Annals of Nuclear Energy 85 (2015) 856–868

Contents lists available at ScienceDirect

Annals of Nuclear Energy

journal homepage: www.elsevier.com/locate/anucene

A domain-specific analysis system for examining nuclear reactor simulation data for light-water and sodium-cooled fast reactors ☆

Jay Jay Billings *, Jordan H. Deyton, S. Forest Hull III [1], Eric J. Lingerfelt, Anna Wojtowicz

Oak Ridge National Laboratory, PO Box 2008, MS6173, Oak Ridge, TN 37831, USA

ARTICLE INFO

Article history:
Received 30 June 2015
Accepted 2 July 2015
Available online 17 July 2015

Keywords:
Nuclear reactors
LWR
SFR
Simulation
Modeling
High-performance computing

ABSTRACT

Building a new generation of fission reactors in the United States presents many technical and regulatory challenges. One important challenge is the need to share and present results from new high-fidelity, high-performance simulations in an easily usable way. Since modern multiscale, multi-physics simulations can generate petabytes of data, they will require the development of new techniques and methods to reduce the data to familiar quantities of interest (e.g., pin powers, temperatures) with a more reasonable resolution and size. Furthermore, some of the results from these simulations may be new quantities for which visualization and analysis techniques are not immediately available in the community and need to be developed.

This paper describes a new system for managing high-performance simulation results in a domain-specific way that naturally exposes quantities of interest for light water and sodium-cooled fast reactors. It describes requirements to build such a system and the technical challenges faced in its development at all levels (simulation, user interface, etc.). An example comparing results from two different simulation suites for a single assembly in a light-water reactor is presented, along with a detailed discussion of the system's requirements and design.

1. Introduction

Modeling and simulation has always played a vital role in nuclear engineering and its applications. It is becoming even more relevant as the community starts examining and constructing new types of reactors. Although existing computational models of the conventional fleet perform very well within the operational experience base of today's power plants, they do not necessarily provide the predictive capability needed to enable the deployment of completely new designs or to enable the use of today's designs for very different modes of operation. New simulation codes are under development in several efforts sponsored by the US Department of Energy's Office of Nuclear Energy, including those from the Nuclear Energy Advanced Modeling and Simulation (NEAMS) program (2011) and the Consortium for Advanced Simulation of Light Water Reactors (CASL) (2011).

These new codes are much different from the existing codes. Whereas tried-and-true codes employ sophisticated engineering models calibrated based on experimental observation to describe system behavior, the new code suites leverage high-performance computing (HPC) platforms to provide truly predictive simulators. The new codes can examine the physics of nuclear reactors with an unprecedented resolution at spatial and temporal scales ranging from the microstructure of the fuel all the way to the plant itself, and they are commonly capable of coupling neutronics, fuel mechanics, structural mechanics, and thermohydraulics (Gaston et al., 2009; Siegal et al., 2007; Turner et al., 2013). While they are very powerful, they inevitably generate many terabytes and even petabytes of data that greatly exceed what any single person can absorb and interpret alone. Even results that are refined through post-processing can still be gigabytes in size.

Additional challenges arise when the user actually examines and reviews the results. Large tables of data, even if well-organized, are not easily consumed by humans because of their size and general lack of sufficient context to eliminate

☆ *Notice of Copyright:* This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

* Corresponding author. Tel.: +1 865 241 6308.
E-mail address: billingsjj@ornl.gov (J.J. Billings).
[1] Florida Institute of Technology, 150 W. University Blvd., Melbourne, FL 32901, USA.

http://dx.doi.org/10.1016/j.anucene.2015.07.002
0306-4549/

ambiguity. Furthermore, in this particular area, there are often very large numbers (fluxes, $\approx 10^{12}$) associated with very small numbers (displacements, $\approx 10^{5}$) that make intuitive interpretation of the results difficult. More often than not, users are forced to implement their own codes on top of the post-processing routines to collect, assimilate, rescale, normalize, and plot the very small amount of data that they actually need, which is a very costly chore in terms of both time and resources. Comparing the results from new simulations to those from old simulations or experiments also requires custom code, thereby multiplying the cost of adoption.

The authors assert that it is not sufficient to leave these challenges unaddressed and "pitch it over the fence" to users and analysts. Instead, just as new codes are under development to address the physics questions, new technologies need to be developed to address data analysis. The system presented in Section 2 seeks to address this issue in three ways. First, it provides specialized data structures and input/output (I/O) libraries designed specifically for storing quantities of interest from nuclear simulations. Second, it provides a user interface that complements the I/O libraries to provide highly tailored views that put data in the proper context and reduce ambiguity. Finally, it provides an extension interface for adding custom analysis routines that can be easily coupled to routines for data mining and tailored analysis. An example of the utility of the system provided in Section 3 demonstrates the ease with which interesting information between two codes, one new and one seasoned, can be extracted from the system.

