A Dynamic Waste Isolation Pilot Plant Performance Assessment Tool - 12490

Anthony M. Scopatz*, Moo Lee**, Chris Camphouse**, Jonathan March*, Warren Weckesser*, Eric Jones*
*Enthought Inc, Austin, Texas, 78701
**Sandia National Laboratories, Carlsbad, NM, 88220

ABSTRACT

The Waste Isolation Pilot Plant (WIPP) Performance Assessment (PA) methodology comprises a toolbox used to demonstrate regulatory compliance of the repository after facility closure. The PA framework rests upon an extensive suite of computational codes. In some cases, significant alteration of code inputs is a tedious and difficult task. Due to the nature of the application for which they are used, PA codes used in support of WIPP regulatory compliance demonstration must satisfy stringent quality assurance requirements. Consequently, many of the coding practices used during original code development are still implemented today. A more efficient workflow configuration has the potential to alleviate difficulties associated with extensive code input modifications. Here, this potential is assessed via an implementation of a more flexible scientific workflow system for a subset of the codes used in WIPP PA.

INTRODUCTION

Performance assessment (PA) tools, while algorithmically very sophisticated within their domain, can be difficult to use by all but a select group of experts [1]. The work proposed here seeks to increase usability and accessibility of existing tools in the Waste Isolation Pilot Plant (WIPP) PA configuration through the use of high-level workflow management and a visual interface.

The Waste Isolation Pilot Plant is the first geologic repository built to permanently dispose of transuranic radioactive waste [2]. Sandia National Laboratories, headquartered in Albuquerque, NM, has been the official scientific advisor on WIPP since 1974 when the project entered its initial planning stages. Since the original Compliance Certification Application (CCA) of 1996, regulatory compliance of the WIPP has been demonstrated by performance assessment calculations undertaken by Sandia. The models, parameters, and numerical codes that underlie the performance assessment methodology are continually refined and improved as part of the WIPP recertification process.

The recertification process provides the impetus for bringing modern and innovative software development tools to bear on WIPP PA. Performance assessment software should enable the dynamic exploration and analysis of data, physical models, and the system as a whole while maintaining the necessary quality assurance pedigree. Since the WIPP project was first undertaken, computer science has undergone several paradigm shifts. The focus of this work is an investigation of advantages provided by these new methods in the context of WIPP PA, particularly by implementation of improved workflow management tools developed since the days of the original CCA.

Scientific workflows are an increasingly essential portion of any fully-fledged scientific application. They comprise the infrastructure around the computational kernel(s), delegate tasks, and interface with other more general tools (the operating system, other languages, etc.) While scientists are proficient in developing algorithms in their domain, and software developers

are competent at creating general tools, the skills and knowledge to develop workflows are often lacking [3].

Though scientific workflows are often under-developed, they support core research aims. The primary goal they serve is persisting how, when, and where an application is executed. This affords a degree of provenance which is directly tied to the reproducibility of the work. This is especially important in stochastic algorithms where the initial conditions of each run must be stored.

Additionally, a modular workflow system allows the researcher to execute part or the entire suite of tools, as desired. Especially for computationally expensive kernels, such a capability enables a more judicious use of resources. This relates to the goal of overall ease of use. Other factors affecting this goal include interactive data exploration and real-time data analysis.

The following section discusses the historical state of the WIPP PA project and the proposed initial workflow additions. Next, the current implementation is evaluated. Finally, concluding remarks are presented.

METHODOLOGY

The primary solver in the WIPP PA suite of codes is called BRAGFLO (Brine and Gas Flow), and is executed with several support applications [4]. BRAGFLO is a two-phase flow solver primarily used to compute pressures and brine/gas saturations in the WIPP and its surrounding environment after facility closure. The suite of BRAGFLO codes includes Genmesh, Matset, LHS, ICSet, Algebra1, Prebrag, Postbrag, and Algebra2. Figure 1 is a diagram of how the codes in this suite are connected. Most of these are either written in FORTRAN or are VAX/VMS DCL scripts.

