Updating to a GNDS-based Nuclear Data Workflow

Caleb Mattoon
Nuclear Data and Theory Group, NACS/PHY

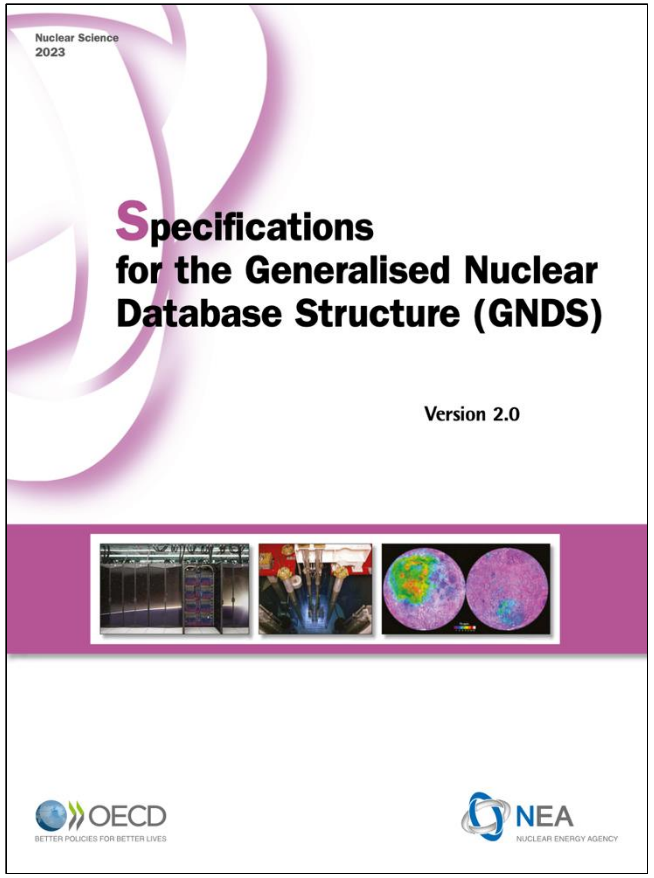
WANDA 2025
February 12, 2025

LLNL-PRES-2002648
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

The Generalized Nuclear Database Structure (GNDS) is a new standard for storing and exchanging nuclear reaction data.

- GNDS-1.9 was published in 2020, GNDS-2.0 in 2023.
  - With extensive input from an international collaboration at WPEC!
  - **GNDS-2.1:** we're exploring publishing it as a working paper.
- Challenge now: make the transition from ENDF-6 to GNDS smooth while preserving or **expanding** old capabilities
- We're making good progress!
  - ENDF-6 ⟺ GNDS translator / error checker
  - several major libraries available in both ENDF-6 and GNDS
  - growing list of nuclear data tools support checking / processing / visualizing / using GNDS data
  - extensive work with using GNDS-based pipeline and comparing to previous results


Nuclear Science
2023

Specifications for the Generalised Nuclear Database Structure (GNDS)

