DEVELOPMENT OF AN AUTOMATED TESTING SYSTEM FOR VERIFICATION AND VALIDATION OF NUCLEAR DATA

By

BRIAN SCOTT TRIPLETT

A THESIS PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

UNIVERSITY OF FLORIDA

2008

ACKNOWLEDGMENTS

I would like to acknowledge my advisor, Dr. Samim Anghaie, for his mentoring and coordination of my graduate career to date.

I would also like to acknowledge Dr. Morgan White of Los Alamos National Lab for his ideas and insights into the nuclear data world. I would also like to thank the other members of the nuclear data team at Los Alamos including Robert Little, Angela Herring, David Pimental and Doug Coombs.

I was not the only contributor to this computer program. Many lines of code were also written by Christopher Sommer, John Hopkins, Kristen Triplett, Angela Herring, David Pimental, Morgan White and Doug Coombs.

Finally, I would like to thank my wife, Kristen, not only for her direct contributions in the form of computer code during both summers at Los Alamos but also for the support and love she provided during my entire graduate work.

I gratefully acknowledge the Los Alamos National Laboratory, in particular the Nuclear Data Team within X-1-NAD, for funding much of the work presented herein.

The research at the University of Florida was performed under appointment of the Office of Civilian Radioactive Waste Management Fellowship Program administered by Oak Ridge Institute for Science and Education under a contract between the U.S. Department of Energy and the Oak Ridge Associated Universities.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

QA	Quality Assurance
NDVV	Nuclear Data Verification and Validation
V&V	Verification and Validation
LANL	Los Alamos National Lab
GUI	Graphical User Interface
ICSBEP	International Criticality Benchmark Experiment Project
CSEWG	Cross-Section Evaluation Working Group
XML	Extensible Markup Language
BIRT	Business Intelligence and Reporting Tools
OO	Object Oriented

Abstract of Thesis Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Master of Science

DEVELOPMENT OF AN AUTOMATED TESTING SYSTEM FOR VERIFICATION AND VALIDATION OF NUCLEAR DATA

By

Brian Scott Triplett

August 2008

Chair: Samim Anghaie
Major: Nuclear Engineering Sciences

Verification and validation of nuclear data is critical to the accuracy of both stochastic and deterministic particle transport codes. In order to effectively test a set of nuclear data, the data must be applied to a wide variety of transport problems. Performing this task in a timely, efficient manner is tedious. The nuclear data team at Los Alamos National Laboratory in collaboration with the University of Florida is developing a methodology to automate the process of nuclear data verification and validation (V&V). This automated V&V process tests a number of data libraries using well defined benchmark experiments, such as those in the International Criticality Safety Benchmark Experiment Project (ICSBEP). The automation also serves to reduce errors and increase efficiency. The process is implemented through an integrated set of Python scripts. Material and geometry data are read from an existing medium or given directly by the user to generate a benchmark experiment template file. The user specifies the choice of benchmark templates, codes, and libraries to form a V&V project. The Python scripts generate input decks for multiple transport codes from the templates, run and monitor individual jobs, and parse the relevant output automatically. The output can then be used to generate reports directly or can be stored into a database for later analysis. This methodology eases the burden on the user by reducing the amount of time and effort required for obtaining and compiling calculation

results. The resource savings by using this automated methodology could potentially be an enabling technology for more sophisticated data studies, such as nuclear data uncertainty quantification. Once deployed, this tool will allow the nuclear data community to more thoroughly test data libraries leading to higher fidelity data in the future.

CHAPTER 1
NUCLEAR DATA VERIFICATION AND VALIDATION OVERVIEW

Introduction

Nuclear and radiological engineering applications require a large degree of quality assurance (QA) in order to guarantee safety and reliability. These applications may include nuclear reactors, radiation diagnosis/treatment devices, and radiation detector systems. The complexity of nuclear physics and the underlying nuclear interaction data can make the QA process difficult. The interaction data for neutrons alone encompass thousands of isotopes over many decades of energy and the nuclear physics involved makes generalizing particle behavior extremely difficult. For this reason, the nuclear interaction data are often stored in tabular form, typically in formatted files of particle energy versus the nuclear cross section. Several organizations have produced such evaluated data libraries, (e.g., ENDF/B in the United States,[1] JEFF in Europe,[2] JENDL in Japan).[3] These evaluated data libraries must be further processed into an application specific library, (e.g., into ACE format for use in MCNP[4] or NDI format for PARTISN).[5]

In the course of producing an application library, it is desirable to perform detailed QA on the data itself. This is done to verify that data were translated and processed correctly and to validate that the library performs within expectations for a problem set of interest. Given the immense phase space that nuclear interaction data encompass and the limited models available, this can be an exceptionally burdensome task. Often the problem set against which the library is validated is rather small due to time and/or financial constraints. It is for these reasons that the nuclear data team at Los Alamos National Laboratory, in collaboration with University of Florida, is developing a robust testing harness that will automate many of the validation tasks associated with nuclear data libraries.

The goal of this project is to create a generalized modular testing harness that is publicly available to test both nuclear transport applications and their underlying data libraries. One of the initial aims of this project is the creation of a suite of benchmark problems for testing data using both MCNP (a continuous energy Monte Carlo code)[4] and PARTISN (a deterministic S_N code).[5] A key feature of the test harness is a modular design that will allow the application to other codes in the future. Additionally, such a system may enable other types of studies in the future, (e.g., integration in regression testing for automated code V&V and automation of uncertainty quantification).

Motivation

The inspiration for this project stems from a proprietary data testing system developed by Steven van der Marck at the NRG laboratory in the Netherlands used for testing the ENDF/B-VII.0 library during its development.[6] Similar to van der Marck's work, the nuclear data verification and validation (NDVV) project aims to increase efficiency and accuracy when testing nuclear data.

To demonstrate the necessity of an NDVV tool, we can consider the latest release of the US data library, ENDF/B-VII.0. This library has undergone significant benchmarking to date and is expected to be scrutinized further as it continues to be used for more diverse applications.[6,7] This latest release contains 14 sublibraries. The neutron interaction sublibrary contains data for some 393 nuclides, each for typically dozens of reaction types and spanning many orders of magnitude in particle energy. Performing V&V for this sublibrary alone can be an immense task without taking into account this library's inter-relationship with the other sublibraries such as the thermal scattering sublibrary, the photonuclear sublibrary, the radioactive decay library and so on. It would take many man-hours to perform quality assurance on these data libraries with little or no automation.

Another significant motivation for the NDVV project is the testing of new/improved transport methods. Upon the development or deployment of a nuclear transport application, it is common to perform some type of regression test in which the results of many well-defined problems are compared to benchmarks. This is necessary to ensure that the application is performing as intended and no developer errors (bugs) or platform specific issues are present. It is the burden of the application developers to implement their own regression testing harness (if any is created at all). However, if the NDVV system is used, it is possible that the developers will need only to develop a way to "plug-in" their application to the NDVV system in order to perform regression testing. If a system has already been incorporated into the NDVV system, such as MCNP or PARTISN, the regression testing using the NDVV system is already present and can be used by developers.

The growing need for nuclear data uncertainty quantification is another motivation for this development. Tabular data libraries, such as ENDF/B, are usually the conglomeration of both experiment and nuclear physics models, both of which have a degree of uncertainty. Many in the nuclear data community desire to quantify how nuclear data uncertainty affects their complicated transport application and how that uncertainty will propagate into further calculations.

A recent study at LANL utilized a Monte Carlo method to perturb the fission cross section of ^{239}Pu in order to evaluate the uncertainties in the fission cross section.[12] This type of undertaking involves a number of similar calculations to determine how slight perturbations in a parameter of interest (e.g., a cross section) affect another parameter (e.g., k_{eff}). In this case, a one dimensional PARTISN calculations were performed. If driven by an automated process such as the NDVV system, the resulting time savings could potentially allow broader and more detailed types of uncertainty studies.

Current Benchmarking Efforts

Benchmarking nuclear data libraries is not a novel concept. Upon every release of a new data library, many institutions set out to perform some degree of V&V on the new data set. Similarly whenever an institution or individual releases a new calculation methodology (typically in the form of a computer program), the burden is on the developers to validate their program's results versus experiments or at least verify that their results are consistent with similar calculation methods. As already mentioned, the ENDF/B-VII.0 release was tested extensively by S.C. van der Marck upon its release. Similar studies have also been performed such as the shielding validation performed on the European data library release, JEFF 3.1 by W. Haeck.[8] Inter-library comparisons are also performed regularly (see Reference 9 for one such example). Many transport applications will pick-and-choose different isotopic evaluations from different libraries in order to obtain the best data possible. However, these studies tend to be focused on a single goal and are the result of many hours of dedication by a number of individuals. The intent of the NDVV automation system is that these studies will become more commonplace.

Scope of this Thesis

It is important to point out that this work is not focused on the nuclear data analysis itself. Those topics can be found in the many volumes of journal articles and reports published regularly. This paper will put forth some example calculations performed using the NDVV system which involve MCNP and a number of different libraries. It is not within the scope of this paper to analyze the accuracy of these codes and libraries but merely to present a typical application of the NDVV system.

