This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

SAND2018-6798C

Recent Analysis and Capability Enhancements to the ADAPT Dynamic Event Tree Driver

Zachary Jankovsky[ab*], Matthew Denman[b], and Tunc Aldemir[a]
[a]The Ohio State University, Columbus, Ohio, USA
[b]Sandia National Laboratories, Albuquerque, New Mexico, USA

Abstract: Dynamic probabilistic risk assessment (DPRA) methodologies and the dynamic event tree (DET) methodology specifically allow traditional PRA to be complemented by insights into time-dependent behavior. The ADAPT DET driver has been enhanced recently to provide greater capability to generate DETs and analyze results.

The functions that ADAPT uses to gather and present output data have been standardized and enhanced with the goal of automating as much of the process as feasible while remaining simulator and technology agnostic. These individual enhancements come together to reduce the burden on the analyst and allow insights to be discovered more quickly.

A recent goal has been the use of ADAPT on high performance computing (HPC) platforms. The number and granularity of treatment of uncertain parameters in a DET may lead to a state space explosion unless DETs are truncated (e.g., using a probability threshold) which may make the complete DET to be infeasible to run on local machines or small computer clusters. Progress can be greatly accelerated by using the large capacity of HPCs. A scheme is presented by which ADAPT gains the capability to distribute jobs to an HPC and retrieve the results seamlessly alongside other types of computation hosts.

Keywords: Dynamic PRA, ADAPT, HPC, DET.

1. INTRODUCTION

The ADAPT[1] tool [1, 2, 3] uses the Dynamic Event Tree (DET) approach to Dynamic Probabilistic Risk Assessment (DPRA)[2]. The effects of the DET branching rules are evaluated using an accident simulator code (such as MELCOR [4, 5], RELAP-5 [6], or SAS4A/SASSYS-1 [7, 8]). Once a DET has been generated, ADAPT provides a number of capabilities to use in mining the results for risk insights [9]. DET drivers with overlapping sets of features include Risk Analysis Virtual Environment (RAVEN) [10, 11], Accident Dynamics Simulator (ADS) [12], and MCDET [13], each of which has a slightly different development focus.

The DPRA tools require large computational resources and produce very large amounts of data (possibly on the order of tens of terabytes). In these respects, in order to fully benefit from the use of DPRA for decision making:

- DPRA tools must run more efficiently and effectively such that a rich data set is generated within a targeted uncertainty space with a reasonable investment of time and computational resources and
- the resulting data set must be mined using techniques that better leverage the time-dependent

[*] Corresponding author, Zachary.Jankovsky@sandia.gov
[1] ADAPT is no longer an acronym.
[2] No distinction is made between DPRA and Dynamic Probabilistic Safety Assessment (DPSA) in this paper.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

nature of the data.

In response to user needs and with a focus toward the above goals, a number of significant changes have recently been made to ADAPT to increase its capability to generate and analyze DETs. ADAPT enhancements for the analysis of a generated data set are summarized in Section 2. General improvements to ADAPT as a software project are summarized in Section 3. A scheme for running ADAPT using High Performance Computing (HPC) resources is presented in Section 4 along with preliminary results showing the acceleration that may be realized. The impacts of the work described in this paper on the capabilities of ADAPT are explored in Section 5.

2. NEW ADAPT DET ANALYSIS CAPABILITIES

The DET analysis capabilities described in this section were initially introduced previously but have since been refined and applied to new problems. They are summarized and the relevance of each capability to the goal of enhancing DPRA is explained. This section is arranged as follows:

- Section 2.1 describes the importance measures to capture the time-dependent effects of input parameters,
- Section 2.2 describes the use of multiple simulators to enhance the fidelity of an analysis and streamline data processing, and
- Section 2.3 describes how DETs may be selectively examined to focus on sequences[3] that meet a desired set of time-dependent characteristics.

2.1. Dynamic Importance Measures

Traditional importance measures [14] are ill-suited to many DPRA applications as they do not explicitly capture time-dependent effects. Previous dynamic approaches have used extensions of the traditional importance measures which calculate the change in the probability of reaching some failure state given a change in the probability of an input parameter [15, 16]. A difficulty with this approach is that there may be physical states of interest that do not constitute failure and may not even be represented as an event. For example, traditional importance measures and previous dynamic measures are capable of determining the relationship between an input parameter and the probability of reaching one of multiple fuel failure states. However, these approaches are not equipped to determine the relationship between an input parameter and a continuous output such as the fuel temperature which may give additional insight into the state of the system.