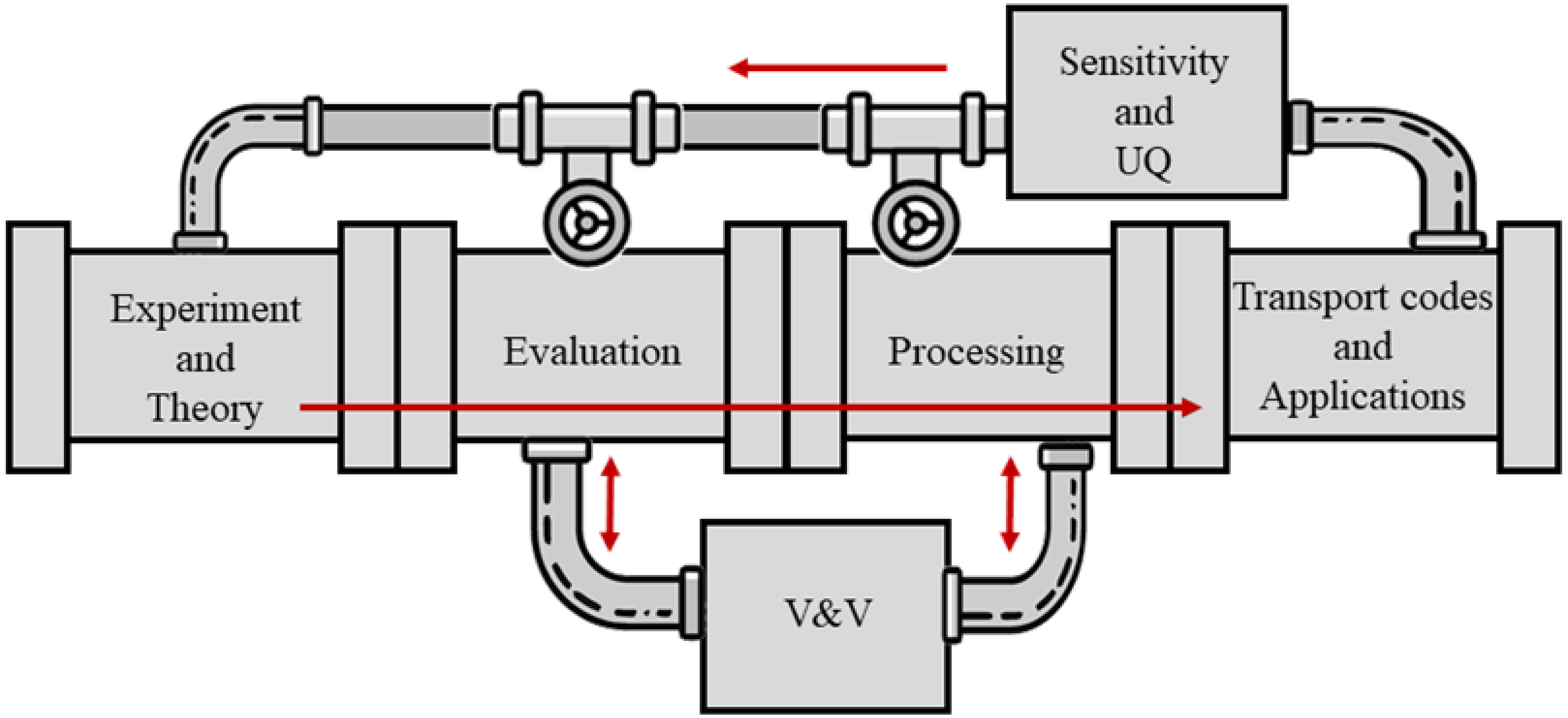
Version 2.0

OECD
BETTER POLICIES FOR BETTER LIVES

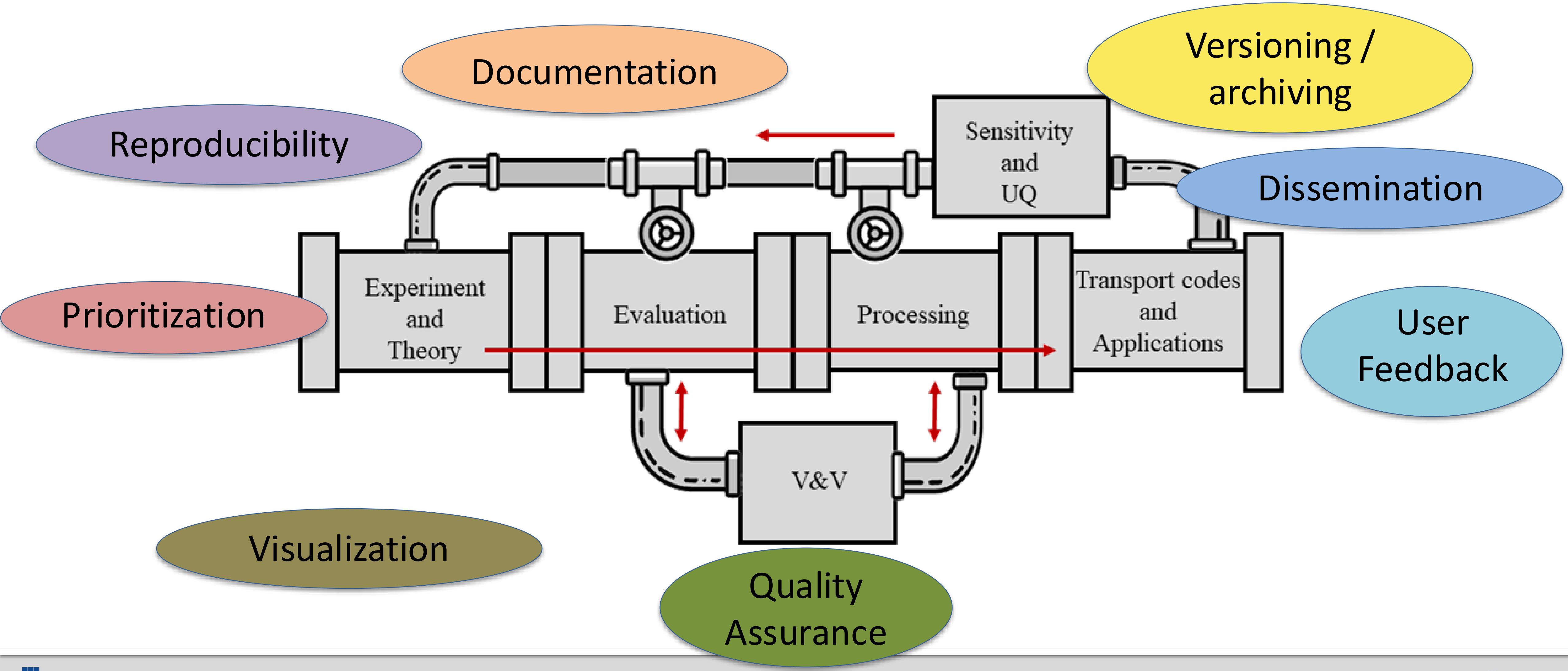
NEA
NUCLEAR ENERGY AGENCY

The transition to GNDS is an opportunity to *revise* and *improve* the nuclear data workflow