CHAPTER 2
DEVELOPMENT AND IMPLEMENTATION OF THE NDVV SYSTEM

History

The current NDVV system is the result of over two years of development by a number of students and staff at Los Alamos National Lab (LANL) and the University of Florida. The automated verification and validation of nuclear data concept was pioneered by Dr. Morgan White at LANL. As a member of the LANL nuclear data team and the Cross-Section Evaluation Working Group (CSEWG), he believed that this type of automated testing harness would be a valuable asset to the nuclear data community.

The software development was first started in the summer of 2006 by a team of four students at LANL with backgrounds in nuclear engineering and computer software. The initial focus was the generation of problem inputs and conversion to and from generalized problem descriptions to an application specific format. During the summer of 2007, the application control/execution and output processing was further developed. The development continued into early 2008 to complete the prototype system. In its current state the NDVV process represents a complete prototype and can be used for a fairly wide array of data studies.

Overview of the NDVV System

The NDVV testing harness automates the creation of input, handles transport code execution, post-processes results, and generates standardized reports. All of these tasks are accomplished using the Python programming language. Python was chosen because its object-oriented capabilities make for robust, flexible coding. Python also features built-in capabilities to facilitate using a number of key file formats selected for the NDVV system. Python also offers tools for interacting with important computer resources such as databases and the system shell.

Python was also chosen because of its portability. A key NDVV goal is to maximize system independence whenever possible. The ability to run on many different types of systems maximizes the computing resources available, enables more transport codes to be used, and more easily facilitates new uses of the system. Python is an "interpreted" language and is generally system independent.[11] Provided a user has installed the appropriate version of the Python virtual environment most Python applications will run on any type of system (e.g., Windows, Macintosh, Linux), without need for modification or recompilation. In the NDVV system's current implementation, Python is only required on one master system (typically the user's local computer). Any computation-only nodes simply require command line interfaces to the transport codes. The reporting tools are also only required on a single system. Minimizing the resource requirements maximizes the systems flexibility while reducing the installation burden.

The first priority of the NDVV developers was to provide a full system demonstration before widening the scope of the NDVV system. The NDVV prototype system currently supports only the transport codes MCNP and one-dimensional (i.e., spherical geometry) PARTISN. These two codes are different in their calculation approach and in the type of data they require. MCNP is a stochastic, Monte Carlo based code that utilizes continuous energy cross section data. PARTISN is a deterministic code that uses a number of discrete energy groups for its cross section data. It is the view of the developers that these two codes provided an appropriate demonstration the flexibility of the NDVV system for the initial prototype.

At this current point, the development for deterministic codes, such as PARTISN, requires further study; specifically in regard to automating the discretization of both space and energy without the introduction of error or the wasting of computer resources. There is a delicate, sometimes artful, balance required when determining the balance between accuracy and speed

when performing deterministic calculations. Therefore, automating these types of decisions can be challenging. Some studies have been performed in the development of the NDVV system but have not progressed past the scoping phase yet. Due to this current limitation, PARTISN only functions in 1D geometries. Even a severely over-meshed 1D problem does not overwhelm computer resources on most modern systems. This allows the system to err on the side of finer spatial meshing for 1D cases without the worry of wasting computer resources.

Data and results are stored in many different forms throughout the NDVV process including: Extensible Markup Language (XML), relational databases, and formatted text. The templates used to store benchmark descriptions are formatted in XML because of the self-documenting nature and the ease of Python to parse XML. The input and configuration files to the NDVV system are formatted in XML for the same reasons. Code input and output formats are dictated by the transport codes themselves but usually are given as semi-structured ASCII text or binary files. Creation of these inputs and parsing of the subsequent output must be programmed into the system on a code-by-code basis. Any addition of a new transport code will require development of a new code module.

Analyzing the simulation outputs is possibly the most daunting and time consuming task in the V&V process. If any automation exists in traditional methods, it is typically employed during this stage. The transport code outputs are typically text or binary files that have evolved over time and rarely present the information in all the ways desired. Often users must find a means to conglomerate the results into another desirable medium, such as a spreadsheet. The NDVV automated output parsing offers faster access to the parameters of interest (e.g., k_{eff}, fluxes, run times, memory usage, etc). Once stored in the NDVV system they can then be

recombined and presented in a number of formats. Possible formats include formatted text, tables, plots and relational databases.

One of the most important means of long-term storage is a relational database. By combining results from multiple studies into an integrated database, the data may be reused and mined in multiple ways. Using a relational database such as MySQL or PostgreSQL offers some of the best capabilities of modern data storage techniques and tools. When properly implemented, a database can store large amounts of data while eliminating redundancies and allowing versatile searching and presentation of data.

The NDVV process has the ability to connect to a database and insert results from the NDVV runs. Currently the NDVV system includes an interface to store results in a PostgreSQL database. As the project expands it is expected that more types of databases will be included.

Another benefit of relational databases is the third party tools that support them. These tools are not tied to any specific types of data and are available to any users of the database. Once data are in a database, an existing tool can be used to search and present data without the need for specific development by the NDVV developers or the user.

The following discussion provides greater detail regarding the three major steps in the NDVV process: Input Generation, Code Execution, and Output Processing.

Input Generation

In order to validate a code/library one must have some a-priori results for comparison. This can be an analytical model or an actual experiment. Analytical models tend to be limited in scope for typical nuclear physics applications. Therefore, for most validation problems, an experiment forms the basis by which accuracy is determined. For a code-to-code or library-to-library comparison an experimental or analytical basis is not needed but the accuracy can only be inferred based on the agreement between results.

A common example of problems of interest for data benchmarking are those experiments included in the International Criticality Safety Benchmark Experiment Project (ICSBEP) Handbook[12] and those in the ENDF-202 report by the Cross Section Evaluation Working Group (CSEWG).[13] These experiments are well-defined criticality and shielding experiments that cover a wide range of scenarios and offer an excellent base with which to test the NDVV system. For this reason, the examples selected for this paper are from the ICSBEP Handbook.

Using the benchmark experimental data as a starting point, the user will generate a template that describes each problem of interest for data benchmarking. These template files store the necessary information needed to create transport code inputs for the various codes used by the NDVV system.

The templates are formatted in XML. XML is a portable, hierarchical markup language that is used to store data. It is commonly found in many web-based applications because of its system independent nature. XML also has strong software support from the commercial sector since it aides many businesses in making their data portable.

In the current system, it is possible to "initialize" a template file from a pre-existing MCNP input file. One reason for this is because many benchmark experiments in the ICSBEP have an MCNP input attached as an appendix. Although the editors of the ICSBEP do not guarantee the accuracy of those inputs, they can often serve as a good starting point. This feature has been added as a convenience so that not every template file must be generated from scratch.

A full template file is demonstrated in Figure 2-1. This template describes the Jezebel critical assembly used at LANL for nuclear physics research.

The template generation phase typically occurs once for each problem. Once a template has been created and its quality been assured, it will typically be stored in a secure location

and/or placed under revision control. If a repository of templates is already available, then the user only needs to specify the location of those templates in the input file prior to the NDVV run.

Once a template file is complete, the NDVV conversion scripts read the template and generate inputs for the transport codes of interest. For every transport code a geometry and material specification is required as a minimum. There is also typically code specific information required, (e.g., KCODE cards in MCNP). All of this information is currently stored in code specific sections of the XML template file that describes the experiment. A goal of this system is to specify as much of this information as possible in a code independent manner to eliminate redundancies and ensure input consistency.

There may also be computer dependent information required in the transport code inputs (file locations, specific system directives, etc). This is not stored in the problem template but given at run time by the user (either directly or stored in machine description files). This is to ensure that a suite of experimental benchmark templates will remain portable and can be easily transmitted among the nuclear data community.

When the inputs are first generated, any computer specific input requirements (e.g., file locations) are filled with placeholders. These inputs are labeled as "code input template" in the Figure 2-2. When the run phase begins, these placeholders are filled in just before the input is sent to a specific machine. This ensures that the input has been properly customized for the given computer. Figure 2-2 describes this process and how transport code inputs are generated by the NDVV system.

This input generation approach yields a number of benefits in regard to quality assurance. The first is the reduction of input files to be reviewed. The XML templates are tied to the problem of interest and not to the code/library to be used. Therefore, this approach only requires

one file per benchmark experiment instead of one for each code/library combination. This reduction of files simplifies the benchmark data review process.

This input model also increases the consistency by which inputs are generated. Often user preferences can translate into small differences between results. This typically manifests itself in a few common ways: specifying of a certain variable versus using the code default, differing floating point precision, breaking up isotopes versus using natural evaluations. These user decisions do not necessarily lead to inaccurate results but can sometimes lead to confusion when results differ slightly. The input files generated by the NDVV process are linked to the benchmark templates and therefore maintain consistency with the original template from run to run.

Arguably the most probable error during problem setup is a transcription error. An user-introduced typographical error in the input file can change the material or geometry specification and lead to erroneous results. Often the typographical errors have a small effect and are not immediately noticed until detailed analysis is performed. Given that the initial template has been fully vetted, automated generation of input decks are immune to typographical errors (or at least errors are propagated consistently).

Code Execution

When the user is ready to execute a V&V calculation, a NDVV queuing system controls the execution of the transport codes. The user specifies the computers to use and the transport codes and data libraries available on each computer. The NDVV control process determines if computers are available and sends the input file(s) and execution instructions as appropriate.