DYnamic Importance (DYI) measures[4] are a recent type of sensitivity measure in which a change in an input parameter may be directly interrogated for its impact on a continuous output variable. The DYI platform and three example measures were introduced in [17] and were exercised in a sample analysis in [8]. DYIs allow the analyst to explore relationships between parameters in an ordered way that has the potential to reveal phenomena of interest that might otherwise be lost in the mass of data that are created by DPRA.

[3] Sequence is used to refer to an ordered set of branches spanning from the initial branch of an event tree to a unique end state.

[4] DYIs were initially called Dynamic Importance Measures (DIMs) but were changed to avoid potential confusion with differential importance measures.

2.2. Linking of Multiple Simulators under ADAPT

There have been several efforts to link multiple simulators [18] or surrogate models [19] under a single DET driver. The implementation of this feature in ADAPT takes advantage of the loose coupling between ADAPT and simulators and was initially described in [20]. Since its introduction, the process by which multiple simulators communicate under ADAPT has been refined to reduce the effort required by the analyst to generate a DET. These improvements have taken the form of a more intuitive input format and a more flexible template for the wrapper script which manages communication between ADAPT and each of the simulators under its control.

Capturing additional phenomena under a single DET increases the value of the analysis as it may be mined easier with a common set of tools. Using this approach, time may be spent in searching for insights from the data rather than in retrieving and formatting the data [21]. The iterative passing of control of a single DET between simulators holds particular promise in providing insights that may have otherwise been missed as multiple specialized codes may each be leveraged against their areas of expertise within an overlapping time domain. An analysis is presented in [22] which takes advantage of the linking of MELCOR and the Radionuclide Transport, Removal, and Dose Estimation (RADTRAD) dose estimation code to extract insights relating to mitigation of a Residual Heat Removal (RHR) Interfacing System Loss of Coolant Accident (ISLOCA).

2.3. Reduction of a DET according to User-Input Rules

DETs generated to date under ADAPT have comprised tens of thousands to hundreds of thousands of branches (see Section 4.3). Such a large tree is infeasible for an analyst to examine at once and may include areas of the uncertainty space that are not of interest in the immediate analysis. A method was presented in [23] and applied to a test case in [8] by which a reduced copy of a DET may be generated for focused analysis.

With the ADAPT tree reduction capability, a set of time-dependent rules are specified by the analyst and then applied to the DET. For example, the user may wish to see only sequences involving:

- primary pressure below 3 MPa before 1 hour and
- onset of fuel damage before 3 hours.

The data files representing primary pressure and the intact fraction of the fuel will be assembled across the DET and compared to the rules. If all rules are met in any given branch, its entire sequence (from the initial branch to an end state) is marked to be copied into the reduced DET. Once all branches have been tested for all rules, a reduced DET of those sequences meeting the rules is created in the database and may be interrogated using the same tools as its source DET. This tool allows a DET to be examined with a focus on a specific area of the parameter space. Insights may be discovered within the reduced parameter space that would be obscured if the entire DET was evaluated at once.

3. GENERAL CODE IMPROVEMENTS

A number of incremental improvements have been made recently to the ADAPT code to improve its performance and capabilities for analyzing results. These improvements are presented as follows:

- Section 3.1 describes a process of caching outputs for later use to avoid duplication of effort,
- Section 3.2 describes the distribution of post-processing tasks among multiple processors,

- Section 3.3 discusses the refactoring of functions to perform fewer database operations, and
- Section 3.4 describes tools that provide insight into the progress of a DET.

3.1. Caching of Simulator Outputs

When using ADAPT features such as DYIs (see Section 2.1) or tree reduction (see Section 2.3), simulator outputs for an entire DET may be requested repeatedly. This can be a costly process as it involves reading a file from the remote run directory for every branch and then assembling the data together in memory. For a DET with 80,000 completed branches, this process takes multiple days running serially or 8 hours with 10 concurrent threads (see Section 3.2 for an explanation of parallel operation).

These outputs are now cached by ADAPT along with metadata concerning the state of the DET when results were gathered. Relevant functions in ADAPT check for a cached version before beginning the process of gathering outputs. If the DET has not changed since the last time information was gathered, the cached version is now used. The user may also choose to use the cached version even if the DET has changed. This allows a common reference point to be chosen for all measures within an analysis.

Caching has been observed (for the test application in Section 4.3) to decrease the time required for the second and subsequent requests for simulator data by a factor of four from the first request. For a commonly used output such as primary pressure or the fuel intact fraction, this may result in savings of many days both in terms of processor workload and the time the analyst must wait before examining results. Output caches may grow to require a significant amount of storage space, although typically not on the same order as the experiment itself. For example, each cached output (e.g., primary pressure) for the case study in Reference [22] required about 4 GB of storage while the entire DET required 30 TB.