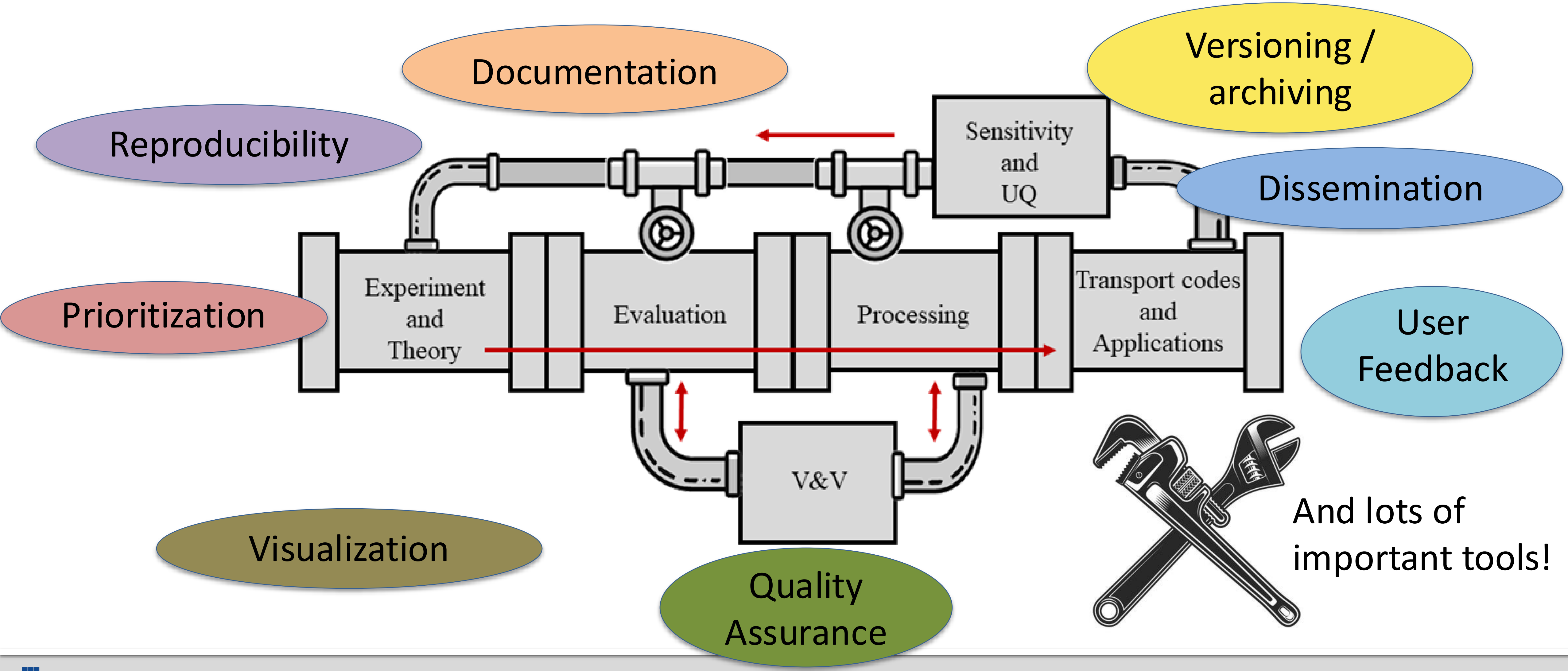
Nuclear data *workflow* follows the familiar *pipeline* analogy,



Nuclear data *workflow* follows the familiar *pipeline* analogy, but with several extra areas of emphasis:

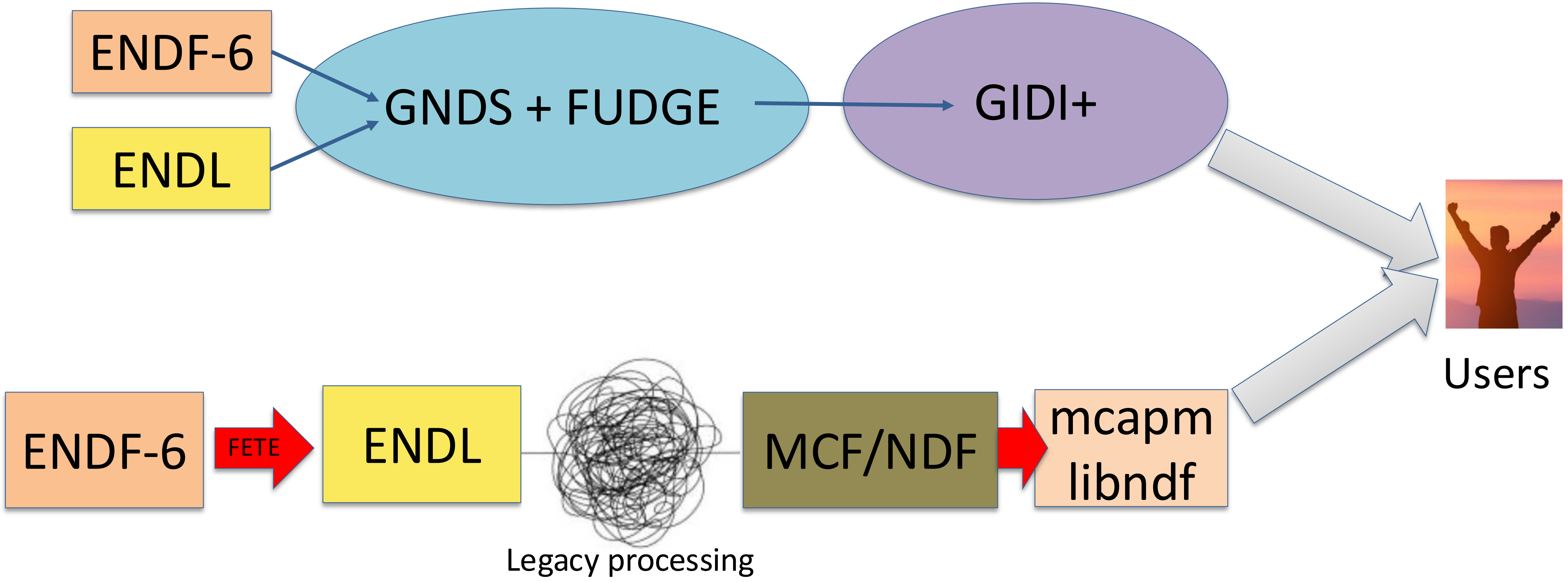


Nuclear data *workflow* follows the familiar *pipeline* analogy, but with several extra areas of emphasis:



And lots of important tools!