The specification of the V&V job to run and the resources available are given in a NDVV user input file. This file tells the NDVV process a number of specific parameters required to control and execute the V&V calculations. Like the benchmark templates, this input file is in

XML format. Although XML offers a moderate amount of readability to the user, it is the intent of the developers to eventually drive the user input setup process with a graphical user interface (GUI). This not only facilitates quick straightforward input creation but also can be used as a means to control/limit the choices a user can make and reduce errors. For example, objects such as drop-down menus and radio buttons can limit user's choices to a few safe options; whereas in a XML file the user is free to type anything he or she wants leading to potential typographical errors.

The first requirement in the user input file is the choice of benchmark templates and their location. It is the developers' expectation that a template repository or collection of templates will be located on a system available to the user (either locally or remotely). In the input file, the user will specify their desired subset of templates (or the entire repository if applicable) and the templates' location.

The user must also furnish his or her choice of transport codes. This may be one or more codes to be included in the run. As previously noted, the current implementation of the NDVV system only includes support for MCNP and PARTISN (1D) with more code modules to be added in the future. The location of the code executables and any specific command line requirements must also be included in the user input file.

For each code specified, one or more libraries must be specified. The most pertinent library information required is the location of these libraries on the computer systems to be used.

Finally, the computer systems themselves may require some additional information from the user. This typically includes connection parameters (if it is a remote machine), queuing limits, and system type. The underlying assumption in this user input preparation methodology

is that the user knows his or her system(s) best and it the most qualified to determine how and where system resources should be used.

Once all the necessary information has been input, the NDVV control process handles the sending of NDVV jobs to each of the available computers via its queuing system. It is possible to have different codes available on different computers. For example the user may specify two computers and one is only able to run code X and has library A but the other computer has code X and Y and only library B. The NDVV control process will handle the distribution of tasks based on the capability of each computer to perform a calculation.

A NDVV job consists of a unique combination of template, code, and library. NDVV runs can consist of few or many jobs. Once the control process determines that a given job is ready to run on a specific computer, it sends the required input(s) and a set of system commands to that computer. These commands are the instructions to execute the code on that computer and are contained in a system-dependent shell script. A command is issued to run the shell script and the connection is closed with that computer. The NDVV process periodically reconnects with each computer and assesses the status of each job. When a job is completed, any relevant outputs are copied back to the user's computer and placed in the NDVV project directory. The user can provide the specifics regarding the maximum number of jobs that can run on each computer and the frequency at which the jobs' status is checked in the NDVV user input file.

It is important to note that all of the NDVV computations can be run on a single local computer or make use of simple or complex cluster computing. In striving for maximizing efficiency while maintaining as much system independence as possible, the simple queuing presented here provides minimal facilities that will work anywhere. If an alternate queuing system is available and desired, job control can be ceded to that system simply by submitting all

jobs to that system. Examples of these alternative queuing systems include the LSF and MOAB systems at various national labs. It is anticipated that for large verification and validation (V&V) jobs more powerful computer resources will be used in this manner. Figure 2-3 gives an overview of the execution phase of the NDVV system.

Output Processing

Upon completion of each code execution, the NDVV control system collects the relevant transport code output files back to the master computer. After the output files are collected, post-processing begins. This is done on the master host because of the availability of more advanced tools, (e.g., Python) which are needed to perform the post processing. The files are parsed for pertinent information (k_{eff}, flux distribution, reaction or leakage rates, etc) via the NDVV post-processing system. The data may be stored in a database, written to a formatted file, or used to generate plots and reports.

The NDVV scripts search the outputs from the various transport codes and gather various important parameters. The data from each code are stored in a custom NDVV data structure so that it can be accessed in a standardized manner. Once these results have been gathered and stored in the NDVV structure, the user may choose any of the supported NDVV formats for displaying/storing results. The NDVV process always produces some limited textual summary information from each run but also contains more elegant post-processing options with more to come as development continues.

Plotting with gnuplot

The first post-processing option is the gnuplot option. This option searches through the NDVV result data and creates the files necessary to plot those results using the open source plotting software, gnuplot.[15] The gnuplot program uses these files as the basis to create postscript plots based of the NDVV results. This saves the user the time and effort required to

create plots of their results. The user is also free to modify the gnuplot script files in order to customize the plot appearance if the defaults are insufficient or undesirable. Figure 2-4 give an overview of the output processing phase of the NDVV system.

Database storage

In addition to plotting results with gnuplot, NDVV results can be stored into a PostgreSQL database for later querying and retrieval by the user. The user is required to have access to a PostgreSQL database with the proper table structure. Once the tables are set up, the automated NDVV system inserts results as rows in the appropriate tables.

The algorithm that inserts NDVV results is based on a specific database design. The design of this database is based on the selection of variables the NDVV developers felt would be important for V&V. This design will continue to evolve and improve with the NDVV system.

The NDVV table layout in the NDVV database follows the principles of database normalization. By following these guidelines, the database will be free of redundancies and less prone to data anomalies. An overview of this NDVV database design, including table names, keys, and columns, is shown in Figure 2-6.

The NDVV database design, like the rest of the current NDVV system, is a working prototype that will continue to evolve with time. As more input and requirements are included the database design will mature and become more extensive. Since the SQL directives generated by the NDVV code are intimately tied to the design of the database, it is expected that a database creation tool will also be packaged with the NDVV system. This will ensure the compatibility of the NDVV database storage mechanisms and the database structure. This tool will allow users to create the required tables for storing their own NDVV results. Currently, a simple SQL script exists for the initial creation of the database tables. This method is the one presently used by the NDVV developers.

Once data are in a database, a wealth of tools exists to query and present data from the database. On such tool is the Business Intelligence and Reporting Tools (BIRT) program. BIRT is an open source reporting system that can generate standardized reports based on a database source.[14] BIRT uses a "What-You-See-Is-What-You-Get" interface so that users know exactly how their reports will appear. BIRT is able to be deployed on a web server so that anyone with internet access can view the reports created. BIRT also allows reports to be exported to common formats such as DOC and PDF. BIRT also includes built in methods for creating plots and tables and is able to perform calculations using JavaScript.

An example BIRT report is shown in Figure 2-6. This report displays the benchmark k_{eff} along with the k_{eff} results from MCNP and PARTISN. The calculated/experimental (C/E) values are performed internally by BIRT.

The results given in this example are from previous calculations performed at LANL and were not obtained via an NDVV run. However, once full PARTISN support is implemented in the NDVV system, they could easily have been the result of the NDVV process.

It is important to point out that BIRT is not part of the NDVV system. It is merely an example of how a third-party tool can be used to present results once they are contained in a relational database. Users are free to use the display mechanism of their choice (or none at all) when presenting their own results. The important point from this demonstration is the benefits of using a relational database. Once data are stored in a database, such as PostgreSQL, an entire world of software tools exist to search and display those results as the user requires.

```
<?xml version="1.0"?>
<template>
  <benchmark>
    <problem_name>                  PU-MET-FAST-001 </problem_name>
    <description>
      Jezebel
      Bare sphere of Pu-239 with 4.5% Pu-240
      See also CSEWG-F1
    </description>
    <source>                                  ICSBEP </source>
    <version>                                09/2007 </version>
    <date>                                2007-09-01 </date>
    <keff>                                     1.000 </keff>
    <uncertainty>                              0.002 </uncertainty>
  </benchmark>

  <title>
    Jezebel - Bare sphere of Pu-239 with 4.5% Pu-240; CSEWG-F1; PU-MET-FAST-001
  </title>

  <cell_cards>
    <cell_definition>
      <cell_name>             Pu Sphere </cell_name>
      <cell_number>                   1 </cell_number>
      <material>                      1 </material>
      <atom_density>         0.04029014 </atom_density>
      <geometry>                     -1 </geometry>
      <importance>                    1 </importance>
    </cell_definition>
    <cell_definition>
      <cell_name>                  Void </cell_name>
      <cell_number>                   2 </cell_number>
      <material>                      0 </material>
      <geometry>                     +1 </geometry>
      <importance>                    0 </importance>
    </cell_definition>
  </cell_cards>

  <surface_cards>
    <surface_definition>
      <surface_name>             Pu Sphere </surface_name>
      <surface_number>                1 </surface_number>
      <surface_type>                 so </surface_type>
      <surface_parameters>         6.38493 </surface_parameters>
    </surface_definition>
  </surface_cards>

  <material_cards>
    <material_definition>
      <material_name>   Plutonium (4.5 wt% Pu-240) </material_name>
      <material_number>          1 </material_number>
      <composition>
        <component>
          <nuclide>     Ga </nuclide>    <atom_fraction>    1.3752e-3 </atom_fraction>
        </component>
        <component>
          <nuclide>  Pu239 </nuclide>    <atom_fraction>    3.7047e-2 </atom_fraction>
        </component>
        <component>
          <nuclide>  Pu240 </nuclide>    <atom_fraction>    1.7512e-3 </atom_fraction>
        </component>
```

Figure 2-1. Example XML template file used to specify benchmark problems. This file describes PU-MET-FAST-001 from the ICSBEP handbook, otherwise known as the Jezebel critical assembly. XML was truncated for brevity.