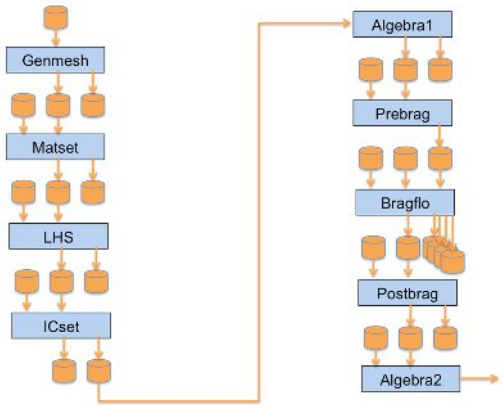

Fig 1. BRAGFLO Suite Schematic

The computational grid used in BRAGFLO is defined in the Genmesh step. Making changes to the BRAGFLO grid requires the user to manually make corresponding changes to grid element and node references in several of the codes downstream from Genmesh. Such rote tasks, while necessary in low-memory computing environments of past decades, are easily automated in modern environments. Specifically, the problem of implementing design geometries is well handled by graphical computer-assisted design tools.

The framework presented here leverages an existing, dynamic, and well-tested development stack. Tools for visualization, provenance, and scalability already exist in the Python ecosystem [5], and have been used in high-performance scientific software. The core scientific Python development stack (NumPy, SciPy, MatPlotLib, and Chaco) [6] were used here to provide a human interface layer on top of the BRAGFLO suite.

Furthermore, portions of the BRAGFLO workflow may be replaced by a more sophisticated workflow management strategy. In this strategy, computationally expensive kernels remain in a low-level language (typically statically typed and compiled). However, this core solver will expose an application programming interface (API) to a high-level language (often dynamically typed and executed). The workflow software is implemented in the high-level language, and controls the execution of the suite.

While applying this workflow strategy to the BRAGFLO suite, the core BRAGFLO solver remains unchanged from its Fortran implementation. Input and output readers and writers were implemented to pipe data from BRAGFLO to a variety of other tools. The workflow itself, whose purpose it is to interconnect the BRAGFLO suite, is then written in Python.

The non-BRAGFLO tools in the suite may also remain in their original Fortran or VAX scripts. However in many cases, new high-level implementations of supporting codes in the suite can be used. For example, Prebrag and Postbrag are algorithms which validate BRAGFLO input decks and perform standard operations on outputs. These standard tasks are handled more simply by implementing them in the same language as the workflow (Python). Moreover, codes such as LHS, which performs Latin hypercube sampling, could be replaced by an existing external Python implementation. By encapsulating supporting algorithms in the workflow itself, both maintenance overhead and initial knowledge needed to use the application are reduced.

To further reduce the prior knowledge requirements, a 2D graphical repository grid designer can both aid the creation of hypothetical repository inputs, and plot existing repositories. In conjunction with the workflow strategy above, the grid designer removes the need to manually edit many input files to capture changes in geometry.

One of the primary goals of a scientific workflow system is to aid in the dynamic exploration of data. In WIPP PA, repository conditions are typically generated from many (300) sets of initial conditions. One must be able to ‘drill down’ into any runs which produce outliers, to discover the cause of the atypical results. For example, particular combinations of repository initial conditions and sampled parameter values for the surrounding geology can yield unexpected results. A flexible workflow paradigm allows the analyst to quickly investigate and understand the underlying causes of such outliers.

Raw data exploration is handled by the interactive interpreter IPython. This tool connects with location-specific, dependent variable plotting and other context-sensitive parameter aggregation tools. This workflow system, which is continually extended by an open source community, aids both novice and expert PA users.

Finally, the proposed WIPP workflow wraps all of above components into a single application. This application would allow the construction of complete or partial execution pipelines. Most importantly, the pipeline built would also be able to be saved, reloaded from a saved state, and executed later or elsewhere. Combining pipeline persistence with version control grants the PA suite reproducibility and provenance.

RESULTS

The WIPP PA workflow outlined in the previous section has yet to be fully implemented due to time constraints. However, major portions of the replacement workflow have been demonstrated. The figures below are screen captures from the end-user application.

The first, crucial piece implemented was a port of the BRAGFLO code itself from VAX to Solaris and Linux systems. In addition to supporting the Intel Fortran compiler (ifort), BRAGFLO was also updated to support the GNU Fortran compiler (gfortran). This enabled the computational kernel to be executed on a variety of modern platforms.