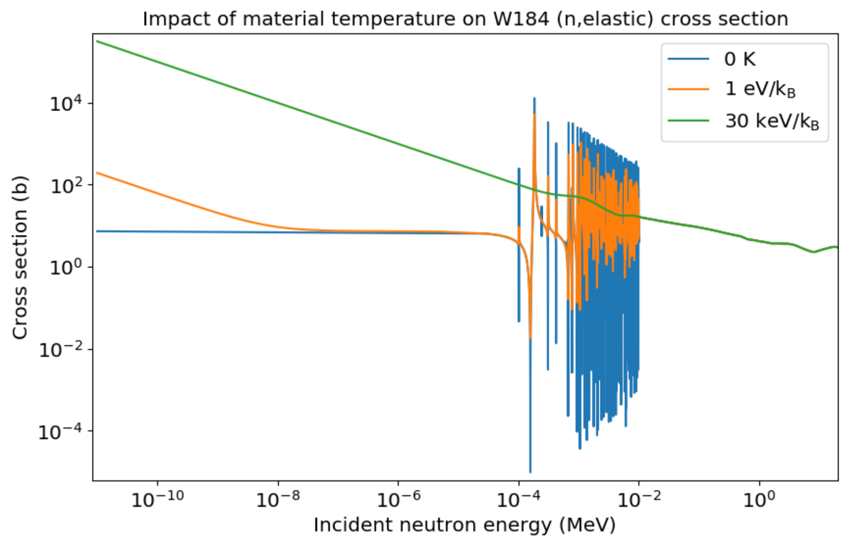
LLNL Nuclear Data and Theory group has supported two separate workflows for the past ~10 years:

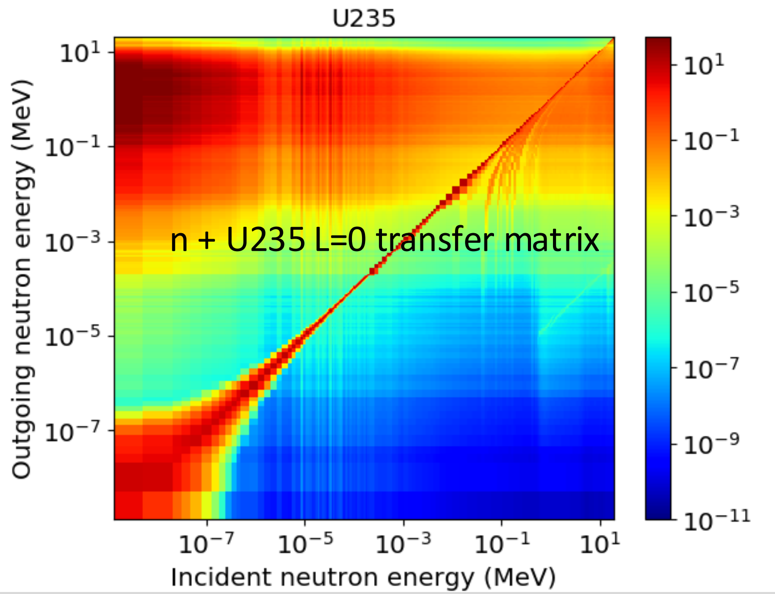


LLNL's new toolkit is focused on reorganizing our workflow around GNDS:

- FUDGE: For Updating Data and Generating Evaluations
  - Python-based code for reading, writing, modifying, viewing and processing nuclear data
  - Computationally intensive routines written in C and C++

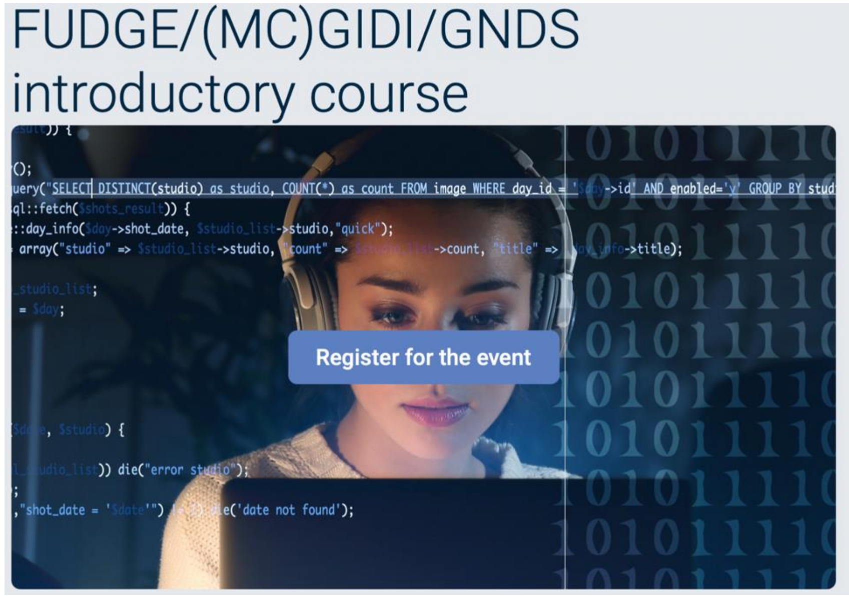
- GIDI+: General Interaction Data Interface+
  - Suite of C++ APIs for accessing GNDS data for use in transport codes
  - Includes API for sampling GNDS data as needed by Monte Carlo codes
  - Used in GEANT-4 (G4LEND) and LLNL codes Ardra and Mercury