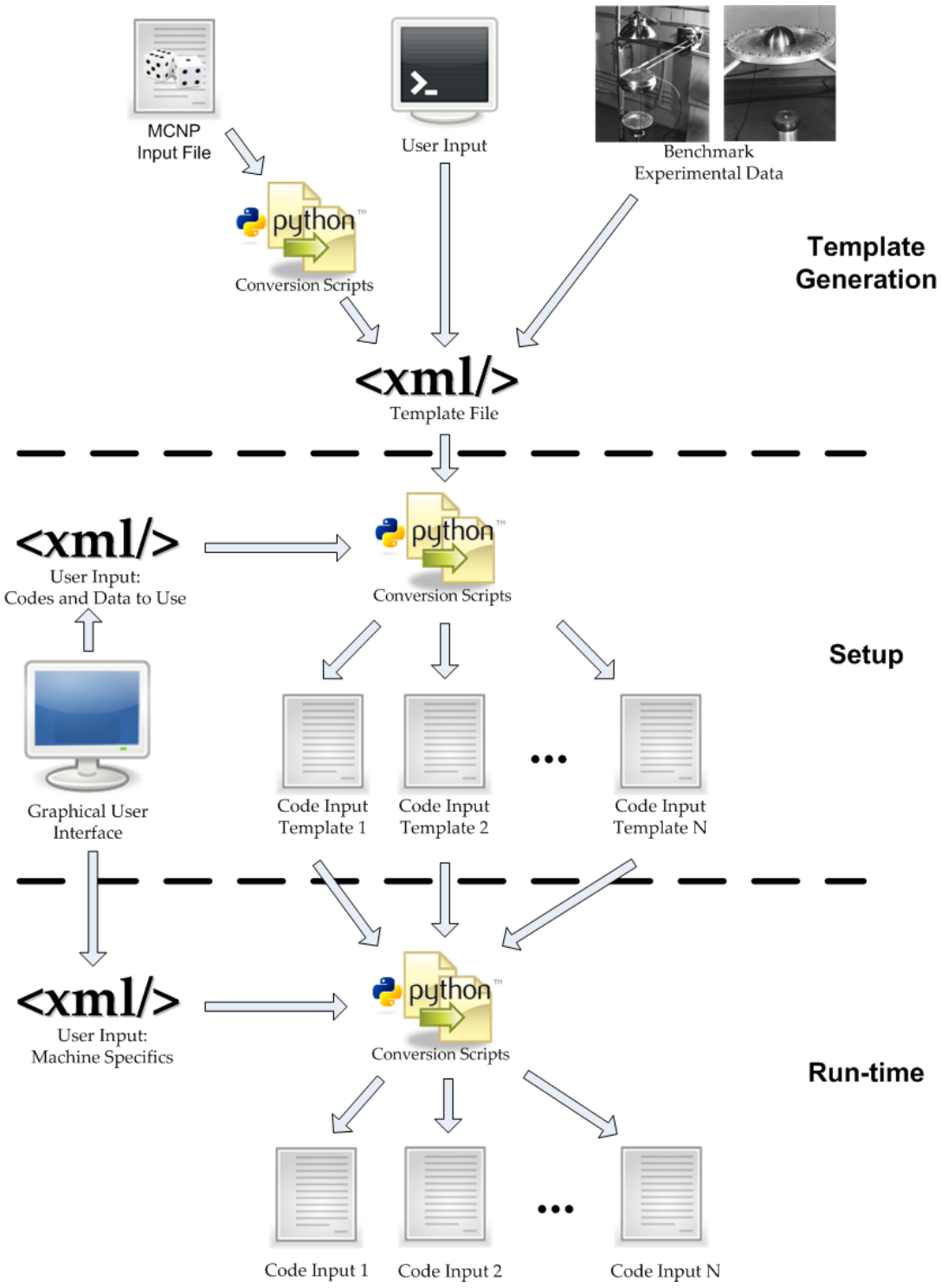

Figure 2-2. Input generation process for the NDVV system. Divided up into template generation phase, setup phase, and run-time phase. The template generation phase typically occurs once, in order to generate a template repository. The setup and run-time phases occur for every NDVV run.

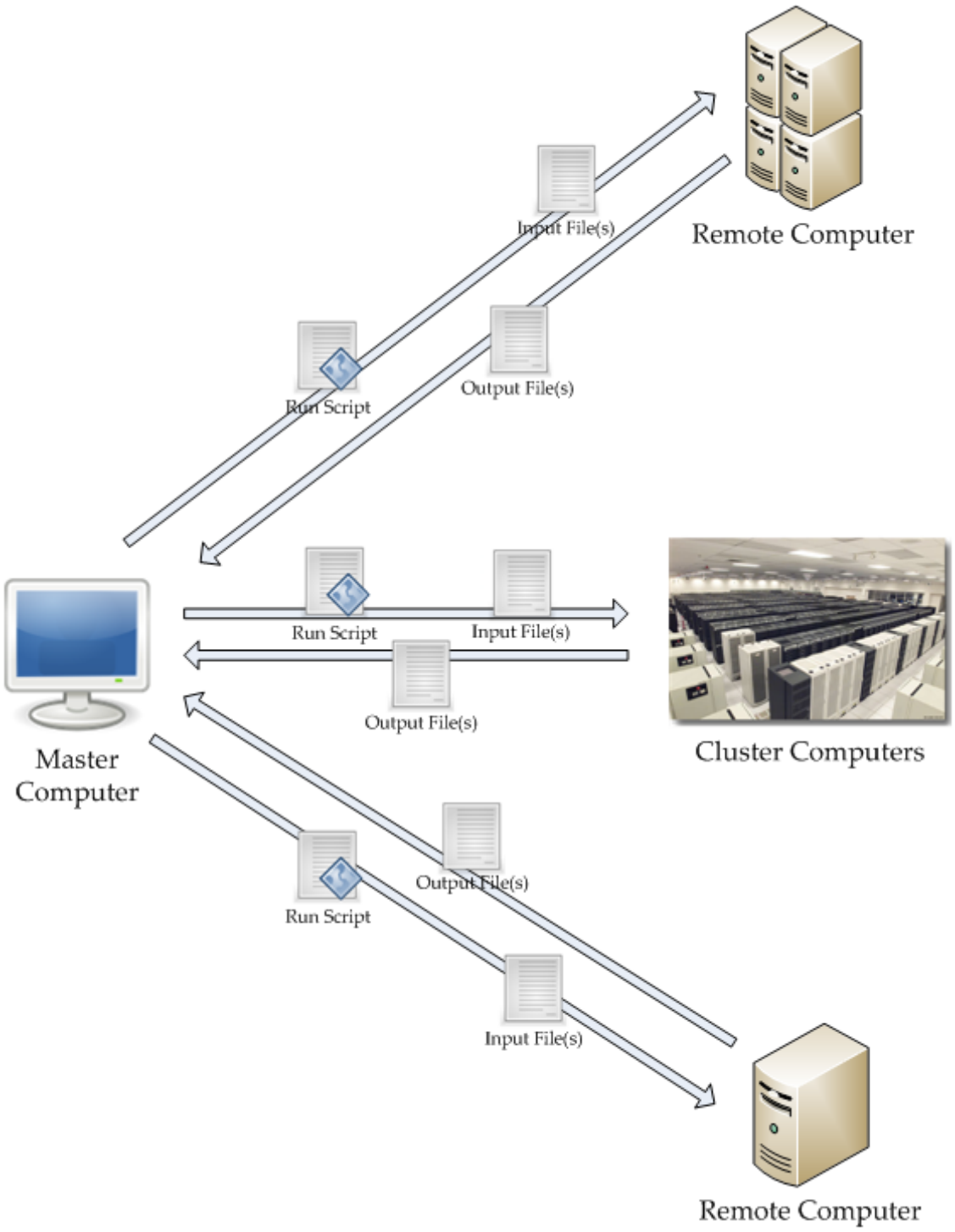

Figure 2-3. Job execution in the NDVV system. A master computer sends input file(s) and scripted instructions to each computer. The results are copied back to the master computer when execution completes. The master computer can also act as a computational node.