The BRAGFLO runs are specified with a highly formatted text file. (In effect, BRAGFLO's input is a custom language for simulating repositories.) Python functions were written to read, parse, and write such files. This granted a high-level API to BRAGFLO. End users need only know the widely used open-source Python language to be able to specify repositories.

BRAGFLO's primary output is a binary file in a custom format. While this specification is useful for some purposes, it can become a limiting factor in being able to interpret run results as a non-specialist. Thus two converters were implemented for BRAGFLO output. The first takes the binary output and loads it into a Python dictionary in memory. The second rewrites the output information to the standard binary hierarchical data format (HDF5) [7]. This exposes BRAGFLO output to Python and all other languages with HFD5 bindings.

From here, because the appropriate APIs were available, the Prebrag and Postbrag portions of the suite were replaced with simple Python modules which implemented the same functionality. Moreover, the Algebra2 code, which computes standard aggregations on BRAGFLO output, was also reimplemented in Python. A method was used here whereby independent blocks of code are executed in a sequence of context managers. This is known as blocks & contexts. It is similar to a procedural programming paradigm that hides the underlying function calls from the user. Furthermore, blocks may be *a priori* combined to form larger blocks; this is distinct from creating a function that calls many other functions. The Python version of Algebra2 therefore presents a significantly more dynamic capability than its forebear. This enables users to create custom versions of Algebra2, which perform only the desired aggregations. Upon adoption, this capability aids efficiency and testing.

The remaining PA work sought to provide a high-level graphical user interface (GUI). This is the last, fundamental step in scientific workflow applications. Though this portion contains relatively little of the computational burden, the inclusion of a GUI vastly increases the pool of potential users. In the contemporary software development world many user interface options are available. These are dominated by two prevailing paradigms: traditional local rich clients and web-based applications. For reasons of security and the existence of tools in the scientific Python stack, a rich client was chosen for WIPP PA.

Prior to executing BRAGFLO, a grid designer GUI may be used to construct a repository geometry either from scratch or from pre-built models. Figure 2 displays a WIPP BRAGFLO model 2200 years after closure with a sample borehole incursion. Regions in this model have coloring based on material. Additionally, materials have physical properties associated with them, such as permeability and porosity. Changes to regions, materials, or geometry are reflected in any new BRAGFLO input decks exported from the designer. The designer, more than simply being a convenience tool, enables perturbation studies previously inconceivable due to the time required for repeated manual reconfiguration.