3.2. Parallel Execution of Post-Processing Functions

Processes such as the extraction and gathering of simulator outputs are mostly a collection of independent tasks and are therefore prime candidates for parallelization. The independent portion of these tasks has been divided in ADAPT from the portion where they must be combined. The independent portion may now either be divided among a user-defined number of processes or run in the historical serial fashion if no direction is given.

Tasks such as running the simulator tend to be CPU-bound and will reach maximum efficiency when run in a number of processes equal to or slightly less than the number of physical cores on the machine. However, some tasks such as output data extraction and gathering are more likely to be bound by the speed of reading the disk rather than the CPU. With modern hardware, these tasks may be efficiently launched in a number of processes larger than the number of physical cores for a great increase in speed. There is a small loss due to the additional work required to separate the tasks and bring their results back together. Allowing the gathering of simulator outputs in ADAPT to be performed using multiple processes has resulted in time savings that scale to approximately 98% of the increase in processes for the test application in Section 4.3. That is, the process scales nearly linearly with the number of processors.

3.3. Reducing Database Calls

When analyzing outputs of a DET, ADAPT previously gathered data about each branch or sequence independently. The cost of such an approach was most apparent in assembling the relationships between each branch in a DET. Each branch and its immediate parent were read from the database independently resulting in a number of database operations roughly equal to the number of branches. Each database operation requires an unavoidable fixed amount of time to send the request and receive the results.

To decrease the cost of common tasks such as branch submittal, functions in ADAPT that are likely to perform many independent database operations have been refactored or replaced with different functions that accomplish similar goals using fewer operations. In the case of establishing parent-child branch relationships, the work of associating a large number of branches with each other can now be done on the client side. The entire table of branch relationships is requested in a single database operation and the relationships are resolved in a loop in local memory. This has resulted in time savings of over 60% every time the DET-wide relationships are requested for a DET with 20,000 branches.

3.4. Progress Indication

ADAPT has historically provided a web interface that displays the total number of branches, the number of branches running, and the number of branches completed in a DET. This interface gives limited insight into the progress of the DET as the times of branching events are not known a priori. The histogram in Figure 1 offers a new view to the progress of the DET (see Section 4.3). The simulation time is recorded each time a branch finishes and Figure 1 shows the distribution of these times for the current end states[5].

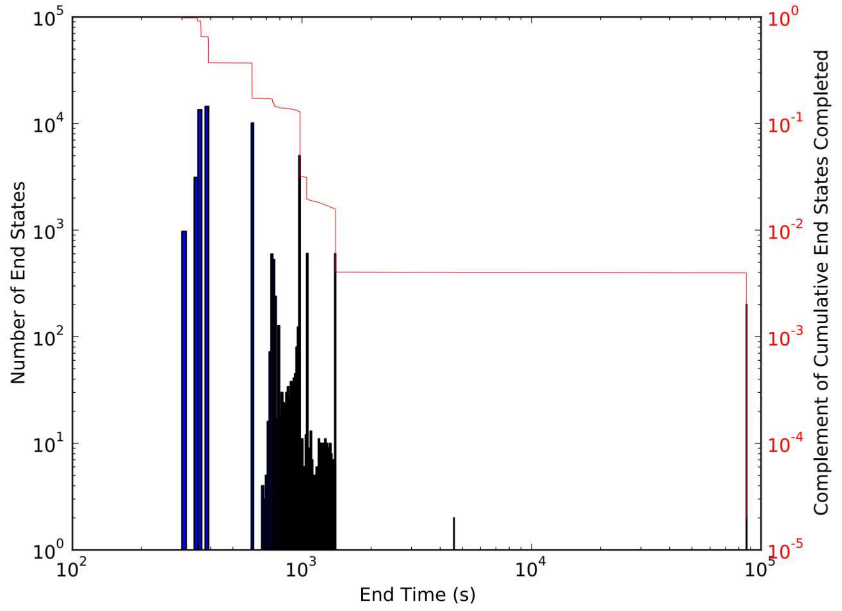

Figure 1: Progress of ADAPT End States

The DET associated with Figure 1 represents a situation where ADAPT has finished running 217,365 of 480,301 recognized branches which gives little insight into its progress. The final simulation time in this example DET is 1 days or 86,400 seconds. Figure 1 illustrates that approximately 200 branches

[5] In this context, an end state is a DET branch that has no children. If the branch is currently running, it may eventually spawn child branches and cease to be an end state.

have reached this final time while tens of thousands of end states have progressed less than 2,000 seconds. The red line in Figure 1 offers a clearer view of the fraction of currently identified branches that have reached any point in simulation time. In this test case, the bulk of the branching conditions occur before a simulation time of 1,000 seconds. This view allows the analyst to know that on the order of 10% of branches have passed that point and may be expected to produce fewer new branches as the DET progresses.