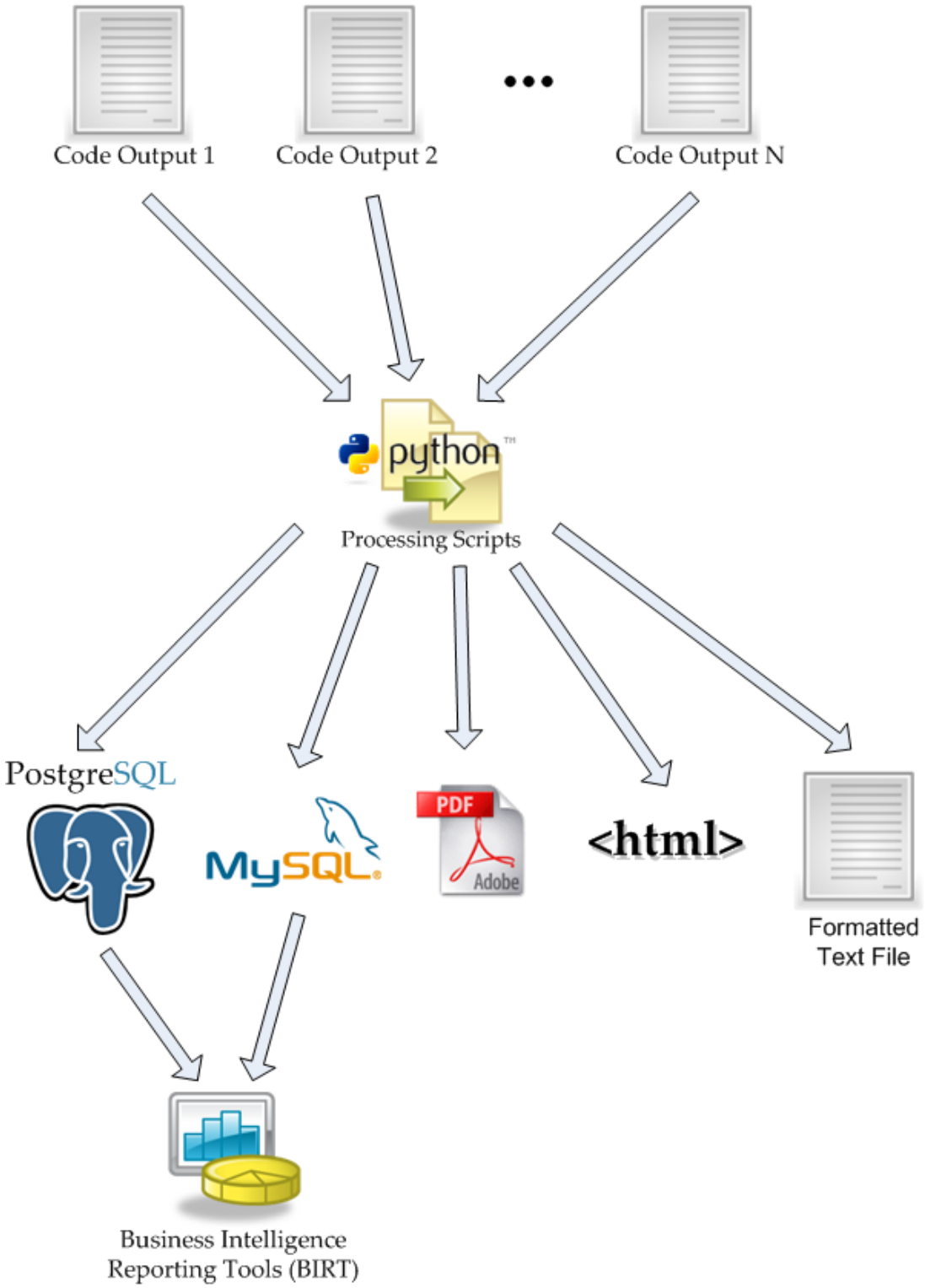

Figure 2-4. Output processing in the NDVV system. Output files are parsed for information and the resulting data are stored in the desired format. Databases such as PostgreSQL and MySQL can be used with third party tools such as BIRT for further data processing/presentation.

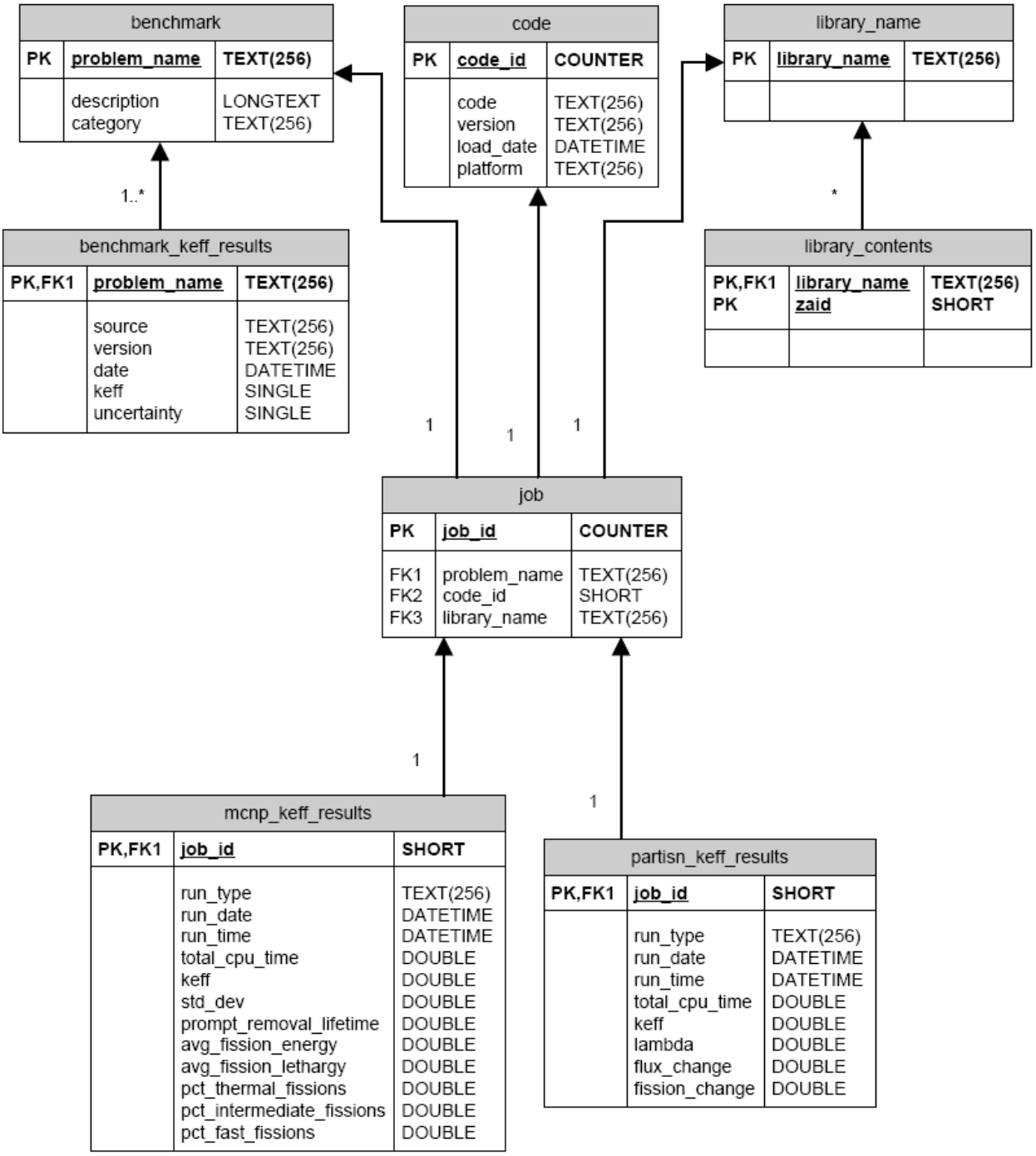

Figure 2-5. NDVV database design layout. Tables are shown with their linking primary and foreign keys. Variable types are specified with PostgreSQL types.

BIRT Report Viewer

Showing page 1 of 1 Go to page:

Aluminum-Reflected Assemblies

ICSBEP Name	Benchmark Description	Benchmark Keff	MCNP t16_2003	MCNP/ Benchmark Keff (C/E)	PARTISN pre7_618	PARTISN/ Benchmark Keff (C/E)
HEU-MET-FAST-022	Duralumin-Reflected HEU Sphere, VNIIEF	1.0000±0.0021	0.9954±0.0006	0.9954±0.0022	1.0033±0.00001	1.0033±0.0021
IEU-MET-FAST-006	Duralumin-Reflected IEU Sphere (36 wt.%), VNIIEF	1.0000±0.0023	0.9931±0.0006	0.9931±0.0024	0.9962±0.00001	0.9962±0.0023
PU-MET-FAST-009	Aluminum-Reflected Pu (94.8) Sphere, Comet Assembly	1.0000±0.0027	1.0042±0.0006	1.0042±0.0028	1.0161±0.00001	1.0161±0.0027

Bare Metal Assemblies

ICSBEP Name	Benchmark Description	Benchmark Keff	MCNP t16_2003	MCNP/ Benchmark Keff (C/E)	PARTISN pre7_618	PARTISN/ Benchmark Keff (C/E)
HEU-MET-FAST-001.a	Godiva, Unreflected Sphere of HEU, Simple Sphere representation	1.0000±0.0010	0.9988±0.0006	0.9988±0.0012	1.0009±0.00001	1.0009±0.0010
HEU-MET-FAST-001.b	Godiva, Unreflected Sphere of HEU, Nested Spherical Shell representation	1.0000±0.0010	0.9993±0.0006	0.9993±0.0011	1.0009±0.00001	1.0009±0.0010
HEU-MET-FAST-008	Bare HEU Sphere, VNIITF, 3D model	0.9989±0.0016	0.9953±0.0006	0.9964±0.0017	0.9915±0.00001	0.9926±0.0016
HEU-MET-FAST-015	Bare HEU Cylinder, VNIITF	0.9996±0.0017	0.9937±0.0006	0.9941±0.0018	0.9896±0.00001	0.9900±0.0017
HEU-MET-FAST-018	Simplified Bare HEU Sphere, VNIIEF	1.0000±0.0016	0.9995±0.0006	0.9995±0.0017	1.0008±0.00001	1.0008±0.0016
IEU-MET-FAST-003	Bare IEU Sphere (36 wt.%), VNIIEF	1.0000±0.0017	1.0032±0.0006	1.0032±0.0018	1.0061±0.00001	1.0061±0.0017
PU-MET-FAST-001	Jezebel-Pu (4.5%), Bare Sphere of Pu-239 with 4.5% Pu-240	1.0000±0.0020	1.0004±0.0006	1.0004±0.0021	0.9989±0.00001	0.9989±0.0020
PU-MET-FAST-002	Jezebel-Pu (20%), Bare Sphere of Pu-239 with 20% Pu-240	1.0000±0.0020	1.0018±0.0006	1.0018±0.0021	0.9993±0.00001	0.9993±0.0020
PU-MET-FAST-022	Simplified Plutonium (98%) Bare Sphere, VNIIEF	1.0000±0.0021	1.0001±0.0006	1.0001±0.0022	0.9978±0.00001	0.9978±0.0021
U233-MET-FAST-001	Jezebel-23, Bare Sphere of U-233	1.0000±0.0010	0.9980±0.0006	0.9980±0.0012	1.0002±0.00001	1.0002±0.0010