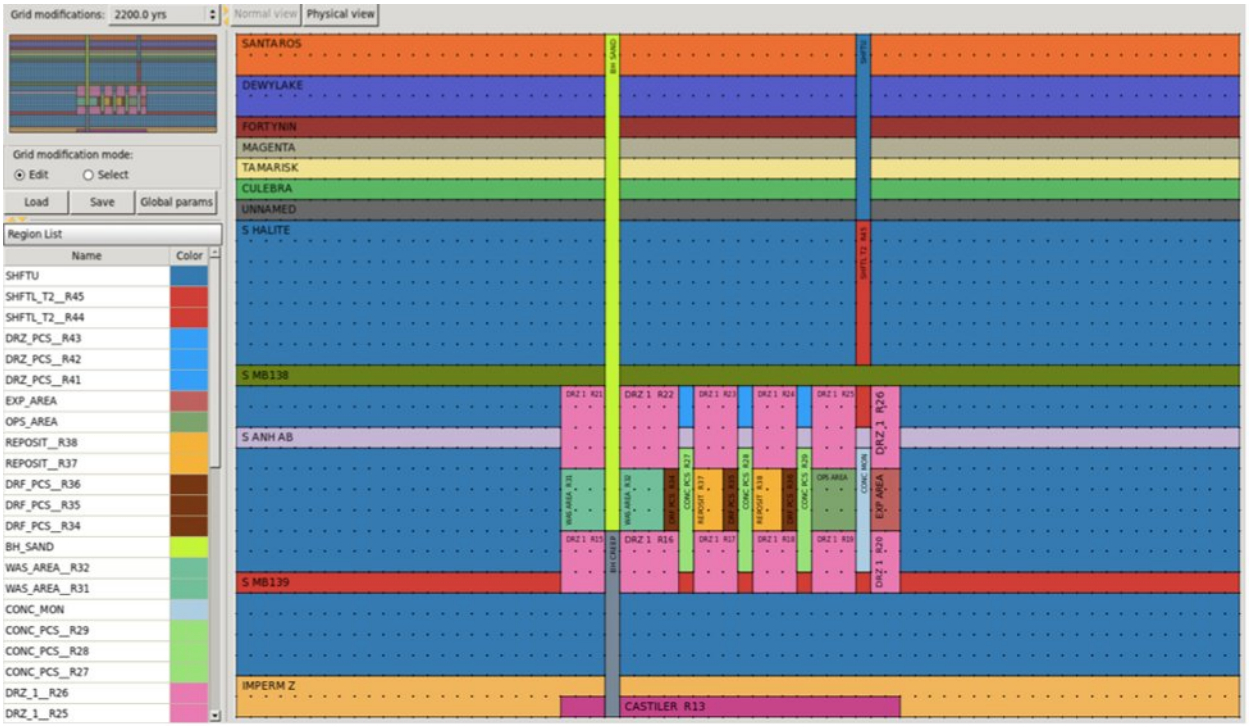

Fig 2. Repository Grid Designer Sample

After a set of BRAGFLO runs has been executed, an analysis application may be run. Figure 3 shows an example of this GUI. This tool converts all BRAGFLO output files into their HDF5 representations and loads the data into memory. From here, a specific run, time after closure, color palette, and other options may be selected from the top drop-down menus. This in turn determines the repository visualization seen below the selection menus. By selecting any grid element in the visualization, the user may display graphs of the independent or dependent

variables at the selected location. Additionally, figures that are aggregations over all runs, such as horsetail plots, may be computed (shown in Figure 3). Finally, at the bottom of the application, an IPython terminal is presented. Pre-loaded into this terminal is all of the data from each of the BRAGFLO runs. This allows the user to inspect the raw data directly and derive other quantities from this context.