In Section 4.4, Figures 3 and 4 provide further insight into the progress of a DET with regards to the number of jobs[6] running and finished. These figures may be used to tune parameters within ADAPT to the computational environment.

4. HPC OPERATION

ADAPT users have encountered situations where a DET grows too large to be finished in a practical amount of time by the computer system being used to generate it due to combinatorial explosion of the state space [24, 25, 26]. Some approaches to remedy this situation are pruning the DET of similar branches as it grows [27] or reducing the number of branching conditions or sampled values of parameters. This section presents an approach that may be used to greatly expand the computational resources available to ADAPT within the Sandia National Laboratories (SNL) environment and others where HPCs are available.

HPCs are large computer clusters which may comprise hundreds or thousands of individual machines. For comparison, ADAPT is typically run at SNL on a small locally-administered cluster made up of 11 machines. Because HPCs typically have many users, they often require the use of highly formal job scheduling systems. ADAPT was not previously capable of using SNL's HPCs because it was not compatible with the formal job scheduling system used at SNL, Slurm [28]. Slurm assumes control of the compute nodes in a cluster and manages their allocation between submitted jobs of differing requirements and priority. The new ADAPT module Parallel Evaluation of Reliability Scenarios for Estimation of Uncertainty and Sensitivity (PERSEUS) adds the capability to ADAPT to launch and retrieve jobs using HPCs under the Slurm scheduler. This capability includes the selection of one HPC from multiple options within the same computer network.

The interface of PERSEUS with ADAPT and general mode of operation are diagrammed in Figure 2. The ADAPT script that runs for each branch (the wrapper) is modified to watch for a token file (`Magic File`) that indicates whether PERSEUS is to be used to assist completion of the DET. If not, the simulator is run normally on the assigned ADAPT host. Otherwise, a token file (`Queue File`) is created and the wrapper pauses at the stage where it would normally run the simulator. The wrapper will wait infinitely for the token `Final File` to appear which indicates that PERSEUS has executed the simulator and returned the outputs to the directory on the ADAPT host.

The *perseus-watcher* tool is explained in detail in Section 4.1. *perseus-wrapper* is explained in Section 4.2. When a simulator job is returned to the ADAPT host having reached a stopping point, ADAPT handles it as though it had run on the assigned host. This action may result in new branches being added to the DET if a branching condition was reached. The new branches may later be run either on a traditional ADAPT host or on the HPCs through PERSEUS. A test case is briefly introduced in Section 4.3 and the impact on the time required to run it is evaluated in Section 4.4.

[6] In ADAPT, a job is an attempt to run a branch on a computation host.

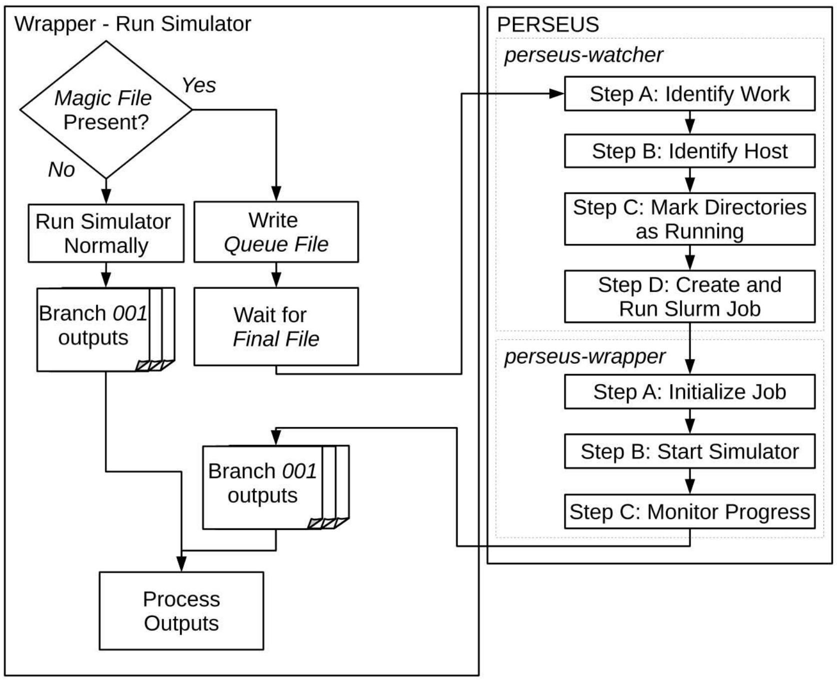

Figure 2: PERSEUS ADAPT Diagram of Operation

4.1. PERSEUS Watcher

The *perseus-watcher* tool watches directories on the ADAPT host for the `Queue File` to identify work to be done. Directories on the ADAPT host are evaluated for their PERSEUS status through signal files. A `Queue File` is written by ADAPT to indicate that the ADAPT job is to be handled by PERSEUS. *perseus-watcher* writes the `Running File` when an ADAPT job directory has been bundled into a Slurm job[7]. Jobs are submitted to Slurm via a script that contains a requested time limit and the work to be done during that time period. The job is added to a queue and is executed when resources are available. The requested time limit is a user input to *perseus-watcher*.