Beryllium and Beryllium-Oxide-Reflected Assemblies

ICSBEP Name	Benchmark Description	Benchmark Keff	MCNP t16_2003	MCNP/ Benchmark Keff (C/E)	PARTISN pre7_618	PARTISN/ Benchmark Keff (C/E)
233U-MET-FAST-005.1	0.805" Be-Reflected Sphere of U-233, Planet Assembly	1.0000±0.0030	0.9968±0.0006	0.9968±0.0030	0.9998±0.00001	0.9998±0.0030
233U-MET-FAST-005.2	1.652" Be-Reflected Sphere of U-233, Planet Assembly	1.0000±0.0030	0.9968±0.0007	0.9968±0.0031	1.0015±0.00001	1.0015±0.0030
HEU-MET-FAST-009.1	Be-Reflected HEU (~89.6) Sphere, VNIITF	0.9992±0.0015	0.9995±0.0006	1.0003±0.0016	0.9933±0.00001	0.9941±0.0015
HEU-MET-FAST-009.2	BeO-Reflected HEU (~89.6) Sphere, VNIITF	0.9992±0.0015	0.9962±0.0006	0.9970±0.0016	0.9923±0.00001	0.9931±0.0015
PU-MET-FAST-018	Be-Reflected Pu (94.79) Sphere, Planet Assembly	1.0000±0.0030	1.0016±0.0006	1.0016±0.0031	1.0034±0.00001	1.0034±0.0030

Done

Figure 2-6. Example BIRT report. Table can be customized to show whatever the user desires. This example shows k_{eff} results from MCNP and PARTISN with the k_{eff} values color coded based on their agreement with the benchmark

CHAPTER 3
DEMONSTRATION OF THE NDVV SYSTEM

Example Problem Specification

To demonstrate the NDVV system, a set of thirteen experiments from the ICSBEP Handbook have been selected for evaluation using MCNP5-1.40 with three different data libraries: JEFF31, ACTI, and ENDF70. In all, 39 simulations were performed using the automated NDVV system. More benchmarks could have easily been included; however, the author felt this subset provided an adequate demonstration without overwhelming readers with superfluous results.

Prior to the selection of the thirteen benchmarks for this demonstration, a much larger repository of benchmark templates were constructed. The MCNP team at LANL retains a set of approximately 90 MCNP input files based on ICSBEP experiments. NDVV initialization tools were used to convert those 90 MCNP inputs into the beginnings of a template library. Those templates were then reviewed and compared with the official ICSBEP specifications. Any errors were corrected and additional benchmark data was added. Once the review was complete, each benchmark was placed into an internal LANL document control system so that changes require official review. This is expected to be the model most users will follow when creating their own template library.

The thirteen ICSBEP criticality experiments for this example were chosen to demonstrate the breadth of the problem type. They feature a number of fuel and reflector types and span the full spectrum of neutron energy. They were also chosen because they were included in the ENDF/B-VII.0 benchmarking performed by van der Marck. Therefore, the MCNP results have some basis for comparison. These ICSBEP benchmarks are described in detail in Table 3-1.

For this example MCNP5-1.40 was used for all calculations. It is typical for high-fidelity library comparisons to prefer the Monte Carlo method due to its "exact" treatment of geometry and cross sections.[9,20] Also the limited support for PARTISN in the current NDVV implementation would constrain the experiments to spherical geometry only. It was determined that performing MCNP calculations with a wide variety of problems and libraries would represent a reasonable example of how the NDVV system would be applied by end-users.

The three libraries included are common evaluations used in many nuclear calculations. The JEFF31 MCNP library is based on the JEFF-3.1 nuclear data library currently used in Europe. This library was created by using cross section data from the JEFF-3.1 nuclear data library and processing it with the NJOY code into the ACE format used by MCNP.[16]

The ACTI library is a combination of 41 newer evaluations and the ENDF66 MCNP data library. Most of the ACTI evaluations were eventually included in the US library ENDF/B-VI.8.[18] This library was used with the thermal scattering data from a 2002 evaluation as detailed in Reference 19.

The ENDF70 MCNP library is based on the latest release of the US ENDF/B series and represents the current standard for nuclear data in the US. This MCNP library was created using ENDF/B-VII.0 data and includes the latest 2007 thermal scattering data. The processing of ENDF70 is detailed in Reference 7.

It is important to emphasize that the intent of this example is to demonstrate the capability of the NDVV system and not for the purposes of validating the libraries included in the example. Many works have already been published with the focus of data library accuracy. Such validation studies can be found in References 9 and 20.

Execution of the NDVV system

A simple NDVV input file was built to specify the inclusion of the aforementioned templates. This file also included the location of the MCNP5-1.40 executable and the locations of the three libraries on the available computer resources. Using this input specification and the high-performance computer resources at LANL, this NDVV project was executed to demonstrate the prototype system.

The time required to complete this run is not of first importance. Users' computing resources can vary greatly. In this case, the high-performance cluster computers at LANL could execute all 39 simulations in parallel and greatly reduce the time required. The important point from this exercise was the limited need for user involvement offered from the NDVV system. The creation of transport inputs was handled entirely by the NDVV system as well as the execution of the 39 MCNP calculations. None of these steps required any user intervention.

Following the NDVV run phase, the NDVV output parsing capability was executed. This placed the MCNP results into a special NDVV data structure and produced a summary table similar to that in Table 3-2. As an added confirmation, the results for the ENDF70 libraries were compared to those obtained by van der Marck in Reference 6 (outside the NDVV system). The values of k_{eff} obtained by the NDVV calculations were found to be consistent with van der Marck's results for ENDF70 (ENDF/B-VII.0).

Storage and Displaying of Results

The database storage option was executed to place the example problem results into a PostgreSQL database. This database was created on a server at LANL prior to the NDVV run using an SQL script. Following the storage in the database, a custom BIRT view was created to present the k_{eff} results from each library and the benchmark k_{eff}. The resulting report is shown in Figure 3-1. This view is entirely customizable and can display any parameters the user desires.

An important benefit of this report generation methodology is its ability to automatically update based on the results in the NDVV database. If more runs are performed in the future and the results are inserted into the same database, then the BIRT view will automatically update to show those new results. For example, a NDVV run could be performed using these same templates with another data library, such as the Japanese library, JENDL. After the results are stored in the database the BIRT view will automatically update to show those new results. This feature is not found with static files such as Excel plots or even the gnuplot files created by NDVV.

This example BIRT report is hosted on a server at LANL and was viewed through a web-browser. This demonstrates how a user could execute NDVV runs, store their results to a database server, and then create a custom view of the results to be accessed via a website. This would allow members of the nuclear data community to collaborate and compare results easily.

The gnuplot option was also run to create the plot featured in Figure 3-2. All the necessary files for gnuplot were automatically created by the NDVV system based on the MCNP results.

NDVV Results Table

Font color key:

- C/E within 1 std. deviation from unity.
- C/E between 1 and 2 std. deviations away from unity.
- C/E between 2 and 3 std. deviations away from unity.
- C/E more than 3 std. deviations away from unity.

ICSBEP Name	Benchmark k_{eff}	Description
HEU-MET-FAST-022	**1.0000 ± 0.0019**	*Duralumin Reflected HEU Sphere (90% U-235)* *Simplified Model* *VNIIEF Facility*

Library	MCNP k_{eff}	MCNP C/E
ENDF70	0.9972 ± 0.0006	0.9972 ± 0.0020
ACTI	0.9921 ± 0.0006	0.9921 ± 0.0020
JEFF31	0.9939 ± 0.0006	0.9939 ± 0.0020

ICSBEP Name	Benchmark k_{eff}	Description
HEU-MET-FAST-028	**1.0000 ± 0.0030**	*Natural Uranium Reflected High-Enriched Uranium Sphere* *Flattop Assembly* *See also CSEWG-F22*

Library	MCNP k_{eff}	MCNP C/E
ENDF70	1.0028 ± 0.0007	1.0028 ± 0.0031
ACTI	1.0026 ± 0.0006	1.0026 ± 0.0031
JEFF31	1.0004 ± 0.0006	1.0004 ± 0.0031

ICSBEP Name	Benchmark k_{eff}	Description
HEU-SOL-THERM-032	**1.0015 ± 0.0026**	*48" Unreflected Sphere of Uranyl Nitrate (93.2 wt% U-235) Solution* *See also ORNL-10 and CSEWG T-5*

Library	MCNP k_{eff}	MCNP C/E
ENDF70	0.9996 ± 0.0003	0.9981 ± 0.0026
ACTI	0.9988 ± 0.0003	0.9973 ± 0.0026
JEFF31	0.9994 ± 0.0003	0.9979 ± 0.0026

Figure 3-1. BIRT report generated from example run. Only first three benchmarks are shown for brevity. Data in BIRT view is obtained from PostgreSQL database were result data is stored. Once a view has been defined the data is automatically updated as results change.