2. Architecture

A high-level overview of the system's logical architecture is presented in Fig. 1. The system is split into three parts, one each for users who examine data, developers who store data, and those interested in implementing custom analysis routines to examine the data. In each case, care was taken to make sure that the workflow of the respective actor was easy and intuitive for that use case.

Five high-level functional and nonfunctional requirements were used to guide the detailed design of the system based on feedback from interviews and experiments with prototypes, all of which were derived from the assumption that the simulation results must be shared with one or more people.

1. The data should be organized according to the natural layout of a nuclear reactor and not on the discretized geometry of the simulation code.
2. A data point has a position in three dimensions plus time with a value, uncertainty, and units.
3. A reactor is composed of *Parts*, which are represented by custom data structures, and data can be stored for every Part (literally *on* every Part) available.
4. The information is stored in a cross-platform, self-describing, binary compatible, open format.
5. The data must be easily and equally accessible from languages used for user interfaces *and* from languages used to author simulation codes.

The need to compare data from two simulations led directly to requirement 1. Storing output data on the grid or mesh used by the simulation would allow only immediate and direct comparisons with other results from the same code unless grid-point mapping or mesh-to-mesh transfers (Tautges and Caceres, 2013) were performed. However, storing data based on the layout of a reactor—on cores, assemblies, pins, etc.—allows for immediate comparison of results, albeit at the cost of pulling that information from the grid or mesh during the simulation. One drawback of this design is that some data *could* be lost when the grid or mesh is removed. However, this is not strictly the case, since data can be mapped to any Part in the reactor and given any position.

It is extremely important that the data be stored in a binary compatible open format available on multiple platforms and readable in multiple languages (requirements 4 and 5) for three reasons. First, the subject matter experts will examine the data on workstations, not clusters or supercomputers. Second, in many situations, it must be stored in such a way that it can be reviewed if required, possibly at a much later date. Finally, someone may need to view the data who does not have the expertise to write the code to read the database but is otherwise qualified to evaluate it via a user interface.

The definition of a reactor, requirement 3, loosely follows that of real light water reactors (LWRs) and sodium-cooled fast reactors (SFRs). Reactors are hierarchically composed of a set of Parts, where a Part describes both the large-scale structures of a reactor, such as assemblies, and smaller Parts like pellets or "material blocks". Data is stored for these Parts according to the description in requirement 2, including time, space, uncertainty and dimensional information, in addition to the value of interest.

Exact descriptions of the Parts of the nuclear reactors supported by this project are provided in Section 3. The Parts are exactly the same in both supported languages, the user interface, and the I/O

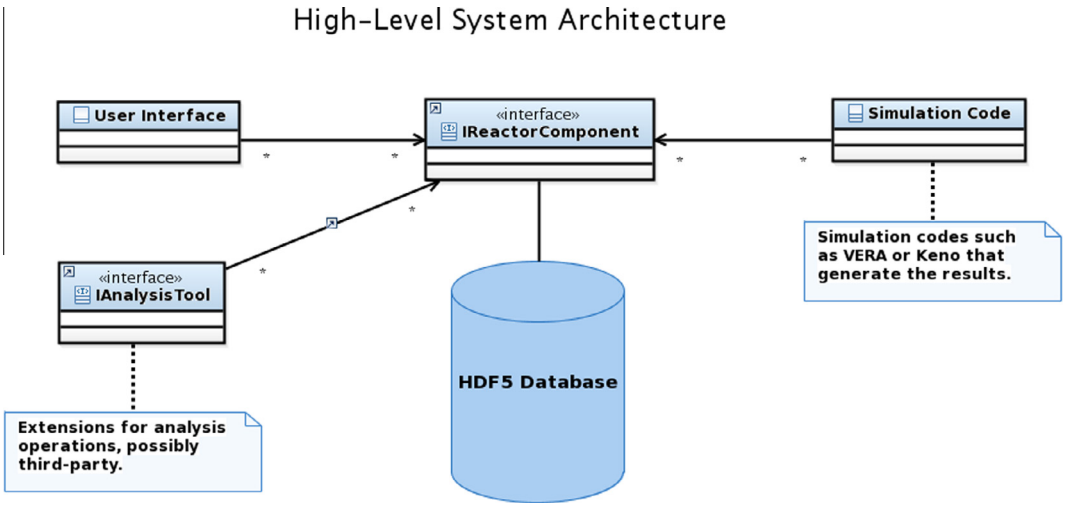

Fig. 1. A high-level Unified Modeling Language class diagram of the system's logical architecture, highlighting its focus on interacting with different Parts, or IReactorComponents, of the reactor data.