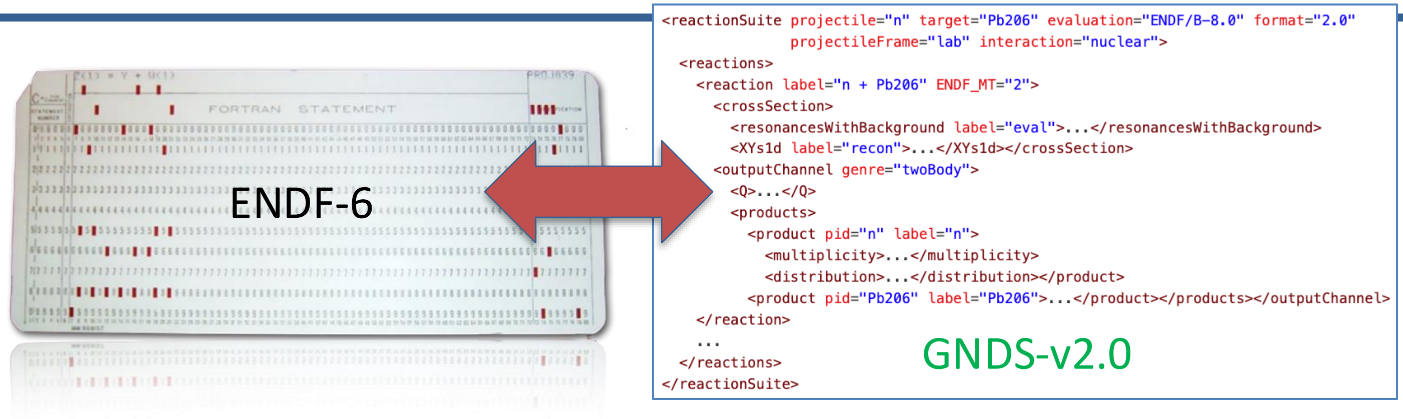


FUDGE-6.8 was released on Github in January 2025

- Available at https://github.com/LLNL/fudge
  - Requires Python3.7 or later, numpy, matplotlib, C++ compiler for extensions.
  - Install with pip or with make
  - Release schedule: around 4 public versions per year
- Recent updates:
  - Full support for translating and processing ENDF/B-VIII.1
  - Improved physics checking tools, focus on diagnosing and fixing energy balance
- Second FUDGE training course will be held June 10 – 13 at the NEA
  - https://www.oecd-nea.org/jcms/pl_98517/fudge/-mc-gidi/gnds-introductory-course



Most GNDS evaluations still start out as ENDF-6 files and are translated using the FUDGE tools *endf2gnds.py* and *rePrint.py*



```xml
<reactionSuite projectile="n" target="Pb206" evaluation="ENDF/B-8.0" format="2.0"
               projectileFrame="lab" interaction="nuclear">
  <reactions>
    <reaction label="n + Pb206" ENDF_MT="2">
      <crossSection>
        <resonancesWithBackground label="eval">...</resonancesWithBackground>
        <XYs1d label="recon">...</XYs1d></crossSection>
      <outputChannel genre="twoBody">
        <Q>...</Q>
        <products>
          <product pid="n" label="n">
            <multiplicity>...</multiplicity>
            <distribution>...</distribution></product>
          <product pid="Pb206" label="Pb206">...</product></products></outputChannel>
    </reaction>
    ...
  </reactions>
</reactionSuite>
```

- Translator loads data into the FUDGE class hierarchy (closely mirroring GNDS), then writes out to either format.
- Translator is strict! Helps uncover format / physics issues in many evaluations.

```
rePrint.py 02-He-003.C32
…
Raising due to following errors:
   WARNING: non-integer ZAP 1.001 encountered in MF
```

Next step: bypass ENDF-6 with evaluator toolkit for working directly with GNDS

- Already have several options:
  - parseYAHFC: generates GNDS from LLNL ‘Yet Another Hauser-Feshback Code’
  - TAGNDS: convert TALYS output to GNDS. Recently expanded to include more of the ‘T6’ code suite
  - Ferdinand: translates between various R-Matrix code inputs, using GNDS as the common exchange format

- Evaluators also need tools for merging / splicing / patching evaluations.
  - May encounter inconsistent target mass / spin / energy levels
  - Expand product distributions to span the full range of the evaluation
  - Beware of double-counting, e.g. between MT=3 photons and specific reactions

GNDS supports finer-grained documentation that we can leverage to improve reproducibility

- Examples of GNDS-2.0 documentation:

```
<documentation
        doi="..."
        publicationDate="..."
        version="...">
    <authors>...</authors>
    <contributors>...</contributors>
    <collaborations>...</collaborations>
    <dates>...</dates>
    <copyright>...</copyright>
    <acknowledgements>...</acknowledgements>
    <keywords>...</keywords>
    <relatedItems>...</relatedItems>
    <title>...</title>
    <abstract>...</abstract>
    <body>...</body>
    <computerCodes>...</computerCodes>
    <experimentalDataSets>...</experimentalDataSets>
    <bibliography>...</bibliography>
    <endfCompatible>...</endfCompatible></documentation>
```