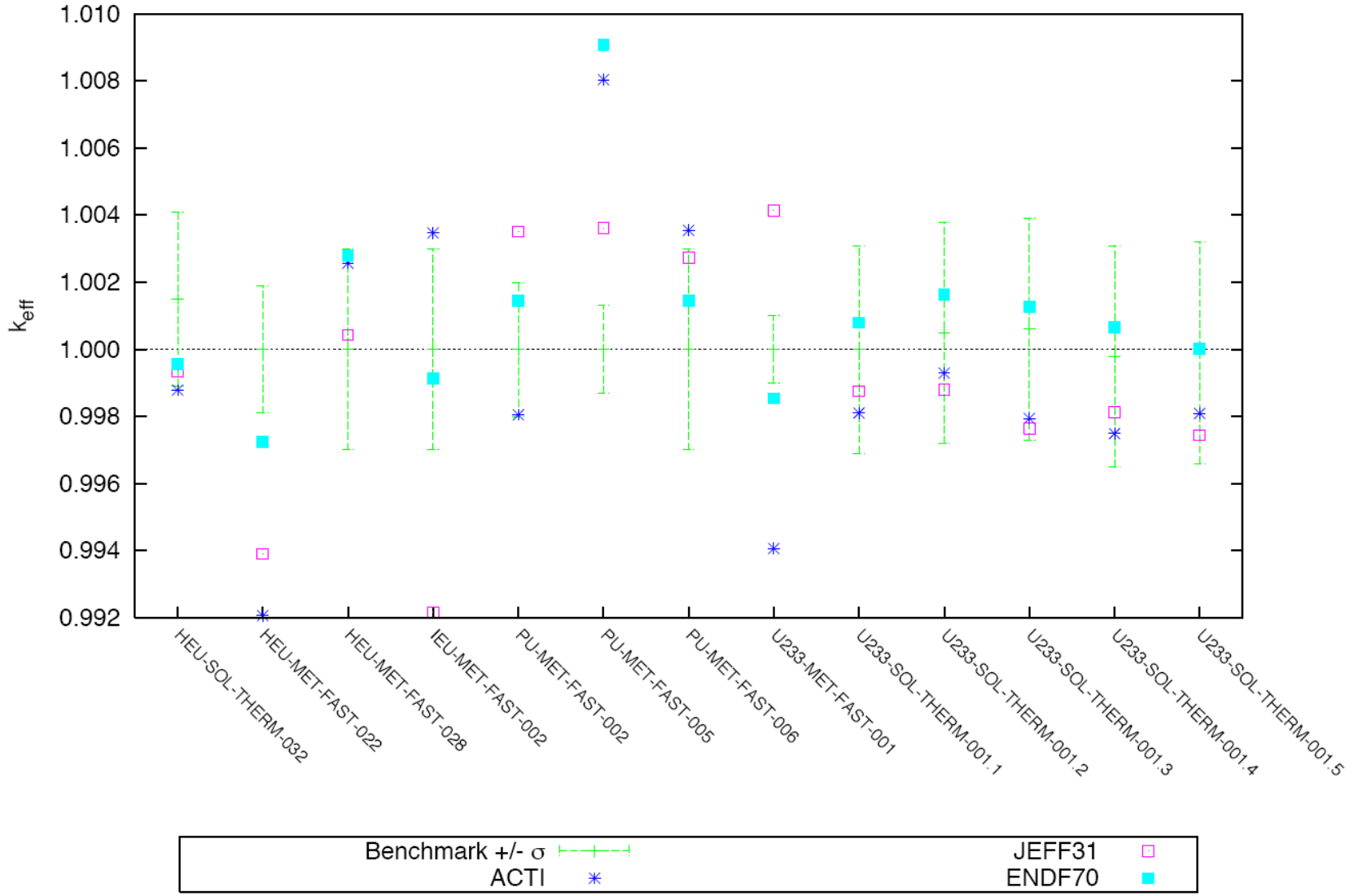

Figure 3-2. Example problem results from gnuplot. The Monte Carlo uncertainty is not shown on the plot for readability and because it is small relative to the benchmark uncertainty.

Table 3-1. ICSBEP experiments used in example NDVV run

ICSBEP Name	Description
HEU-MET-FAST-022	Duralumin Reflected HEU Sphere (90% U-235) Simplified Model
HEU-MET-FAST-028	Natural Uranium Reflected High-Enriched Uranium Sphere Flattop Assembly
HEU-SOL-THERM-032	48" Unreflected Sphere of Uranyl Nitrate (93.2 wt% U-235) Solution
IEU-MET-FAST-002	Jemima - Idealized Natural Uranium reflected stack of Natural Uranium and HEU plates
PU-MET-FAST-002	Jezebel - Bare sphere of Pu-239 with 20% Pu-240
PU-MET-FAST-005	Tungsten Reflected Plutonium Sphere with 4.9% Pu-240 Planet Assembly
PU-MET-FAST-006	Natural Uranium Reflected Pu Sphere - Flattop Assembly
U233-MET-FAST-001	Jezebel-233 - Bare Sphere of U-233
U233-SOL-THERM-001.1	Unreflected 27.24" Sphere of U-233 Nitrate Solution Experiment 1 - 1.0226 g/l
U233-SOL-THERM-001.2	Unreflected 27.24" Sphere of U-233 Nitrate Solution Experiment 2 - 1.0253 g/l
U233-SOL-THERM-001.3	Unreflected 27.24" Sphere of U-233 Nitrate Solution Experiment 3 - 1.0274 g/l
U233-SOL-THERM-001.4	Unreflected 27.24" Sphere of U-233 Nitrate Solution Experiment 4 - 1.0275 g/l
U233-SOL-THERM-001.5	Unreflected 27.24" Sphere of U-233 Nitrate Solution Experiment 5 - 1.0286 g/l

Table 3-2. Example results summary. Monte Carlo uncertainty not shown for clarity since it is significantly smaller than the benchmark uncertainty.

Problem	k_{eff}			
	Benchmark	JEFF31	ACTI	ENDF70
U233-SOL-THERM-001.1	1.0000 +/- 0.0031	0.99875	0.99810	1.00080
U233-SOL-THERM-001.2	1.0005 +/- 0.0033	0.99881	0.99929	1.00164
U233-SOL-THERM-001.3	1.0006 +/- 0.0033	0.99765	0.99794	1.00127
U233-SOL-THERM-001.4	0.9998 +/- 0.0033	0.99812	0.99749	1.00067
U233-SOL-THERM-001.5	0.9999 +/- 0.0033	0.99744	0.99809	1.00002
HEU-MET-FAST-022	1.0000 +/- 0.0019	0.99390	0.99206	0.99724
HEU-MET-FAST-028	1.0000 +/- 0.0030	1.00043	1.00257	1.00280
HEU-SOL-THERM-032	1.0015 +/- 0.0026	0.99936	0.99879	0.99956
IEU-MET-FAST-002	1.0000 +/- 0.0030	0.99215	1.00347	0.99914
PU-MET-FAST-002	1.0000 +/- 0.0020	1.00351	0.99805	1.00146
PU-MET-FAST-005	1.0000 +/- 0.0013	1.00361	1.00804	1.00907
PU-MET-FAST-006	1.0000 +/- 0.0030	1.00273	1.00355	1.00145
U233-MET-FAST-001	1.0000 +/- 0.0010	1.00414	0.99406	0.99854

CHAPTER 4
FUTURE WORK AND CONCLUSIONS

Future Work

Although the current NDVV system is complete and functional, most of the NDVV code is in the prototype phase. The development started from scratch and the software design has evolved as goals and requirements were still being determined. In order to ensure that the system remains modular, expandable, and maintainable, the software design should be reevaluated in a number of ways.

The first area that warrants revisiting is the XML benchmark template. The original specification was based mainly on MCNP because it allows for very flexible geometry and material definitions. Geometry issues arise with deterministic codes because they typically only allow for fixed coordinate systems (Cartesian, cylindrical, etc). Special geometry processing tools will be required to translate the MCNP surface based geometry into the mesh structure required for many deterministic codes.

The automatic determination of mesh fineness is another significant issue still to be solved. It is difficult for an automated system to determine the proper mesh-sizing for every region of a problem. It is possible to set rules based on particle mean free path in each cell; however, this would require an intimate knowledge of the problem prior to the NDVV run. This data would then have to be included in some manner in the benchmark template.

Until these issues are resolved, there will be significant limitations on the NDVV systems ability to utilize deterministic transport codes.

Another critical aspect of the NDVV system that warrants review is the controller/machine interface, particularly the manner in which commands are sent to remote machines. The need to send more complex commands will likely become more apparent as the system involves. Up to

this point the developers have relied on a minimal set of scripted system commands existing on each type of system. This can greatly limit the types of command that can be sent to each system. Given that Python is a system-independent language, it is possible that keeping the commands internal to Python may help meet the requirements of the NDVV system in a more robust, straightforward way. By using a wholly Python client-server system to send commands to remote machines, the developers can easily guarantee the same behavior on a wide variety of systems. The only drawback to this approach is that the minimum version of Python will be required on each system the user intends to use; however, for most users, this would not be a prohibitive requirement.