perseus-wrapper writes the `Final File` (see Section 4.2) if an ADAPT job finishes during its PERSEUS run rather than being closed at the end of the Slurm job period. ADAPT job directories that are intended for PERSEUS but are not running and not finished are added to the queue. Once all directories have been evaluated, a decision is made based on whether any directories have been added to the queue. If the queue is empty, the cycle begins again after a wait. If the queue has directories in it, a PERSEUS job package is created and control moves to Step B (see Figure 2). This job package is used throughout PERSEUS to define a Slurm job.

Once work is identified by *perseus-watcher*, a host must be found. The SNL HPCs are queried according to a user-adjustable order of preference to find a suitable host with open capacity. The first HPC in the preference list that has idle capacity is selected as the target. The target HPC is queried to evaluate the number of processors (cores) on its nodes. An option exists in PERSEUS to launch a Slurm job with fewer directories than cores on the target machine. By default, PERSEUS will wait to

[7] Although PERSEUS has been built around the Slurm job scheduler, adaptation to other HPC schedulers is not expected to be difficult. Most major HPC schedulers have similar requirements for submission and monitoring of jobs.

have enough ADAPT job directories to fill a node before launching. The target information is added to the job package and control moves to Step C (see Figure 2).

Once a set of ADAPT job directories and a host have been identified, the job directories are marked on the ADAPT host as running. Step C in Figure 2 is implemented so that *perseus-watcher* will not attempt to launch ADAPT job directories multiple times in quick succession. The `Running File` is detected in Step A and prevents the directory from being added to the queue.

In Step D (see Figure 2), a Slurm job is created and executed on the HPC host. The job package is used to create a script to run under Slurm. Both the job package and the Slurm script are sent to the target HPC and the Slurm script is executed. After this point, *perseus-watcher* has no further interaction with this particular job package. Individual ADAPT job directories may rejoin the queue later and become part of new job packages. Control is passed back to Step A in Figure 2 for the cycle to begin anew.

4.2. PERSEUS Wrapper

Each Slurm job executes *perseus-wrapper* (see Figure 2) which runs on the HPC target machine and manages the running of the simulator. The Slurm job first copies inputs from the ADAPT host (Step A in Figure 2). Next, each simulator job is started and runs until it reaches an ADAPT branching condition, the end of the simulation, or a simulator failure (Step B in Figure 2). *perseus-wrapper* monitors the progress of the simulator until 5 minutes remain in the Slurm job (Step C in Figure 2). At that point, any instances of the simulator still running are stopped and their outputs are returned to the ADAPT host to be run further.

4.3. Test Application

A test case was run to demonstrate the feasibility of running ADAPT in an HPC environment. The test case is a DET that explores the uncertainties relating to a Pressurized Water Reactor (PWR) ISLOCA [9, 22]. In this case, isolation valves are inadvertently opened between a low pressure and a high pressure system in a nuclear power plant. The DET explores the impact of potential mitigative actions and the radiation doses operators are expected to incur while performing the actions. This test case uses both the MELCOR and RADTRAD simulator codes under ADAPT.

The test case was run first solely on a small cluster at SNL and then to a similar level of progress (see Section 3.4) on the SNL HPCs using ADAPT with PERSEUS. The small SNL cluster is composed of 11 nodes with similar hardware and software to the SNL HPCs which have approximately 4,000 nodes between them. Only MELCOR branches were designated for execution on the HPCs. This is because RADTRAD runs significantly faster than MELCOR and the execution of those branches was not expected to benefit significantly from such a running scheme. Therefore, RADTRAD branches were run locally on the SNL small cluster. The physical results from this test case are presented in [22].

4.4. Preliminary Results

Figure 3 shows the progress of the locally-run ADAPT test case (see Section 4.3). For approximately 2 weeks it was allowed to run on all 132 cores of the small cluster. For another two weeks it was restricted to 55 cores in order to share resources with another ADAPT experiment. This pattern of resource allocation is fairly typical for small clusters. ADAPT cycles continuously to launch new jobs to replace those that have finished. The number of new jobs to launch is calculated such that if the cycle were to complete immediately all cores would be occupied. A fairly constant number of running branches was maintained within the time span of each core limit particularly with the limit of 55 cores. The locally-run experiment completed 66,076 jobs after running for 27.5 days.