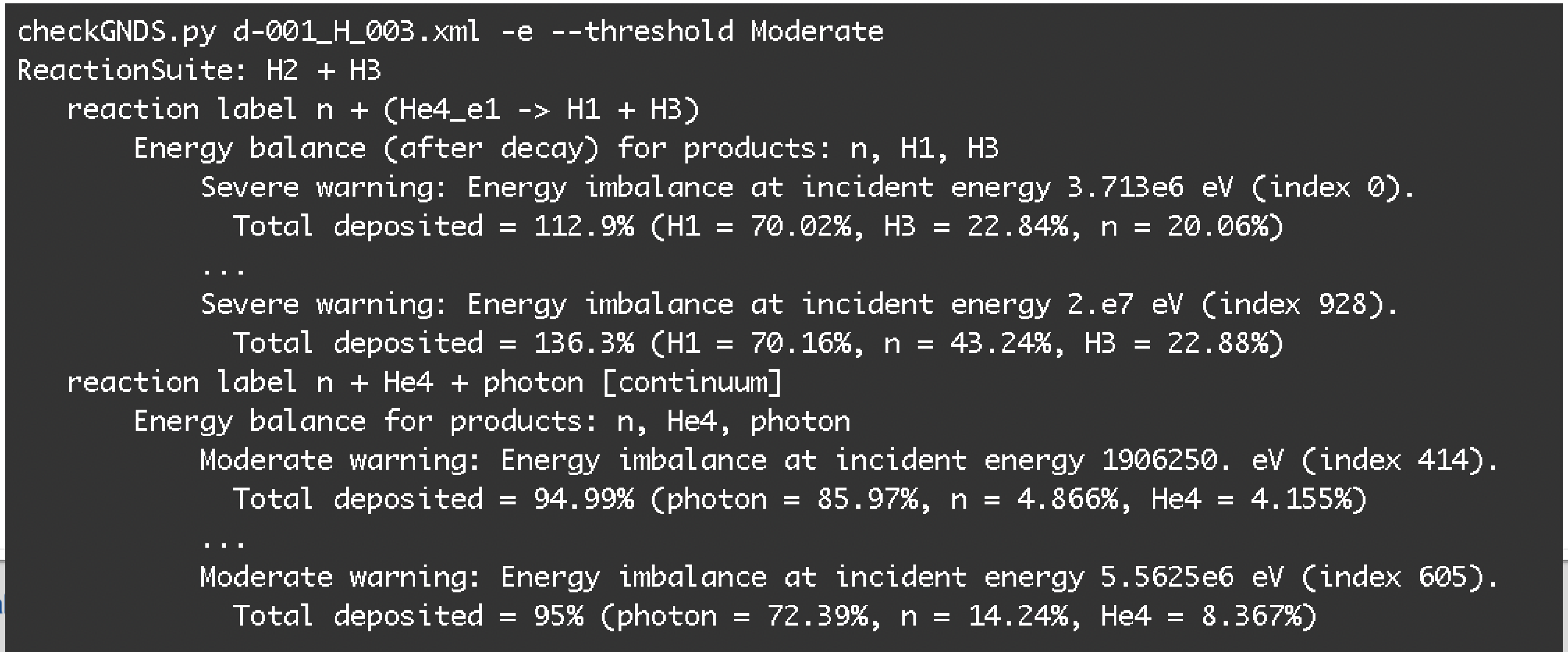
```
<computerCode
        name="..."
        version="...">
    <executionArguments>...</executionArguments>
    <codeRepo>...</codeRepo>
    <note>...</note>
    <inputDecks>...</inputDecks>
    <outputDecks>...</outputDecks></computerCode>
```

```
<exforDataSet
        subentry="..."
        retrievalDate="...">
  <covarianceScript>...</covarianceScript>
  <correctionScript>...</correctionScript>
  <note>...</note></exforDataSet>
```

Q/A

Format checking courtesy of XML schema
Physics checking implemented in FUDGE tool *checkGNDS.py*

- xmllint --noout --schema gnds.xsd *.gnds.xml
- checkGNDS.py produces warnings + severity level ranging from pedantic to fatal
  - checkGNDS.py supports filtering by severity and/or by warning types.
  - Default warning threshold = Moderate
- checkGNDS was used extensively to test ENDF-VIII.1:



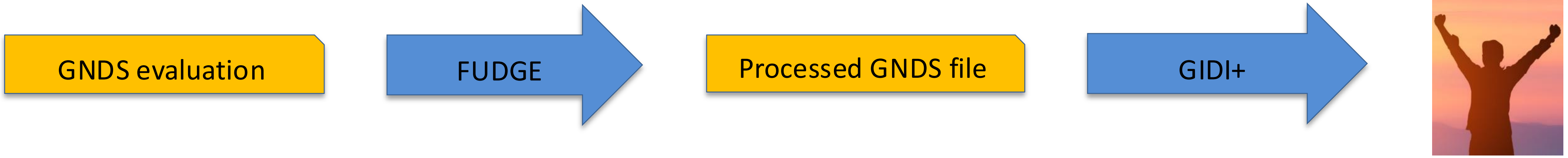
```
checkGNDS.py d-001_H_003.xml -e --threshold Moderate
ReactionSuite: H2 + H3
   reaction label n + (He4_e1 -> H1 + H3)
       Energy balance (after decay) for products: n, H1, H3
           Severe warning: Energy imbalance at incident energy 3.713e6 eV (index 0).
             Total deposited = 112.9% (H1 = 70.02%, H3 = 22.84%, n = 20.06%)
           ...
           Severe warning: Energy imbalance at incident energy 2.e7 eV (index 928).
             Total deposited = 136.3% (H1 = 70.16%, n = 43.24%, H3 = 22.88%)
   reaction label n + He4 + photon [continuum]
       Energy balance for products: n, He4, photon
           Moderate warning: Energy imbalance at incident energy 1906250. eV (index 414).
             Total deposited = 94.99% (photon = 85.97%, n = 4.866%, He4 = 4.155%)
           ...
           Moderate warning: Energy imbalance at incident energy 5.5625e6 eV (index 605).
             Total deposited = 95% (photon = 72.39%, n = 14.24%, He4 = 8.367%)
```