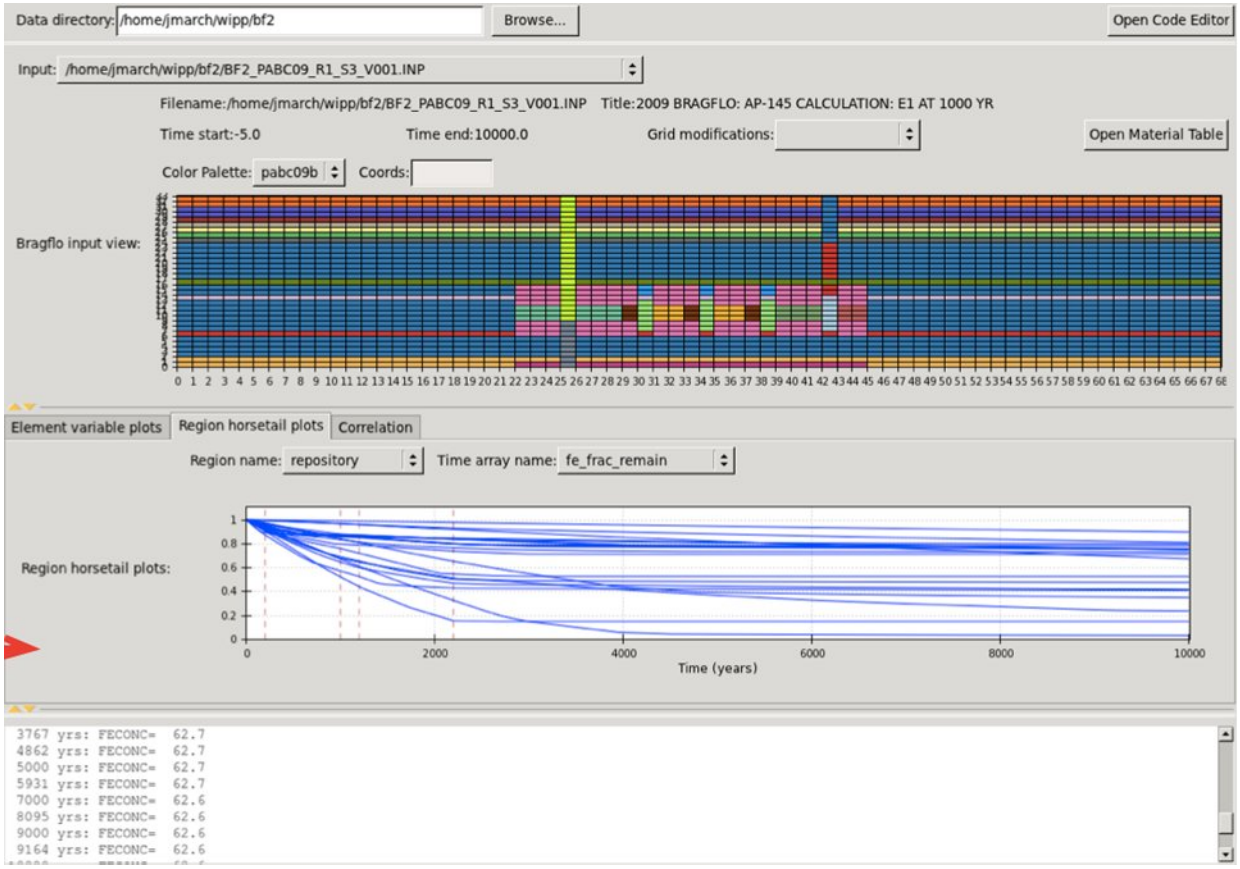

Fig 3. WIPP PA Analysis Application Sample

Genmesh, Matset, LHS, and ICSet were not directly considered in this study except to the extent that their grid definition capabilities are performed by the grid designer GUI. Similar to Prebrag and Postbrag, these codes are likely candidates for a pure Python reimplementation.

It should be noted that the BRAGFLO suite comprises only one of many code bases that comprise the WIPP PA software stack.

CONCLUSIONS & FUTURE WORK

The scientific workflow approach taken here for a dynamic PA system enables users from disparate backgrounds to dramatically shorten the time between hypothesis and analysis by decreasing the amount of a priori knowledge, from a range of disciplines, needed to execute the code. Having smaller iteration times allows for more ideas to be tested and explored, which leads to safer and more optimized systems.

Note that these high-level, dynamic tools are intended only for initial scoping studies on the personal computer of a researcher. Full, regulatory compliance calculations may occur only within a qualified computing environment. However, the WIPP PA tools here may guide future research and indicate regions of the analysis space that are worth further study.

This next generation of PA software provides the ability to perform scoping investigations of repository performance quickly and easily, and has an accessible and useful interface to a variety of users, such as fuel cycle systems designers, domain experts such as repository modelers, and policy makers.

The purview of this project allows for many opportunities for future work. Foremost among these is the desire to implement the full BRAGFLO suite within the workflow. This will entail porting or wrapping Genmesh, Matset, LHS, and ICSet within Python. Moreover, unifying the two GUIs into a single driver application would be a natural next step. Once the BRAGFLO suite is completed, other portions of WIPP PA could be implemented with corresponding and interoperable workflows. Likely first candidates for this are those codes that are similarly computationally intensive, such as the one used to generate complementary cumulative distribution functions used to demonstrate regulatory compliance (code CCDFGF).

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. This research is funded by WIPP programs administered by the Office of Environmental Management (EM) of the U.S Department of Energy.

REFERENCES

1. United States Department of Energy. (2010). ASCEM FY10-FY15 Integrated Modeling Implementation Plan (WSB 1.1). Technical Report ASCEM-PM-10-01-01, Environmental Management, USA.
2. United States Department of Energy. (2007). WIPP Chronology. Fact Sheet, WIPP Information Center, Carlsbad, NM.
3. Kelly, D. F., (2007). A Software Chasm: Software Engineering and Scientific Computing, *IEEE Software*, 24 (6), p. 120-119.
4. Vaughn, P., Bean, J.E., Helton, J.C., Lord, M.E., MacKinnon, R.J., and Schreiber, J.D., (2000) Representation of two-phase flow in the vicinity of the repository in the 1996 performance assessment for the Waste Isolation Pilot Plant. *Reliability Engineering and System Safety*, 69 1–3, pp. 205–226.
5. Martelli, A., Ravenscroft, A., and Ascher, D., (2005). Python Cookbook, Second Edition, O'Reilly Media, Sebastpool, CA.
6. Langtangen, H. P., (2009). A Primer on Scientific Programming with Python, Springer, New York, NY.
7. The HDF Group. Hierarchical data format version 5, 2000-2010. http://www.hdfgroup.org/HDF5.