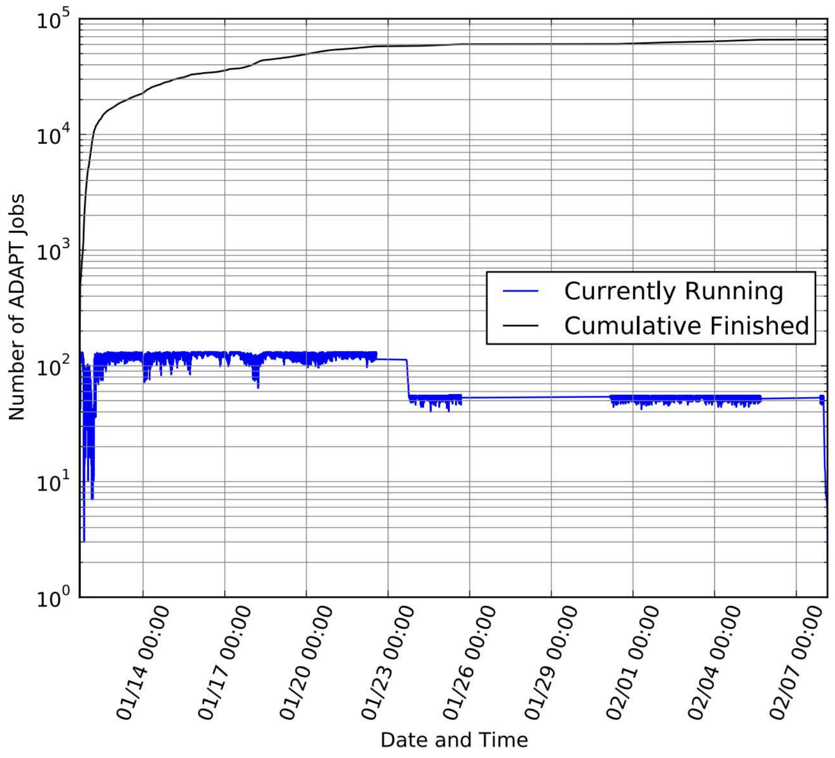

Figure 3: Progress of Locally-Run ADAPT Experiment

Figure 4 shows the progress of the test case run on the HPCs. This ADAPT experiment was started on the local cluster as can be seen in the extreme left side of Figure 4 where the number of running jobs generally remains below 100. After approximately 2,000 jobs were finished, PERSEUS was activated by creating the `Magic File` (see Figure 2). The creation of the `Magic File` instructed ADAPT jobs running MELCOR to enter the PERSEUS-linked behavior shown in Figure 2. The number of cores available to ADAPT was increased to 3,000. This number will be further refined in the future by identifying limitations of the hardware of the local cluster with regards to disk and network traffic.

In comparison to the locally-run experiment in Figure 3, the HPC-run experiment in Figure 4 shows significantly more variability in the number of jobs running at any given time. It should be noted that in this context the number of jobs running is relative to ADAPT. While PERSEUS-enabled jobs are waiting for the `Final File` to appear they are marked as running under ADAPT when in reality they may be either running on an HPC or waiting in a queue on an HPC. This experiment reached the same level of progress as that of Figure 3 (66,076 jobs completed) in 4.7 days. During this period the SNL HPCs had excess capacity and the number of jobs running was limited by the rate of ADAPT in launching new jobs. When launching a job, ADAPT makes calls to the database, copies files between computation nodes, and executes remote commands. This can be a costly process if many jobs are to be launched.