FUDGE supports processing GNDS files for Monte Carlo and deterministic transport

- processProtare.py: main driver in FUDGE for generating processed data

```
processProtare.py evaluation.xml processed.xml @options.input
```

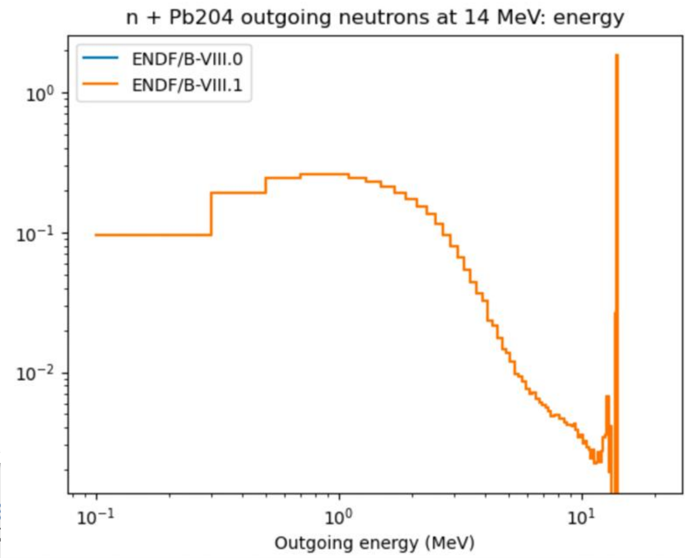
```
cat options.input
--energyUnit eV
--temperatureUnit K
-t 293.6
-t 300
-mc –mg –up
```

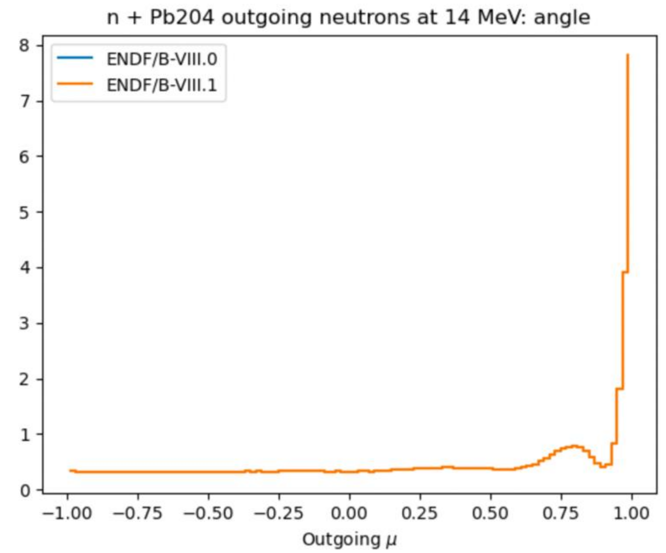


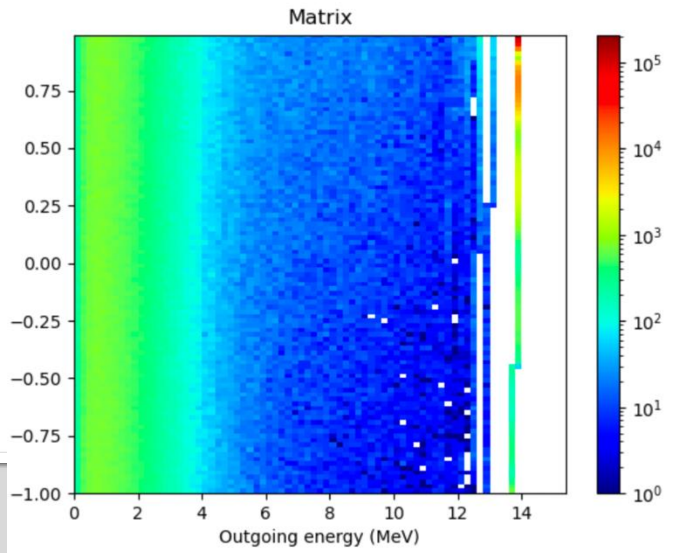
It's not just FUDGE: several other institutions now support processing GNDS!

QA tools for checking processed data are included with GIDIplus

- *depositionEnergy*: similar to KERMA but with the option to include photon / charged particle average energy as well as neutrons
- *readAllProtares*: verify that GIDIplus can load (and optionally sample) from processed files. Useful last check before releasing new libraries to users!
- *broomstick*: simple utility for sampling outgoing product energy and angle, helpful for diagnosing issues and comparing evaluations



