34.5</numvalue>
      </parameter>
      <parameter name="bar">
       <strvalue>BAR</strvalue>
      </parameter>
    </script>
  </step>
</processinginstructions>
```

Example script

```
import sys
import numpy as np
from limatix.dc_value import numericunitsvalue

def run(_xmldoc,_element,dc_magnitudesquared_numericunits):
    magsq=dc_magnitudesquared_numericunits.value('Volts^2')

    mag=np.sqrt(magsq)

    return { “dc:magnitude”: numericunitsvalue(mag,'Volts') }
```

- This example script reads the 'dc:magnitudesquared' element
  - (or the dc_magnitudesquared_numvalue parameter if given in the .prx file)
- converts the value to volts^2, takes the square root, and stores the square root in the 'dc:magnitude' element.