Following these critical maintenance issues, there are a number of prospective modules that could be plugged into the NDVV system. A full PARTISN module is likely to be completed first due to the already existing development. When the deterministic code issues are solved, full support for PARTISN could be implemented. This would allow the NDVV system to run three-dimensional stochastic and deterministic transport calculations.

Another major expansion could be the inclusion of more post-processing modules. Since the NDVV results data are stored in a standardized data structure, it would be possible to display theses results in more formats that those currently available. An example of this could be an auto-generated HTML webpage for quick display of the results on a web server. There are very few limitations to how the data could be displayed from this system. The developers expect the nuclear data community will serve to offer insights into the most convenient ways to display V&V results.

A GUI to drive the user input creation process is another potential addition to the NDVV system. A GUI would better facilitate the selection of templates, codes, and system resources.

Finally, more database modules could be developed for the NDVV system. It not expected of every user to use exclusively PostgreSQL. The current availability of Python modules for interfacing with PostgreSQL was the main factor for its selection during this prototype phase. It is likely that Python modules for interfacing with other databases, such as MySQL will become available soon. A structure already exists in the NDVV software design to allow for more database formats should they be desired.

Conclusions

The goal of the NDVV project is to produce a tool that allows members of the nuclear data community to more easily and quickly test data libraries and assess their applicability for a specific problem. Such a tool may be used to help justify use of a specific code and library for a specific application. Eventually, it may be also be used for formal uncertainty quantification. This should lead to a better understanding of data sensitivities, which can then feed back into improved evaluated data libraries.

The current implementation of the NDVV system represents a full prototype version of the aforementioned V&V tool. A format has been defined to describe benchmark experiments for use in the NDVV system. The processes are in place to convert those templates into full input files for MCNP and one-dimensional PARTISN. A control system is able to automatically manage and monitor the execution of those transport codes on a wide variety of systems. The NDVV system can parse both PARTISN and MCNP outputs for storage into standardized NDVV data structures. NDVV tools can then insert those results in a PostgreSQL database or create gnuplot plots based on the results.

APPENDIX
USER INPUT FILE FOR EXAMPLE PROBLEM

```
<?xml version="1.0" encoding="UTF-8"?>
<input>

    <check_frequency>0.5</check_frequency>

    <templates>
        <template_repository>/home/triplett/exampleTemplates</template_repository>
        <include_all>True</include_all>
    </templates>

    <codes>
        <code>
            <name>mcnp5-1.40</name>
            <type>MCNP</type>
            <libraries>
                <library>
                   <name>ACTI</name>
                </library>
                <library>
                   <name>JEFF31</name>
                </library>
                <library>
                    <name>ENDF70</name>
                </library>
            </libraries>
        </code>
    </codes>

    <machines>
        <machine>
            <name>xcompute3</name>
            <system_type>posix</system_type>
            <connection_type>local</connection_type>
            <num_sessions>4</num_sessions>
            <temp_dir>/home/triplett/scratch</temp_dir>
            <codes>
                <code>
                    <name>mcnp5-1.40</name>
                    <run_command>
                        /home/triplett/mcnp/mcnp5
                    </run_command>
                    <libraries>
                        <library>
                            <name>ACTI</name>
                            <library_file>/home/triplett/libraries/ACTI/xsdir</library_file>
                        </library>
                        <library>
                           <name>JEFF31</name>
                           <library_file>/home/triplett/libraries/JEFF31/xsdir</library_file>
                        </library>
                        <library>
                            <name>ENDF70</name>
                            <library_file>/home/triplett/libraries/ENDF70/xsdir</library_file>
                        </library>
                    </libraries>
                </code>
            </codes>
        </machine>
    </machines>
</input>
```

LIST OF REFERENCES

1. National Nuclear Data Center. 2008. Brookhaven National Laboratory. 6 May 2008 <*http://www.nndc.bnl.gov/endf*>.

2. The JEFF project. 2008. Nuclear Energy Agency. 6 May 2008 <*http://www.nea.fr/html/dbdata/projects/nds_jef.htm*>.

3. JENDL Japanese Evaluated Nuclear Data Library. 2008. Japanese Atomic Energy Agency. 6 May 2008 <*http://wwwndc.tokai-sc.jaea.go.jp/jendl/jendl.html*>.

4. X-5 MONTE CARLO TEAM, *MCNP – A General Monte Carlo N-Particle Transport Code, Version 5*, Los Alamos National Lab report, LA-UR-03-1987, (April. 2003).

5. TRANSPORT METHODS GROUP, *PARTISN: A Time-Dependant Parallel Neutral Particle Transport Code System*, Los Alamos National Lab report, LA-UR-05-3925, (May 2005).

6. S. C. VAN DER MARCK, "Benchmarking ENDF/B-VII.0," *Nuclear Data Sheets,* **107,** *12,* 3061 (2006).

7. M.B. CHADWICK, P. OBLOZINSKY, M. HERMAN, N.M. GREENE, R.D. MCKNIGHT, D.L.SMITH, et al, "ENDF/B-VII.0: Next Generation Evaluated Nuclear Data Library for Nuclear Science and Technology," *Nuclear Data Sheets,* **107,** *12,* 2931 (2006).

8. W. HAECK, *JEFF 3.1 validation results on LLNL pulsed spheres*, JEF/DOC-1141, 2006.

9. A. C. KAHLER, "Monte Carlo Eigenvalue Calculations with ENDF/B-VI.8, JEFF-3.0, and JENDL-3.3 Cross Sections for a Selection of International Criticality Safety Benchmark Experiment Project Handbook Benchmarks," *Nuclear Science and Engineering,* **145,** 213 (2003).

10. T. KAWANO, K. M. HANSON, S. FRANKLE, P. TALOU, M. B. CHADWICK, AND R. C. LITTLE, "Evaluation and Propagation of the ^{239}Pu Fission Cross-Section Uncertainties Using a Monte Carlo Technique," *Nuclear Science and Engineering*, **153**, 1-7 (2006).

11. M. LUTZ, *Programming Python*, 3rd Ed., O'Reilly Media Inc., Sebastopol, CA (2006).

12. J. B. BRIGGS, Editor, *International Handbook of Criticality Safety Benchmark Experiments*, NEA Nuclear Science Committee, NEA/NCS/DOC(95)03, (2007).

13. R. MCKNIGHT, Editor *ENDF-202, Cross Section Evaluation Working Group Benchmark Specifications,* Brookhaven National Laboratory report, BNL 19302 (ENDF-202) with various revisions (1974).

14. BIRT Project. 2008. Eclipse. 6 May 2008 <*http://www.eclipse.org/birt/phoenix/*>.

15. P MIKULÍK and E. MERRITT. gnuplot. 6 May 2008 <*http://www.gnuplot.info*>.

16. O. CABELLOS, *Processing of the JEFF-3.1 Cross Section Library into a Continuous Energy Monte Carlo Radiation Transport and Criticality Data Library*, OECD report NEA/NSC/DOC(2006)18, May, 2006.

17. S. C. FRANKLE, R. C. REEDY, and P. G. YOUNG, "ACTI: An MCNP Data Library for Prompt Gamma-ray Spectroscopy," Proceedings of the 12th Biennial Radiation Protection and Shielding Topical Meeting, Santa Fe, NM, April 15-19, 2002.

18. J. M. CAMPBELL, S. C. FRANKLE, and R. C. LITTLE, "ENDF66: A Continuous-energy Neutron Data Library for MCNP4C," Proceedings of the 12th Biennial Radiation Protection and Shielding Topical Meeting, Santa Fe, NM, April 15-19, 2002.

19. R. C. LITTLE and R. E. MACFARLANE, *SAB2002 - An S(a,b) Library for MCNP*, Los Alamos National Laboratory report, LA-UR-03-808, February 3, 2003.

20. R. D. MOSTELLER, "Comparison of Results for the MCNP Criticality Validation Suite Using ENDF/B-VII.0 and Other Nuclear Data Libraries," *Transactions of the American Nuclear Society*, **96**, 417-419 (2007).

BIOGRAPHICAL SKETCH

Brian Triplett began attending the University of Florida in August of 2001 after graduation from Allen D. Nease High School in St. Augustine, FL. During his second semester, he chose Nuclear and Radiological Engineering as his major. During his undergraduate work, he had the opportunity to intern at the Crystal River nuclear power plant in Crystal River, FL. He also participated in undergraduate research through the University Scholars Program.

Brian is an active member of the American Nuclear Society. He attended many American Nuclear Society national conferences while at UF. He served as the treasurer of the local student chapter of the American Nuclear Society for two years.

Following graduation in December of 2005, Brian began his graduate work also in nuclear engineering at the University of Florida. He worked under the direction of Dr. Samim Anghaie and assisted with Dr. Anghaie's undergraduate Reactor Thermal Hydraulics course.

He obtained his Master of Science degree in nuclear engineering sciences with his thesis titled "Development of an Automated Testing System for Verification and Validation of Nuclear Data." This work was performed collaboratively with Los Alamos National Lab with funding from the Department of Energy's Office of Civilian Radioactive Waste Management.