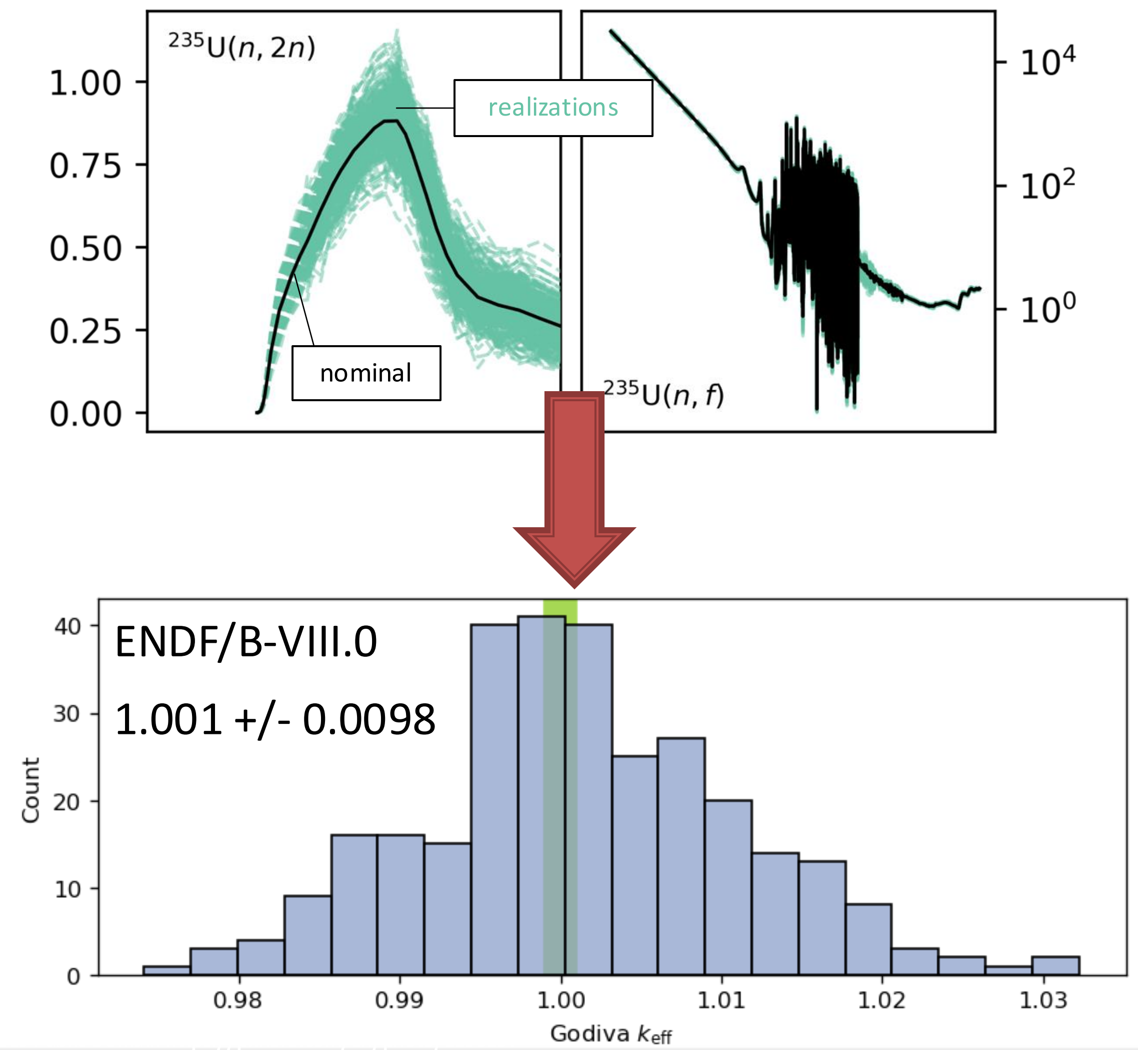




Versioning / user feedback:

- Based on LLNL experience, git works fine for tracking changes in GNDS evaluations. A few caveats:
  - Best to remove any derived data. We're using *cullProcessedData.py* (included with FUDGE) before each commit. That has the added advantage of standardizing the output to minimize unnecessary differences.
  - Tracking full files works fine, but we've also played with splitting evaluations into smaller chunks, e.g. one file for each cross section and one for each reaction product. That has advantages and disadvantages…
  - Continuous integration is still TBD, although the NNDC's ADVANCE is a nice proof-of-principle

EMU is our toolkit for working with nuclear data covariances.



- Suite for generating realizations of GNDS formatted nuclear data
  - Built on top of FUDGE for reading and writing GNDS
    - Library agnostic, only need a translator to/from GNDS
    - Translator must also **correctly** translate covariance data.
- Manages the workflow for multi-material UQ studies. Single Interface for:
  1. Generation of raw nuclear data realization
  2. Processing of realizations into heated and grouped data
  3. Feeding in multi-material variations into application codes
- Python scripting framework to make interaction with sampled libraries extremely simple.

LLNL has made significant progress with migrating our nuclear data workflow to GNDS. Still lots to do!

- Format translators will remain important in the near term BUT we're already starting to use GNDS features that have no ENDF-6 equivalent (especially for documentation).
- We've built up a strong foundation of tools for visualizing, checking, processing and using GNDS-formatted data.
- Big focus now: work with theory / evaluation team on better tools for directly generating GNDS evaluations including covariances.
- Trainings are available for users interested in diving more into GNDS, FUDGE and GIDI+!
  - https://www.oecd-nea.org/jcms/pl_98517/fudge/-mc-gidi/gnds-introductory-course

Lawrence Livermore National Laboratory

Dissemination: beyond GIDI+, FUDGE supports exporting processed data to several other formats

- toACE.py: export processed Monte Carlo data to an ACE file
  - Currently supports incident and outgoing neutron data, TNSL, probability tables
  - Related script toACE_multiTemperature.py for generating ACE files at various temperatures

- Legacy LLNL formats: MCF for Monte Carlo codes, NDF for SN

- Recent addition: exporting some processed data directly to COG library format