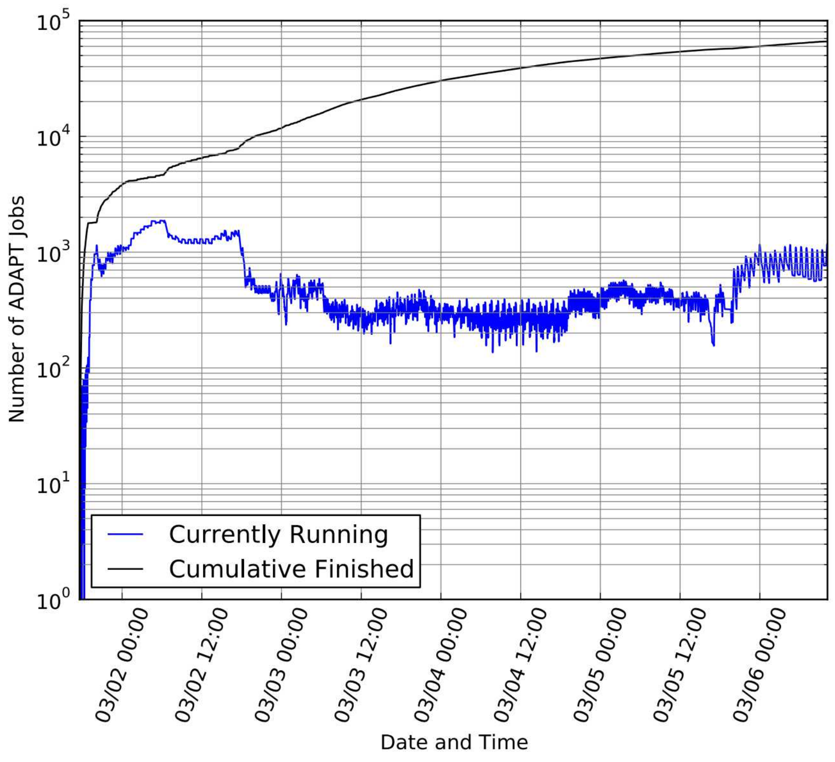

Figure 4: Progress of HPC-Run ADAPT Experiment

5. CONCLUSION

A number of improvements to the ADAPT DET driver have been presented in this paper. The first significant effect of these changes is an increased capability to generate DETs. Experiments may be run faster by spreading the workload across multiple HPCs allowing a greater fraction of the uncertainty space to be handled in the same wall time. Experiments may also leverage multiple specialized full-scale codes within a single DET in order to more faithfully represent the relevant phenomena. Outputs may be gathered in parallel and retained for later use and post-processing functions have been refactored to reduce the time required to run. These enhancements combine to increase the practical size and richness of the data set that is generated by ADAPT.

Enhancements to the capability of ADAPT to analyze the generated data set are summarized in Section 2. These include a platform for the calculation of dynamic importance measures as well as tools to interrogate a focused section of the DET. The DET generation improvements combined with data analysis tools allow ADAPT to consider a greater portion of the parameter space and allow the analyst to discover insights with greater precision and faster than before.

Acknowledgements

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories under a partnership with The Ohio State University Nuclear Engineering Program. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

References

[1] A. Hakobyan, T. Aldemir, R. Denning, S. Dunagan, D. Kunsman, B. Rutt, and U. Catalyurek, "Dynamic Generation of Accident Progression Event Trees," *Nuclear Engineering and Design*, vol. 238, pp. 3457–3467, Dec 2008.

[2] U. Catalyurek, B. Rutt, K. Metzroth, A. Hakobyan, T. Aldemir, R. Denning, S. Dunagan, and D. Kunsman, "Development of a code-agnostic computational infrastructure for the dynamic generation of accident progression event trees," *Reliability Engineering & System Safety*, vol. 95, pp. 278–294, Mar 2010.

[3] Z. Jankovsky and M. Denman, "Recent Developments in the ADAPT Discrete Dynamic Event Tree Framework," SAND2016-7680PE, Sandia National Laboratories, Albuquerque, NM, August 2016.

[4] L. Humphries, B. Beeny, F. Gelbard, D. Louie, and J. Phillips, "MELCOR Computer Code Manuals - Vol. 1: Primer and User's Guide - Version 2.2.9541 2017," SAND2017-0455O, Sandia National Laboratories, Albuquerque, NM, January 2017.

[5] A. Brunett, R. Denning, and T. Aldemir, "A Reassessment of Low Probability Containment Failure Modes and Phenomena in a Long-Term Station Blackout," *Nuclear Technology*, vol. 186, no. 2, pp. 198–215, 2014.

[6] "RELAP5-3D Code Manual Volume II: User's Guide and Input Requirements," INEEL-EXT-98-00834 Rev. 2.3, Idaho National Laboratory, Idaho Falls, ID, April 2005.

[7] T. Fanning, A. Brunett, and T. Sumner, "The SAS4A/SASSYS-1 Safety Analysis Code System," ANL/NE-16/19, Argonne National Laboratory, Argonne, IL, March 2017.

[8] Z. K. Jankovsky, M. R. Denman, and T. Aldemir, "Dynamic Event Tree Analysis with the SAS4A/SASSYS-1 Safety Analysis Code," *Annals of Nuclear Energy*, vol. 115C, pp. 55–72, 2018.

[9] T. Wheeler, M. Denman, R. Williams, N. Martin, and Z. Jankovsky, "Nuclear Power Plant Cyber Security Discrete Dynamic Event Tree Analysis (LDRD 17-0958) FY17 Report," SAND2017-10307, Sandia National Laboratories, Albuquerque, NM, September 2017.

[10] C. Picoco, T. Aldemir, and V. Rychkov, "Dynamic Event Tree Generation with RAVEN-MAAP5 using Finite State Machine System Models," in *ANS PSA 2017 International Topical Meeting on Probabilistic Safety Assessment and Analysis*, (Pittsburgh, PA), September 2017.

[11] B. Cohn, R. Denning, T. Aldemir, J. Hur, and H. Sezen, "Surrogate Model Selection in RAVEN for Seismic Dynamic PRA/PSA," in *ANS PSA 2017 International Topical Meeting on Probabilistic Safety Assessment and Analysis*, (Pittsburgh, PA), September 2017.

[12] M. A. Diaconeasa and A. Mosleh, "The ADS-IDAC Dynamic Platform with Dynamically Linked System Fault Trees," in *ANS PSA 2017 International Topical Meeting on Probabilistic Safety Assessment and Analysis*, (Pittsburgh, PA), September 2017.

[13] M. Kloos and J. Peschke, "MCDET: A Probabilistic Dynamics Method Combining Monte Carlo Simulation with the Discrete Dynamic Event Tree Approach," *Nuclear Science and Engineering*, vol. 153, no. 2, pp. 137–156, 2006.

[14] M. van der Borst and H. Schoonakker, “An Overview of PSA Importance Measures,” *Reliability Engineering & System Safety*, vol. 72, pp. 241–245, June 2001.

[15] T. Tyrväinen, “Risk importance measures in the dynamic flowgraph methodology,” *Reliability Engineering & System Safety*, vol. 118, pp. 35–50, October 2013.

[16] E. Borgonovo, “A new uncertainty importance measure,” *Reliability Engineering & System Safety*, vol. 92, no. 6, pp. 771–784, 2007.

[17] Z. Jankovsky, M. Denman, and T. Aldemir, “Dynamic Importance Measures in the ADAPT Framework,” in *Transactions of the American Nuclear Society*, vol. 115, (Las Vegas, NV), pp. 799–802, American Nuclear Society, November 2016.

[18] Y. Chang and A. Mosleh, “A Dynamic PRA of a Nuclear Power Plant using ADS-RELAP-5 Integrated Software,” in *Probabilistic Safety Assessment and Management* (A. Mosleh and R. Bari, eds.), London: Springer, 1998.

[19] A. Alfonsi, C. Rabiti, D. Maljevoic, D. Mandelli, and J. Cogliati, “Enhancements to the RAVEN code in FY16,” INL/EXT-16-40094, Idaho National Laboratory, Idaho Falls, ID, September 2016.

[20] Z. Jankovsky, M. Denman, and T. Aldemir, “Extension of the ADAPT Framework for Multiple Simulators,” in *Transactions of the American Nuclear Society*, vol. 115, (Las Vegas, NV), pp. 557–560, American Nuclear Society, November 2016.

[21] D. M. Osborn, T. Aldemir, R. Denning, and D. Mandelli, “Seamless Level 2/Level 3 Dynamic Probabilistic Risk Assessment Clustering,” in *ANS PSA 2013 International Topical Meeting on Probabilistic Safety Assessment and Analysis*, (Columbia, SC), September 2013.

[22] Z. Jankovsky, M. Denman, and T. Aldemir, “A Dynamic Coupled-Code Assessment of Mitigation Actions in an Interfacing System Loss of Coolant Accident,” in *Proceedings of the International Conference on Probabilistic Safety Assessment and Management (PSAM 14)*, (Los Angeles, California), September 2018.

[23] Z. Jankovsky, M. Denman, and T. Aldemir, “Conditional Tree Reduction in the ADAPT Framework,” in *Transactions of the American Nuclear Society*, vol. 115, (Las Vegas, NV), pp. 553–556, American Nuclear Society, November 2016.

[24] A. Hakobyan, *Severe Accident Analysis using Dynamic Accident Progression Event Trees*. PhD thesis, The Ohio State University, 2006.

[25] K. Metzroth, *A Comparison of Dynamic and Classical Event Tree Analysis for Nuclear Power Plant Probabilistic Safety/Risk Assessment*. PhD thesis, 2011.

[26] D. M. Osborn, *Seamless Level 2/Level 3 Probabilistic Risk Assessment using Dynamic Event Tree Analysis*. PhD thesis, 2013.

[27] N. S. Martin, M. R. Denman, and T. A. Wheeler, “Pruning of Discrete Dynamic Event Trees Using Density Peaks and Dynamic Time Warping,” in *Transactions of the American Nuclear Society*, vol. 115, (Las Vegas, NV), pp. 783–786, American Nuclear Society, Nov 2016.

[28] “SLURM Reference Manual,” UCRL-WEB-201386, Lawrence Livermore National Laboratory, Livermore, CA, 2006.